UNIVERSITY DATABASE MANAGEMENT

DBMS MINI PROJECT REPORT

Submitted by
NANDHINI RAJAN
(231801115)
NITHESHA.J.S
(231801120)
PAVITHRA.S
(231801123)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE, ANNA UNIVERSITY, CHENNAI: 602 105

MAY 2024

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI : 602105

BONAFIDE CERTIFICATE

Certified that this report title **"UNIVERSITY DATABASE MANAGEMENT"** is the Bonafide work of the students who carried out the mini project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Hod Name with designation
HEAD OF THE DEPARTMENT,
Professor,
Department of AI&DS, Rajalakshmi Engineering College Chennai – 602 105.

SIGNATURE

Supervisor name with designation
SUPERVISOR,
Assistant Professor,
Department of AI&DS,
Rajalakshmi
Engineering College,
Chennai – 602 105.

Submitted for the DBMS Mini project review held on________

Internal Examiner

External Examiner

Project Idea

This project is for the implementation of an efficient management system for university databases.

The idea is to create a set of tables containing student, instructor and other necessary details, insert record values and manipulate as required.

Idea Implementation

The program creates tables for a university database and inserts sample data into those tables. It also includes several queries to retrieve information from the database. Below is a breakdown of the script:

Tables Created:

1. **Students:**
   - StudentID (Primary Key)
   - Name
   - DateOfBirth
   - Address
2. **Courses:**
   - CourseID (Primary Key)
   - CourseName
   - Credits
3. **Departments:**
   - DepartmentID (Primary Key)
   - DepartmentName
4. **Faculty:**
   - FacultyID (Primary Key)
   - FirstName
   - LastName
5. **Enrollments:**
   - EnrollmentID (Primary Key)
   - StudentID (Foreign Key referencing Students)
   - CourseID (Foreign Key referencing Courses)
   - EnrollmentDate
6. **Instructors:**
   - InstructorID (Primary Key)
   - FirstName
   - LastName
   - DepartmentID (Foreign Key referencing Departments)
7. **Classrooms:**
   - ClassroomID (Primary Key)
   - RoomNumber
   - Building
   - Capacity
8. **ClassSchedule:**
   - ScheduleID (Primary Key)
   - CourseID (Foreign Key referencing Courses)
   - InstructorID (Foreign Key referencing Instructors)
   - ClassroomID (Foreign Key referencing Classrooms)

- DayOfWeek

9. **Grades:**
    - GradeID (Primary Key)
    - EnrollmentID (Foreign Key referencing Enrollments)
    - GradeValue
10. **Events:**
    - EventID (Primary Key)
    - EventName
    - EventDate
    - Location

Data Insertion:

- Five records are inserted into each table with sample data.

Queries Executed:

1. Retrieve the names of students along with the courses they are enrolled in.
2. Retrieve the count of students in each department.
3. Retrieve the schedule of classes for a specific instructor.
4. Retrieve the average grade for each course.
5. Retrieve students with their course enrollments and grades (if available).
6. Update the address of a specific student.
7. Delete a specific event.
8. List the top three courses with the highest enrollment.
9. Create a stored procedure to calculate the average grade for a student.
10. Example usage of the CalculateAverageGrade procedure.
11. Retrieve the schedule of classes for a specific day (Monday).

Source Code

-- Create Tables For The University Database

```
CREATE TABLE Students (
  StudentID INT PRIMARY KEY,
  Name VARCHAR(50),
  DateOfBirth DATE,
  Address VARCHAR(100)
);
```

```
Table created.
```

```
CREATE TABLE Courses (
   CourseID INT PRIMARY KEY,
   CourseName VARCHAR(100),
   Credits INT
);
```

```
Table created.
```

```
CREATE TABLE Departments (
   DepartmentID INT PRIMARY KEY,
   DepartmentName VARCHAR(100)
);
```

```
Table created.
```

```
CREATE TABLE Faculty (
   FacultyID INT PRIMARY KEY,
   FirstName
   VARCHAR(50),
   LastName VARCHAR(50)
);
```

```
Table created.
```

```
CREATE TABLE Enrollments (
    EnrollmentID INT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    EnrollmentDate DATE,
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
```

```
Table created.
```

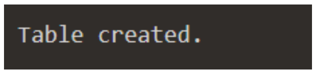
```
CREATE TABLE Instructors (
    InstructorID INT PRIMARY
    KEY, FirstName VARCHAR(50),
    LastName VARCHAR(50),
    DepartmentID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);
```

```
Table created.
```

```
CREATE TABLE Classrooms (
    ClassroomID INT PRIMARY KEY,
    RoomNumber VARCHAR(20),
    Building VARCHAR(50),
    Capacity INT
);
```

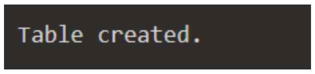
```
Table created.
```

```
CREATE TABLE ClassSchedule (
   ScheduleID INT PRIMARY KEY,
   CourseID INT,
   InstructorID INT,
   ClassroomID INT,
   DayOfWeek VARCHAR(10),
   FOREIGN KEY (CourseID) REFERENCES Courses(CourseID),
   FOREIGN KEY (InstructorID) REFERENCES Instructors(InstructorID),
   FOREIGN KEY (ClassroomID) REFERENCES Classrooms(ClassroomID)
);
```



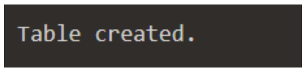
```
Table created.
```

```
CREATE TABLE Grades (
   GradeID INT PRIMARY KEY,
   EnrollmentID INT,
   GradeValue
   VARCHAR(2),
   FOREIGN KEY (EnrollmentID) REFERENCES Enrollments(EnrollmentID)
);
```



```
Table created.
```

```
CREATE TABLE Events (
   EventID INT PRIMARY KEY,
   EventName
   VARCHAR(100), EventDate
   DATE,
   Location VARCHAR(100)
);
```



```
Table created.
```

-- Insert five records for each and every table

-- Insert values into Students table

INSERT ALL

INTO Students (StudentID, Name, DateOfBirth, Address) VALUES (1, ‘John Doe’, TO_DATE(‘1990- 05-15’, ‘YYYY-MM-DD’), ‘123 Main St’)

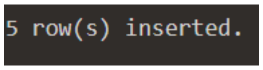
INTO Students (StudentID, Name, DateOfBirth, Address) VALUES (2, ‘Jane Smith’, TO_DATE(‘1992-08-22’, ‘YYYY-MM-DD’), ‘456 Oak St’)

INTO Students (StudentID, Name, DateOfBirth, Address) VALUES (3, ‘Bob Johnson’, TO_DATE(‘1995-03-10’, ‘YYYY-MM-DD’), ‘789 Maple St’)

INTO Students (StudentID, Name, DateOfBirth, Address) VALUES (4, ‘Alice Williams’, TO_DATE(‘1993-11-18’, ‘YYYY-MM-DD’), ‘321 Pine St’)

INTO Students (StudentID, Name, DateOfBirth, Address) VALUES (5, ‘Charlie Brown’, TO_DATE(‘1998-07-05’, ‘YYYY-MM-DD’), ‘654 Cedar St’)

SELECT * FROM dual;



```
5 row(s) inserted.
```

-- Insert values into Courses table

INSERT ALL

INTO Courses (CourseID, CourseName, Credits) VALUES (101, ‘Introduction to Computer Science’, 3)

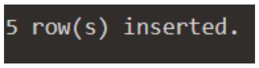
INTO Courses (CourseID, CourseName, Credits) VALUES (102, ‘Database Management’, 4)

INTO Courses (CourseID, CourseName, Credits) VALUES (103, ‘Advanced Mathematics’,

3) INTO Courses (CourseID, CourseName, Credits) VALUES (104, ‘History of Art’, 3)

INTO Courses (CourseID, CourseName, Credits) VALUES (105, ‘Physics 101’, 4)

SELECT * FROM dual;



```
5 row(s) inserted.
```

-- Insert values into Departments table

INSERT ALL

INTO Departments (DepartmentID, DepartmentName) VALUES (1, ‘Computer Science’)

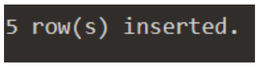
INTO Departments (DepartmentID, DepartmentName) VALUES (2, ‘Mathematics’)

INTO Departments (DepartmentID, DepartmentName) VALUES (3, ‘Art History’)

INTO Departments (DepartmentID, DepartmentName) VALUES (4, ‘Physics’)

INTO Departments (DepartmentID, DepartmentName) VALUES (5, ‘Business Administration’)

SELECT * FROM dual;



```
5 row(s) inserted.
```

-- Insert values into Faculty table

INSERT ALL

INTO Faculty (FacultyID, FirstName, LastName) VALUES (101, ‘Michael’, ‘Smith’)

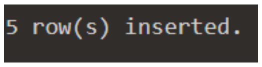
INTO Faculty (FacultyID, FirstName, LastName) VALUES (102, ‘Emily’, ‘Jones’)

INTO Faculty (FacultyID, FirstName, LastName) VALUES (103, ‘Daniel’, ‘Brown’)

INTO Faculty (FacultyID, FirstName, LastName) VALUES (104, ‘Jennifer’, ‘Clark’)

INTO Faculty (FacultyID, FirstName, LastName) VALUES (105, ‘Robert’, ‘Taylor’)

SELECT * FROM dual;



```
5 row(s) inserted.
```

-- Insert values into Enrollments table

INSERT ALL

INTO Enrollments (EnrollmentID, StudentID, CourseID, EnrollmentDate) VALUES (1, 1, 101, TO_DATE(‘2023-01-15’, ‘YYYY-MM-DD’))

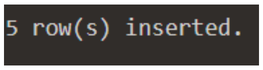
INTO Enrollments (EnrollmentID, StudentID, CourseID, EnrollmentDate) VALUES (2, 2, 102, TO_DATE(‘2023-02-20’, ‘YYYY-MM-DD’))

INTO Enrollments (EnrollmentID, StudentID, CourseID, EnrollmentDate) VALUES (3, 3, 103, TO_DATE(‘2023-03-25’, ‘YYYY-MM-DD’))

INTO Enrollments (EnrollmentID, StudentID, CourseID, EnrollmentDate) VALUES (4, 4, 104, TO_DATE(‘2023-04-10’, ‘YYYY-MM-DD’))

INTO Enrollments (EnrollmentID, StudentID, CourseID, EnrollmentDate) VALUES (5, 5, 105, TO_DATE(‘2023-05-05’, ‘YYYY-MM-DD’))

SELECT * FROM dual;



```
5 row(s) inserted.
```

-- Insert values into Instructors table

INSERT ALL

INTO Instructors (InstructorID, FirstName, LastName, DepartmentID) VALUES (201, ‘David’, ‘Miller’, 1)

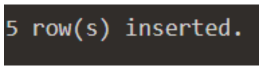
INTO Instructors (InstructorID, FirstName, LastName, DepartmentID) VALUES (202, ‘Linda’, ‘White’, 2)

INTO Instructors (InstructorID, FirstName, LastName, DepartmentID) VALUES (203, ‘William’, ‘Jones’, 3)

INTO Instructors (InstructorID, FirstName, LastName, DepartmentID) VALUES (204, ‘Karen’, ‘Smith’, 4)

INTO Instructors (InstructorID, FirstName, LastName, DepartmentID) VALUES (205, ‘Richard’, ‘Brown’, 5)

SELECT * FROM dual;



```
5 row(s) inserted.
```

-- Insert values into Classrooms table

INSERT ALL

INTO Classrooms (ClassroomID, RoomNumber, Building, Capacity) VALUES (301, ‘101’, ‘Science Building’, 50)

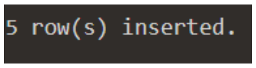
INTO Classrooms (ClassroomID, RoomNumber, Building, Capacity) VALUES (302, ‘201’, ‘Engineering Building’, 40)

INTO Classrooms (ClassroomID, RoomNumber, Building, Capacity) VALUES (303, ‘301’, ‘Arts Building’, 30)

INTO Classrooms (ClassroomID, RoomNumber, Building, Capacity) VALUES (304, ‘401’, ‘History Building’, 35)

INTO Classrooms (ClassroomID, RoomNumber, Building, Capacity) VALUES (305, ‘501’, ‘Business Building’, 45)

SELECT * FROM dual;



```
5 row(s) inserted.
```

-- Insert values into ClassSchedule table

INSERT ALL

INTO ClassSchedule (ScheduleID, CourseID, InstructorID, ClassroomID, DayOfWeek) VALUES (1, 101, 201, 301, ‘Monday’)

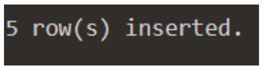
INTO ClassSchedule (ScheduleID, CourseID, InstructorID, ClassroomID, DayOfWeek) VALUES (2, 102, 202, 302, ‘Tuesday’)

INTO ClassSchedule (ScheduleID, CourseID, InstructorID, ClassroomID, DayOfWeek) VALUES (3, 103, 203, 303, ‘Wednesday’)

INTO ClassSchedule (ScheduleID, CourseID, InstructorID, ClassroomID, DayOfWeek) VALUES (4, 104, 204, 304, ‘Thursday’)

INTO ClassSchedule (ScheduleID, CourseID, InstructorID, ClassroomID, DayOfWeek) VALUES (5, 105, 205, 305, ‘Friday’)

SELECT * FROM dual;



```
5 row(s) inserted.
```

-- Insert values into Grades table

INSERT ALL

INTO Grades (GradeID, EnrollmentID, GradeValue) VALUES (1, 1, ‘A’)

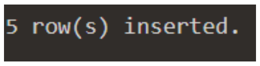
INTO Grades (GradeID, EnrollmentID, GradeValue) VALUES (2, 2, ‘B+’)

INTO Grades (GradeID, EnrollmentID, GradeValue) VALUES (3, 3, ‘A-‘)

INTO Grades (GradeID, EnrollmentID, GradeValue) VALUES (4, 4, ‘C+’)

INTO Grades (GradeID, EnrollmentID, GradeValue) VALUES (5, 5, ‘B’)

SELECT * FROM dual;



```
5 row(s) inserted.
```

-- Insert values into Events table

INSERT ALL

INTO Events (EventID, EventName, EventDate, Location) VALUES (1, ‘University Fair’, TO_DATE(‘2023-06-15’, ‘YYYY-MM-DD’), ‘Main Auditorium’)

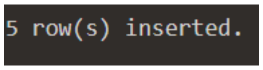
INTO Events (EventID, EventName, EventDate, Location) VALUES (2, ‘Career Workshop’, TO_DATE(‘2023-07-20’, ‘YYYY-MM-DD’), ‘Conference Room’)

INTO Events (EventID, EventName, EventDate, Location) VALUES (3, ‘Art Exhibition’, TO_DATE(‘2023-08-25’, ‘YYYY-MM-DD’), ‘Art Gallery’)

INTO Events (EventID, EventName, EventDate, Location) VALUES (4, ‘Physics Symposium’, TO_DATE(‘2023-09-10’, ‘YYYY-MM-DD’), ‘Physics Lab’)

INTO Events (EventID, EventName, EventDate, Location) VALUES (5, ‘Business Networking’, TO_DATE(‘2023-10-05’, ‘YYYY-MM-DD’), ‘Business Center’)

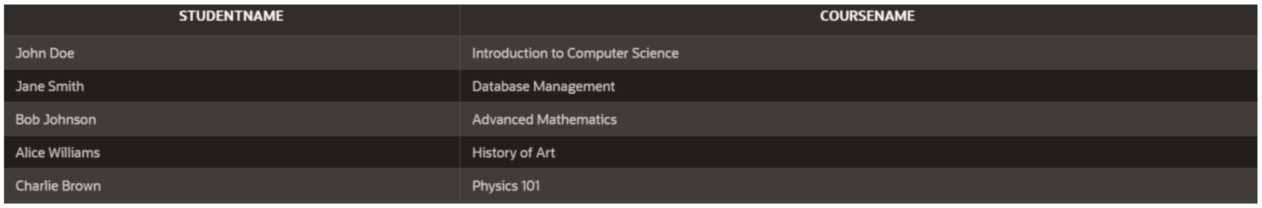
SELECT * FROM dual;



```
5 row(s) inserted.
```

-- Retrieve the names of students along with the courses they are enrolled in

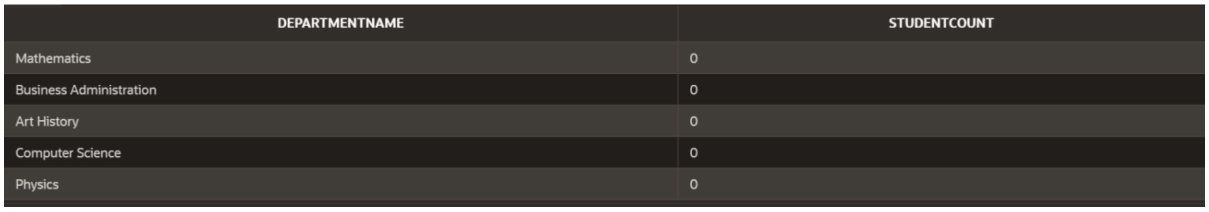
```
SELECT Students.Name AS StudentName, Courses.CourseName
FROM Students
JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
JOIN Courses ON Enrollments.CourseID = Courses.CourseID;
```



STUDENTNAME	COURSENAME
John Doe	Introduction to Computer Science
Jane Smith	Database Management
Bob Johnson	Advanced Mathematics
Alice Williams	History of Art
Charlie Brown	Physics 101

-- Retrieve the count of students in each department

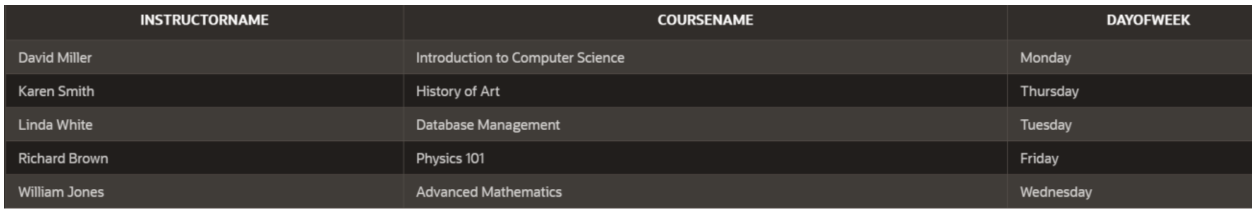
```
SELECT Departments.DepartmentName, COUNT(Students.StudentID) AS StudentCount
FROM Departments
LEFT JOIN Instructors ON Departments.DepartmentID = Instructors.DepartmentID
LEFT JOIN Courses ON Instructors.InstructorID = Courses.CourseID
LEFT JOIN Enrollments ON Courses.CourseID = Enrollments.CourseID
LEFT JOIN Students ON Enrollments.StudentID = Students.StudentID
GROUP BY Departments.DepartmentName;
```



DEPARTMENTNAME	STUDENTCOUNT
Mathematics	0
Business Administration	0
Art History	0
Computer Science	0
Physics	0

-- Retrieve the schedule of classes for a specific instructor

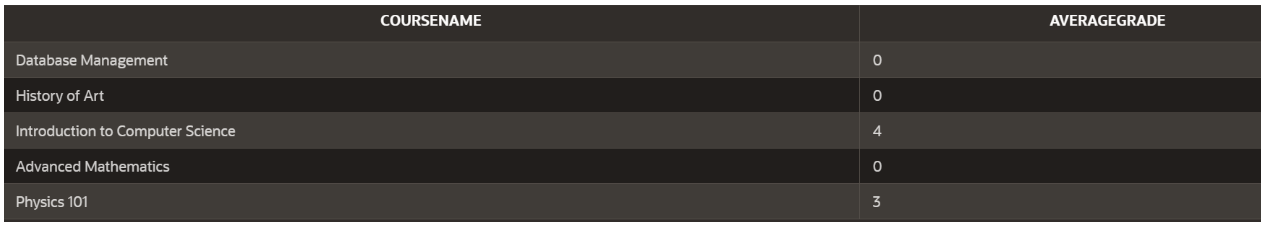
```
SELECT Instructors.FirstName || ‘ ‘ || Instructors.LastName AS InstructorName,
    Courses.CourseName, ClassSchedule.DayOfWeek
FROM Instructors
JOIN ClassSchedule ON Instructors.InstructorID = ClassSchedule.InstructorID
JOIN Courses ON ClassSchedule.CourseID = Courses.CourseID
ORDER BY InstructorName, DayOfWeek;
```



INSTRUCTORNAME	COURSENAME	DAYOFWEEK
David Miller	Introduction to Computer Science	Monday
Karen Smith	History of Art	Thursday
Linda White	Database Management	Tuesday
Richard Brown	Physics 101	Friday
William Jones	Advanced Mathematics	Wednesday

-- Retrieve the average grade for each course

```
SELECT Courses.CourseName, AVG(CASE WHEN Grades.GradeValue = ‘A’ THEN
                    4 WHEN Grades.GradeValue = ‘B’ THEN 3
                    WHEN Grades.GradeValue = ‘C’ THEN 2
                    WHEN Grades.GradeValue = ‘D’ THEN 1
                    ELSE 0 END) AS AverageGrade
FROM Courses
LEFT JOIN Enrollments ON Courses.CourseID = Enrollments.CourseID
LEFT JOIN Grades ON Enrollments.EnrollmentID = Grades.EnrollmentID
GROUP BY Courses.CourseName;
```



COURSENAME	AVERAGEGRADE
Database Management	0
History of Art	0
Introduction to Computer Science	4
Advanced Mathematics	0
Physics 101	3

-- Retrieve students with their course enrollments and grades (if available):

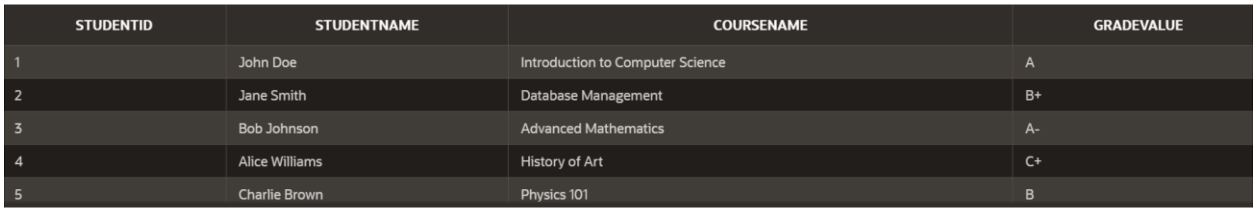
```
SELECT Students.StudentID, Students.Name AS StudentName, Courses.CourseName,
Grades.GradeValue
```

FROM Students

LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID

LEFT JOIN Courses ON Enrollments.CourseID = Courses.CourseID

LEFT JOIN Grades ON Enrollments.EnrollmentID = Grades.EnrollmentID;



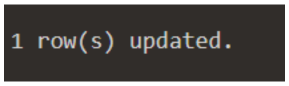
STUDENTID	STUDENTNAME	COURSENAME	GRADEVALUE
1	John Doe	Introduction to Computer Science	A
2	Jane Smith	Database Management	B+
3	Bob Johnson	Advanced Mathematics	A-
4	Alice Williams	History of Art	C+
5	Charlie Brown	Physics 101	B

-- Update the address of a specific student

UPDATE Students

SET Address = ‘456 Oak St’

WHERE StudentID = 1;

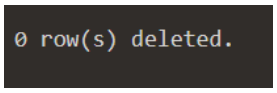


```
1 row(s) updated.
```

-- Delete a specific event

DELETE FROM Events

WHERE EventID = 1;



```
0 row(s) deleted.
```

-- List the top three courses with the highest enrollment

SELECT Courses.CourseID, Courses.CourseName, COUNT(Enrollments.StudentID) AS EnrollmentCount

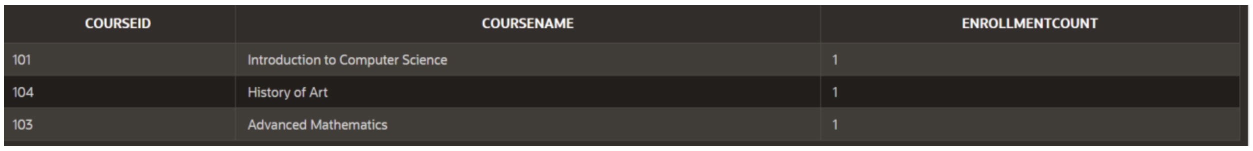
FROM Courses

LEFT JOIN Enrollments ON Courses.CourseID = Enrollments.CourseID

GROUP BY Courses.CourseID, Courses.CourseName

ORDER BY EnrollmentCount DESC

FETCH FIRST 3 ROWS ONLY;

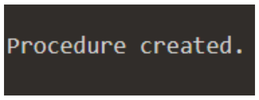


COURSEID	COURSENAME	ENROLLMENTCOUNT
101	Introduction to Computer Science	1
104	History of Art	1
103	Advanced Mathematics	1

-- Creating a stored procedure to calculate the average grade for a student

```
CREATE OR REPLACE PROCEDURE CalculateAverageGrade(
  p_StudentID IN NUMBER,
  p_AverageGrade OUT NUMBER
) AS
BEGIN
  SELECT AVG(
        CASE
          WHEN g.GradeValue = ‘A’ THEN 4.0
          WHEN g.GradeValue = ‘A-‘ THEN 3.7
          WHEN g.GradeValue = ‘B+’ THEN 3.3
          WHEN g.GradeValue = ‘B’ THEN 3.0
          WHEN g.GradeValue = ‘B-‘ THEN 2.7
          WHEN g.GradeValue = ‘C+’ THEN 2.3
          WHEN g.GradeValue = ‘C’ THEN 2.0
          WHEN g.GradeValue = ‘C-‘ THEN 1.7
          WHEN g.GradeValue = ‘D+’ THEN 1.3
          WHEN g.GradeValue = ‘D’ THEN 1.0
          ELSE 0
        END
      ) INTO p_AverageGrade
   FROM Grades g
      JOIN Enrollments e ON g.EnrollmentID = e.EnrollmentID
  WHERE e.StudentID = p_StudentID;
```

END CalculateAverageGrade;


```
Procedure created.
```

-- Example usage of the CalculateAverageGrade procedure

DECLARE

v_StudentID NUMBER := 1; -- Replace with the desired StudentID

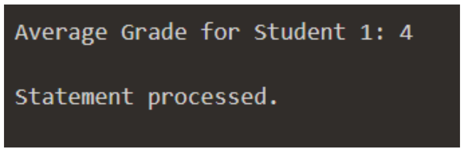
v_AverageGrade NUMBER;

BEGIN

CalculateAverageGrade(v_StudentID, v_AverageGrade);

DBMS_OUTPUT.PUT_LINE('Average Grade for Student ' || v_StudentID || ': ' || v_AverageGrade);

END;


```
Average Grade for Student 1: 4

Statement processed.
```

--Retrieve the schedule of classes for a specific day:

SELECT ClassSchedule.ScheduleID, Courses.CourseName, Instructors.FirstName, Instructors.LastName, Classrooms.RoomNumber, ClassSchedule.DayOfWeek

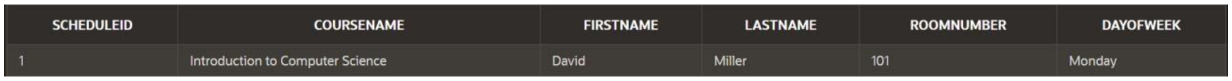
FROM ClassSchedule

JOIN Courses ON ClassSchedule.CourseID = Courses.CourseID

JOIN Instructors ON ClassSchedule.InstructorID = Instructors.InstructorID

JOIN Classrooms ON ClassSchedule.ClassroomID = Classrooms.ClassroomID

WHERE ClassSchedule.DayOfWeek = 'Monday';



SCHEDULEID	COURSENAME	FIRSTNAME	LASTNAME	ROOMNUMBER	DAYOFWEEK
1	Introduction to Computer Science	David	Miller	101	Monday

References

1. www.google.com
2. www.scribd.com
3. www.javatpoint.com