Group : 26 TITLE : University Management System

Suchitra Sahu
Suchetana Mukherjee
Ashrita Vaka Naidu
Shaurya Agarwal
Aasheel Dave

i)Minimum 3 table creation and insertion of atleast 5 tuples in each table .

```sql
-- Table creation
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    FirstName VARCHAR(15),
    LastName VARCHAR(15),
    Age INT,
    DepartmentID INT
);

CREATE TABLE Courses (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(50),
    DepartmentID INT
);

CREATE TABLE Grades (
    GradeID INT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    Grade DECIMAL(5, 3),
    CONSTRAINT FK_Student FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    CONSTRAINT FK_Course FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
```

```sql
INSERT INTO Students (StudentID, FirstName, LastName, Age, DepartmentID)
VALUES
    (1, 'John', 'Doe', 20, 1);
INSERT INTO Students (StudentID, FirstName, LastName, Age, DepartmentID)
VALUES    (2, 'Jane', 'Smith', 22, 2);
INSERT INTO Students (StudentID, FirstName, LastName, Age, DepartmentID)
VALUES    (3, 'Bob', 'Johnson', 21, 1);
INSERT INTO Students (StudentID, FirstName, LastName, Age, DepartmentID)
VALUES    (4, 'Alice', 'Williams', 23, 2);
INSERT INTO Students (StudentID, FirstName, LastName, Age, DepartmentID)
VALUES(5, 'Charlie', 'Brown', 19, 3);
--
INSERT INTO Courses (CourseID, CourseName, DepartmentID)
VALUES
    (101, 'Mathematics', 1);
INSERT INTO Courses (CourseID, CourseName, DepartmentID)
VALUES    (102, 'Physics', 2);
INSERT INTO Courses (CourseID, CourseName, DepartmentID)
VALUES    (103, 'Computer Science', 3);
INSERT INTO Courses (CourseID, CourseName, DepartmentID)
VALUES    (104, 'History', 1);
INSERT INTO Courses (CourseID, CourseName, DepartmentID)
VALUES    (105, 'Biology', 2);
--
INSERT INTO Grades (GradeID, StudentID, CourseID, Grade)
VALUES
    (1, 1, 101, 9.5);
INSERT INTO Grades (GradeID, StudentID, CourseID, Grade)
VALUES    (2, 2, 102, 8.0);
INSERT INTO Grades (GradeID, StudentID, CourseID, Grade)
VALUES    (3, 3, 103, 7.5);
INSERT INTO Grades (GradeID, StudentID, CourseID, Grade)
VALUES    (4, 4, 104, 8.0);
INSERT INTO Grades (GradeID, StudentID, CourseID, Grade)
VALUES    (5, 5, 105, 9.5);
```

TABLE 1: STUDENTS

STUDENTID	FIRSTNAME	LASTNAME	AGE	DEPARTMENTID
1	John	Doe	20	1
2	Jane	Smith	22	2
3	Bob	Johnson	21	1
4	Alice	Williams	23	2
5	Charlie	Brown	19	3

TABLE 2:COURSES

COURSEID	COURSENAME	DEPARTMENTID
101	Mathematics	1
102	Physics	2
103	Computer Science	3
104	History	1
105	Biology	2

TABLE 3: GRADES

GRADEID	STUDENTID	COURSEID	GRADE
1	1	101	9.5
2	2	102	8
3	3	103	7.5
4	4	104	8
5	5	105	9.5

(ii) Query for updating the tuples values based on criteria mentioned

```
67   UPDATE Students
68   SET Age = 26
69   WHERE Age > 23;
70   --
71   UPDATE Courses
72   SET CourseName = 'Advanced Mathematics'
73   WHERE CourseID = 101;
74   --
75   UPDATE Grades
76   SET Grade = 9.0
77   WHERE StudentID = 2 AND CourseID = 102;
78   --
79   UPDATE Courses
80   SET departmentid = 4
81   WHERE CourseID = 101;
82   --
83   UPDATE students
84   SET firstname = 'Joe'
85   WHERE lastname = 'Smith';
86   --
```

A) UPDATE Students
SET Age = 26
WHERE Age > 23;

STUDENTID	FIRSTNAME	LASTNAME	AGE	DEPARTMENTID
1	John	Doe	20	1
2	Jane	Smith	26	2
3	Bob	Johnson	26	1
4	Alice	Williams	26	2
5	Charlie	Brown	19	3

B) UPDATE Courses
SET CourseName = 'Advanced Mathematics'
WHERE CourseID = 101;

COURSEID	COURSENAME	DEPARTMENTID
101	Advanced Mathematics	1
102	Physics	2
103	Computer Science	3
104	History	1
105	Biology	2

C) UPDATE Grades
SET Grade = 9.0
WHERE StudentID = 2 AND CourseID = 102;

GRADEID	STUDENTID	COURSEID	GRADE
1	1	101	9.5
2	2	102	9
3	3	103	7.5
4	4	104	8
5	5	105	9.5

D) UPDATE Courses
SET departmentid = 4
WHERE CourseID = 101;

COURSEID	COURSENAME	DEPARTMENTID
101	Advanced Mathematics	4
102	Physics	2
103	Computer Science	3
104	History	1
105	Biology	2

E) UPDATE students
SET firstname = 'Joe'
WHERE lastname = 'Smith';

STUDENTID	FIRSTNAME	LASTNAME	AGE	DEPARTMENTID
1	John	Doe	20	1
2	Joe	Smith	26	2
3	Bob	Johnson	26	1
4	Alice	Williams	26	2
5	Charlie	Brown	19	3

iii) Query to retrieve the tuples based on GROUP BY , ORDER BY and HAVING clauses

QUERY 1:

```
SELECT CourseID, CourseName, DepartmentID
FROM Courses
ORDER BY CourseName;
```

Output:

COURSEID	COURSENAME	DEPARTMENTID
101	Advanced Mathematics	4
105	Biology	2
103	Computer Science	3
104	History	1
102	Physics	2

QUERY 2:

```
SELECT DepartmentID, COUNT(*) AS StudentCount
FROM Students
GROUP BY DepartmentID
ORDER BY StudentCount DESC
LIMIT 1;
```

Output:

DEPARTMENTID	STUDENTCOUNT
1	2
2	2
3	1

QUERY 3:

```
SELECT g.CourseID, c.CourseName, AVG(g.Grade) AS AvgGrade
FROM Grades g
INNER JOIN Courses c ON g.CourseID = c.CourseID
GROUP BY g.CourseID, c.CourseName
HAVING  AVG(g.Grade) > 7;
```

Output:

COURSEID	COURSENAME	AVGGRADE
101	Advanced Mathematics	9.5
102	Physics	9
105	Biology	9.5
103	Computer Science	7.5
104	History	8

QUERY 4:

```
SELECT DepartmentID, AVG(Age) AS AvgAge
FROM Students
GROUP BY DepartmentID
HAVING AVG(Age) > 22;
```

Output :

DEPARTMENTID	AVGAGE
2	22.5

QUERY 5:

```
SELECT StudentID, FirstName, LastName, Age
FROM Students
WHERE Age > (SELECT AVG(Age) FROM Students);
```

Output:

STUDENTID	FIRSTNAME	LASTNAME	AGE
2	Joe	Smith	22
4	Alice	Williams	23

QUERY 6:

```
SELECT g.CourseID, c.CourseName, COUNT(*) AS HighScorers
FROM Grades g
INNER JOIN Courses c ON g.CourseID = c.CourseID
WHERE g.Grade > 8
GROUP BY g.CourseID, c.CourseName
ORDER BY HighScorers DESC;
```

Output:

COURSEID	COURSENAME	HIGHSCORERS
105	Biology	1
102	Physics	1
101	Advanced Mathematics	1

IV) Queries to retrieve the tuples based on any two joins present in SQL

SUBQUERY 1:

```
---- Join Students with Grades to Get Student Grades:
SELECT s.FirstName, s.LastName, c.CourseName, g.Grade
FROM Students s
JOIN Grades g ON s.StudentID = g.StudentID
JOIN Courses c ON g.CourseID = c.CourseID;
```

OUTPUT:

FIRSTNAME	LASTNAME	COURSENAME	GRADE
John	Doe	Mathematics	9.5
Jane	Smith	Physics	8
Bob	Johnson	Computer Science	7.5
Alice	Williams	History	8
Charlie	Brown	Biology	9.5

SUBQUERY 2:

```
--Join Courses with Grades to Get Average Course Grades:
SELECT c.CourseName, AVG(g.Grade) AS AvgGrade
FROM Courses c
LEFT JOIN Grades g ON c.CourseID = g.CourseID
GROUP BY c.CourseName;
```

OUTPUT:

COURSENAME	AVGGRADE
Mathematics	9.5
Biology	9.5
History	8
Physics	8
Computer Science	7.5

SUBQUERY 3:

```sql
--Join Students with Courses and Filter by Department:
SELECT s.FirstName, s.LastName, c.CourseName
FROM Students s
JOIN Grades g ON s.StudentID = g.StudentID
JOIN Courses c ON g.CourseID = c.CourseID
WHERE s.DepartmentID = 1;
```

OUTPUT:

FIRSTNAME	LASTNAME	COURSENAME
John	Doe	Mathematics
Bob	Johnson	Computer Science

SUBQUERY 4:

```
--Join Students with Courses to Get Enrollment Count by Course:
SELECT c.CourseName, COUNT(g.StudentID) AS EnrollmentCount
FROM Courses c
LEFT JOIN Grades g ON c.CourseID = g.CourseID
GROUP BY c.CourseName;
```

OUTPUT:

COURSENAME	ENROLLMENTCOUNT
Mathematics	1
Biology	1
History	1
Physics	1
Computer Science	1

SUBQUERY 5:

```
--Join Courses with Departments and Calculate Highest Course Grade:
SELECT s.FirstName, s.LastName, MAX(g.Grade) AS HighestGrade
FROM Students s
JOIN Grades g ON s.StudentID = g.StudentID
GROUP BY s.FirstName, s.LastName;
```

OUTPUT:

FIRSTNAME	LASTNAME	HIGHESTGRADE
Charlie	Brown	9.5
Alice	Williams	8
Jane	Smith	8
John	Doe	9.5
Bob	Johnson	7.5

SUBQUERY 6:

```
--Join Courses with Departments and Calculate Lowest Course Grade:
SELECT s.FirstName, s.LastName, MIN(g.Grade) AS LowestGrade
FROM Students s
JOIN Grades g ON s.StudentID = g.StudentID
GROUP BY s.StudentID, s.FirstName, s.LastName;
```

OUTPUT:

FIRSTNAME	LASTNAME	LOWESTGRADE
Bob	Johnson	7.5
Alice	Williams	8
John	Doe	9.5
Jane	Smith	8
Charlie	Brown	9.5

v) Queries to retrieve the tuples based on atleast two aggregate functions.

QUERY 1:

```
SELECT C.CourseID, C.CourseName, COUNT(G.StudentID) AS CourseCount, MAX(S.Age) AS MaxAge
FROM Courses C
LEFT JOIN Grades G ON C.CourseID = G.CourseID
LEFT JOIN Students S ON G.StudentID = S.StudentID
GROUP BY C.CourseID, C.CourseName
ORDER BY C.CourseID;
```

Output:

COURSEID	COURSENAME	COURSECOUNT	MAXAGE
101	Mathematics	1	20
102	Physics	1	22
103	Computer Science	1	21
104	History	1	23
105	Biology	1	19

QUERY 2:

```
SELECT s.StudentID, s.FirstName, s.LastName, SUM(g.Grade) AS TotalGrade, MAX(g.Grade) AS MaxGrade
FROM Students s
INNER JOIN Grades g ON s.StudentID = g.StudentID
GROUP BY s.StudentID, s.FirstName, s.LastName;
```

Output:

STUDENTID	FIRSTNAME	LASTNAME	TOTALGRADE	MAXGRADE
1	John	Doe	9.5	9.5
2	Jane	Smith	8	8
4	Alice	Williams	8	8
5	Charlie	Brown	9.5	9.5
3	Bob	Johnson	7.5	7.5

QUERY 3:

```
SELECT DepartmentID, COUNT(*) AS StudentCount, SUM(Age) AS TotalAge
FROM Students
GROUP BY DepartmentID;
```

Output:

DEPARTMENTID	STUDENTCOUNT	TOTALAGE
1	2	41
2	2	45
3	1	19

QUERY 4:

```
SELECT DepartmentID, AVG(Age) AS AverageAge, MIN(Age) AS MinAge
FROM Students
GROUP BY DepartmentID;
```

Output:

DEPARTMENTID	AVERAGEAGE	MINAGE
1	20.5	20
2	22.5	22
3	19	19

QUERY 5:

```
SELECT c.CourseID, c.CourseName, AVG(g.Grade) AS AverageGrade, MIN(g.Grade) AS MinGrade
FROM Courses c
LEFT JOIN Grades g ON c.CourseID = g.CourseID
GROUP BY c.CourseID, c.CourseName;
```

Output:

COURSEID	COURSENAME	AVERAGEGRADE	MINGRADE
101	Mathematics	9.5	9.5
102	Physics	8	8
105	Biology	9.5	9.5
103	Computer Science	7.5	7.5
104	History	8	8

vi) Subqueries to retrieve the tuples

(FROM THE “Students” TABLE)

SUBQUERY 1:

```
--to retrieve students who belong to the same department as "John Doe."
SELECT StudentID, FirstName, LastName, DepartmentID
FROM Students
WHERE DepartmentID = (SELECT DepartmentID FROM Students WHERE FirstName = 'John' AND LastName = 'Doe');
```

Output:

STUDENTID	FIRSTNAME	LASTNAME	DEPARTMENTID
1	John	Doe	1
3	Bob	Johnson	1

SUBQUERY 2:

```
--to retrieve students of above average age
SELECT StudentID, FirstName, LastName, Age
FROM Students
WHERE Age > (SELECT AVG(Age) FROM Students);
```

Output:

STUDENTID	FIRSTNAME	LASTNAME	AGE
2	Joe	Smith	22
4	Alice	Williams	23

(FROM THE “Courses” TABLE)

SUBQUERY 3:

```
--retrieve courses with the same department id as physics
SELECT CourseID, CourseName
FROM Courses
WHERE DepartmentID = (SELECT DepartmentID FROM Courses WHERE CourseID = 102);
```

Output:

COURSEID	COURSENAME
102	Physics
105	Biology

SUBQUERY 4:

```
--to retrieve courses with average grade greater than 8
SELECT CourseID, CourseName
FROM Courses
WHERE CourseID IN (
    SELECT CourseID
    FROM Grades
    GROUP BY CourseID
    HAVING AVG(Grade) > 8.0
);
```

Output:

COURSEID	COURSENAME
105	Biology
101	Advanced Mathematics
102	Physics

(FROM THE “Grades” TABLE)

SUBQUERY 5:

```
--To retrieve highest grades in each course
SELECT StudentID, CourseID, Grade
FROM Grades g1
WHERE Grade = (
    SELECT MAX(Grade)
    FROM Grades g2
    WHERE g1.CourseID = g2.CourseID
);
```

Output:

STUDENTID	COURSEID	GRADE
1	101	9.5
2	102	9
3	103	7.5
4	104	8
5	105	9.5

SUBQUERY 6:

```
--To retrieve grades lower than average
SELECT StudentID, CourseID, Grade
FROM Grades
WHERE Grade < (SELECT AVG(Grade) FROM Grades);
```

Output:

STUDENTID	COURSEID	GRADE
3	103	7.5
4	104	8

GitHub link :

https://github.com/APUNJIA/DBMS-Projects-3-1