PL_SQL DEMO (Function, Procedures and Triggers)

<Follow associated script for relevant SQL commands>

*Unless otherwise stated, XYZ are the last three digits of your student number.

Task 1: Create table named *std_perf_XYZ* and populate it with the given data.

studentno	subject	ct1	ct2	ct3	ct4	attendance	term_final_70
0605001	CSE303	40	80	90	80	90	-
…	…	…	…	…	…	…	…

**all class test marks i.e. ct1, ct2, ct3, ct4 and attendance are given out of 100 marks, whereas term_final_70 is given out of 70 marks.

Task 2: Create table named *std_grade_XYZ*.

studentno	subject	class_perf_30	term_final_70	total	grade

Task 3: Now write a *function calc_grade_XYZ(total)* that returns the grade according to the BUET undergraduate grading system:

Grade	Grade Points	Numerical Markings
A+	4.0	80% and above
A	3.75	75% to below 80%
A-	3.50	70% to below 75%
B+	3.25	65% to below 70%
B	3.0	60% to below 65%
B-	2.75	55% to below 60%
C+	2.50	50% to below 55%
C	2.25	45% to below 50%
D	2.0	40% to below 45%
F	0.00	Below 40%

For the purpose of testing, use it as follows:-

Query: SELECT calc_grade_001(78) from dual;
OUTPUT: A

Task 4: Now write a procedure *calc_total_XYZ(subj)* that calculates the grades for all the students of a particular subject, the subject code i.e. CSE303 has to be given as parameter *subj*. The calculated data like

- class performance (30 marks),
- term final (70 marks directly copied),
- total (summation of the two previous terms) and

- grade (as calculated by the function done in task 3)

for each student of that particular subject are inserted into the *std_grade_XYZ* table . The calculation of class performance i.e. the value of the attribute *class_perf_30* (30 marks) is calculated just as in the undergrad studies in BUET:-

class_perf_30 = (sum of best 3 class test out of 4, effectively calculated out of 20)+(attendance effectively calculated out of 10).

So, total (100 marks) = class_perf_30 + term_final_70

To test this procedure, you have to call it explicitly to completely grade all the students of a particular subject like 'CSE303':-

```
EXEC calc_total_001 ('CSE303');
SELECT * FROM std_grade_001;
```

Which will gives the following output:

studentno	subject	class_perf_30	term_final_70	total	grade
0605001	CSE303	25.7	67	93	A+
…	…	…	…	…	…

Task 5: You have to explicitly call the above procedure after all the insertions in the *std_perf_XYZ* table has been completed. The procedure will insert relevant data into the *std_grad_XYZ* table consequently. But what about any correction in the *std_perf_XYZ* table? For example, a term final mark or class test mark is re-examined and changed; the procedure in Task 4 will be of no help (it only inserts into the table but does not update it). So define a trigger *tr_std_perf_XYZ* that automatically updates all the attributes in *std_grad_XYZ* for any change in *std_perf_XYZ*. For example:-

The term final marks of studentno 0605010 has been changed from 52 to 0 as it has been detected that the student was absent in the final exam and mark was given erroneously:

Before change in std_perf_XYZ:

studentno	subject	class_perf_30	term_final_70	total	grade
…	…	…	…	…	…
0605010	CSE303	21	52	73	A-
…	…	…	…	…	…

After change in std_perf_XYZ:

studentno	subject	class_perf_30	term_final_70	total	grade
…	…	…	…	…	…
0605010	CSE303	21	0	21	F

…	…	…	…	…	…

Task 6: Create a table *grad_stat_XYZ*

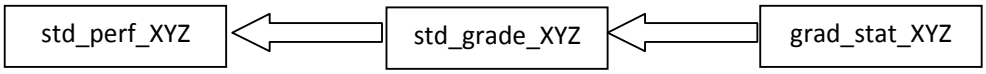
Now define-

i) a procedure that populates (insert operation) the table with computed data as processed from the table *std_grad_XYZ*.

ii) a trigger that updates the table automatically when there is any change in grade in the corresponding subject.

subject	average	minno	maxno	a_plus	a	…	f
CSE303	71.3	21	93	4	1	…	1

Have you noticed the dependency for changes between those tables:-



Task 7: Most important! Clean up all your tables, procedures, functions and triggers.

-END-