Assignment 3

Course Name: Advanced DBMS
Course Instructor: Husnain Haider

PL/SQL Assignment: University Management System

Objective

Design and implement a University Management System using PL/SQL by creating tables, inserting sample data, and writing procedures, functions, and triggers.

Part A: Database Schema and Sample Data

Create the following tables and insert the **given data**:

1. STUDENTS

Primary Key (PK): `student_id`

student_id	first_name	last_name	date_of_birth	email
101	Ayesha	Khan	2001-03-12	ayesha.k@uni.edu
102	Bilal	Ahmed	2000-07-25	bilal.ahmed@uni.edu
103	Sana	Malik	2002-01-30	sana.malik@uni.edu
104	Farhan	Raza	1999-11-15	farhan.raza@uni.edu
105	Zara	Sheikh	2001-06-20	zara.sheikh@uni.edu
106	Hamza	Qureshi	2000-02-17	hamza.q@uni.edu
107	Anam	Yousaf	2002-05-19	anam.yousaf@uni.edu
108	Imran	Shah	1998-09-22	imran.shah@uni.edu
109	Fatima	Tariq	2001-12-10	fatima.t@uni.edu
110	Ali	Rauf	2000-10-05	ali.rauf@uni.edu

2. INSTRUCTORS

Primary Key (PK): `instructor_id`

instructor_id	first_name	last_name	email
201	Usman	Iqbal	usman.iqbal@uni.edu
202	Maria	Zubair	maria.z@uni.edu
203	Kamran	Javed	kamran.javed@uni.edu
204	Lubna	Hassan	lubna.hassan@uni.edu
205	Saeed	Khan	saeed.khan@uni.edu
206	Nida	Rehman	nida.rehman@uni.edu
207	Salman	Mir	salman.mir@uni.edu
208	Saba	Haroon	saba.haroon@uni.edu
209	Faisal	Zaman	faisal.z@uni.edu
210	Hina	Shahid	hina.shahid@uni.edu

3. COURSES

Primary Key (PK): `course_id`
Foreign Keys (FKs): `instructor_id` → references `INSTRUCTORS.instructor_id`

course_id	course_name	instructor_id	credits
CSE101	Data Structures	201	3
CSE102	Web Programming	204	4
MAT101	Calculus I	202	3
MAT201	Linear Algebra	205	3
PHY101	Mechanics	203	4
PHY202	Quantum Physics	206	4

CSE201	Algorithms	207	3
CSE301	Database Systems	210	3
MAT301	Statistics	208	3
PHY301	Electromagnetism	209	3

4. ENROLLMENTS

Primary Key (PK): `enrollment_id`
Foreign Keys (FKs):

- `student_id` → references `STUDENTS.student_id`
- `course_id` → references `COURSES.course_id`

enrollment_id	student_id	course_id	semester	grade
301	101	CSE101	Fall24	A
302	102	MAT101	Fall24	B
303	103	PHY101	Fall24	A
304	104	CSE101	Fall24	B
305	105	MAT201	Fall24	A
306	106	PHY202	Fall24	C
307	107	CSE201	Fall24	B
308	108	MAT301	Fall24	A
309	109	PHY301	Fall24	B
310	110	CSE301	Fall24	A
311	101	CSE201	Spring25	B
312	101	CSE301	Fall25	A
313	102	MAT201	Spring25	A

314	102	MAT301	Fall25	A
315	103	PHY202	Spring25	B
316	103	PHY301	Fall25	A
317	104	CSE201	Spring25	B
318	104	CSE301	Fall25	A
319	105	MAT101	Spring25	A
320	105	MAT301	Fall25	B
321	106	PHY101	Spring25	B
322	106	PHY301	Fall25	A
323	107	CSE101	Spring25	A
324	107	CSE301	Fall25	A
325	108	MAT101	Spring25	B
326	108	MAT201	Fall25	A
327	109	PHY101	Spring25	A
328	109	PHY202	Fall25	B
329	110	CSE201	Spring25	A
330	110	CSE101	Fall25	A

Part B: PL/SQL Tasks

Implement the following using **procedures**, **functions**, and **triggers**:

1. Procedure: Enroll Student

Write a procedure `enroll_student` to enroll an **existing student** in an **existing course** for a given semester. The procedure should first **check whether the student and course exist** before proceeding with the enrollment. If either does not exist, handle the case appropriately (e.g., raise an error or display a message).

2. Function: Calculate GPA

Write a function `calculate_gpa(student_id)` to return the GPA based on the student's grades, using the below formula and criteria:

Formula:
GPA = SUM(grade_points * course_credits) / SUM(course_credits)

Criteria:

Grade	Grade Point
A	4.0
B	3.0
C	2.0
D	1.0
F	0.0
NULL	Ignored in GPA calculation (e.g., course in progress)

3. Trigger: Validate Grade

Create a trigger to **ensure only A, B, C, D, F, or NULL** grades are allowed in the ENROLLMENTS table.

4. Trigger: Prevent Duplicate Enrollment

Prevent a student from enrolling in the **same course and semester** more than once.

5. Procedure: Update Grade

Create a procedure `update_grade(enrollment_id, grade)` that updates a student's grade.

6. Function: Course Load

Write a function `get_credit_load(student_id, semester)` that calculates total **credits** a student is enrolled in during a semester.

7. Trigger: Instructor Course Limit

Create a trigger to **restrict an instructor** from being assigned to more than **3 active courses** at any time.

8. Procedure: Remove Student Record

Write a procedure `delete_student(student_id)` that:

- Deletes the student
- Removes all their enrollment records
- Displays a message if the student doesn't exist

9. Procedure: Print Student Transcript

Write a procedure `print_transcript(student_id)` that displays a student's academic record. For the given `student_id`, the procedure should print:

- Student name
- For each course: course title, credits, semester, and grade

You will need to join the relevant tables inside the procedure and loop through the student's enrollments to display the information clearly.

10. Procedure: List Probation Students

Write a procedure `list_probation_students` that prints the names and IDs of all students whose GPA is less than 2.0. Use the GPA calculation logic already defined to determine eligibility.

Submission Instructions

Submit:

- A `.sql` file with all queries.

Evaluation Criteria

The assignment will be evaluated through a viva, so be prepared to explain your work.