B.Sc. Engg. SWE 4th Semester

15 February 2023 (Afternoon)

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
ORGANISATION OF ISLAMIC COOPERATION (OIC)
Department of Computer Science and Engineering (CSE)

MID SEMESTER EXAMINATION
DURATION: 1 HOUR 30 MINUTES

SUMMER SEMESTER, 2021-2022
FULL MARKS: 50

CSE 4409: Database Management Systems II

Programmable calculators are not allowed. Do not write anything on the question paper.
Answer **all 3 (three)** questions. Figures in the right margin indicate full marks of questions whereas corresponding CO and PO are written within parentheses.

1. a) Consider the following code fragment in PL/SQL — 5 (CO1) (PO1)

```
CREATE OR REPLACE FUNCTION GET_STATUS(P_NAME IN VARCHAR2)
RETURN NUMBER(10,2)
IS
V_SALARY NUMBER:=0;
V_TOTAL NUMBER;
BEGIN
SELECT SALARY INTO V_SALARY
FROM EMP
WHERE NAME LIKE %P_NAME%;
P_NAME:="TEST";
V_TOTAL=V_SALARY*12;
RETURN NVL(V_TOTAL,-1);
END;
```

Code Snippet 1: Code segment for Question 1 a).

Assume the table with records exists and attribute names are correct. Your task is to identify and explain (briefly) the errors (if any) in the above code segment.

Solution:
-LIST OF ERRORS-

```
1. P_NAME:="TEST"; --THIS IS READ ONLY AND "" HAS BEEN USED
2. RETURN NUMBER(10,2) ---NO PRECISION CAN BE GIVEN HERE
3. V_TOTAL=V_SALARY*12;   --- IT SHOULD BE :=
4. SELECT SALARY INTO V_SALARY ---IT MAY RERURN MORE THAN
  --1 RECORD SINCE NAME IS NOT PK
  --AND % LIKE %  OPERATOR IS USED
```

b) State the main advantage of using `%TYPE` over the basic declaration inside any PL/SQL sub-program. — 2 (CO1) (PO1)

Solution:
If the base data type is changed the program will run without recompilation.

c) Consider a table `EMP(ID,Name,DOB,Salary,Performance)`. Now write an UPDATE SQL statement to decrease the salary of the employees by 25% who now receive greater than the existing average salary of the company. — 3 (CO1) (PO1)

Write an anonymous block that will execute the above UPDATE statement. If none of the records are affected by the statement then it will print "NO RECORDS HAVE BEEN UPDATED", otherwise it will count the total number (x) of updated records and will print the message " A TOTAL OF x RECORDS HAVE BEEN UPDATED".

Solution:
Use implicit cursor SQL, SQL%rowcount (subquery is needed)

2. Consider the following description:

Planet Mars(PM) is a large telecom company of Bangladesh. It has customers over 10 million. Only registered customers are eligible to get SIM from PM. Registration requires customer's basic information such as Name, DOB, Address. A customer may get a number of SIMs (i.e. mobile numbers) but he/she registers only once. The company runs a number of *plans* where the important information are *plan name and charge per minute.* A customer may have multiple SIMs along with its plan information. Customer's outgoing calls are logged only. Relevant information for each outgoing call is CallID, SIM no (i.e. mobile no),CallBegin (date-time of call initialization) CallEnd (date-time when the call has been finished) and charge (relevant charge for the call based on both plan and duration).

Your tasks are:

a) Create the required DDL statements to design the given system. You are free to choose attribute name and type as long as they suffice the system requirement. 5 (CO1) (PO1)

Solution:

```
Customers(CID (pk),Name,Address)
Plans(Plan,chargePerMinute)
SIMS(CID(fk),SIM(pk),Plan(fk))
CallLogs(CallID (pk),SIM(fk),CallBegin,CallEnd,CallCharge)
```

b) Write a PL/SQL function using the following guideline: 10 (CO1) (PO1)

Input: SIM no, CallBegin Time, CallEnd Time
Output: Charge calculated based on both plan and duration
Algorithm: For all charge calculation 1 minute pulse is maintained (i.e. 1 min 5 sec will be considered as 2 minutes!!!).

Solution:
Here main operator is date difference =date1-date2 which will return number of days so expression in minutes could be as follows:
Minutes:=Floor(CallEnd - CallBegin)*24*60;

Given SIM fetch its plan and then retrieve the pricing per minute from Plans table. Then calculate the charge by simple multiplication.

c) The format of the CallID is `YYYYMMDD.NNNNNNNN` where YYYYMMDD are the year, month (1 to 12) and day (day of the month) based on the CallBegin Date-time while NNNNNNNN is the incrementing numeric value. For instance the first call on January 23, 2023 will have a CallID 20230123.00000001 10 (CO1) (PO1)

Write a function in PL/SQL to generate the CallID as described and place it in a suitable trigger.

Solution:

Use `x:=to_char(dt,'DDDDMMYY')`

IF it is the first entry then:
`vID:=to_number(x||'.0000001');`
ELSE
RETURN VID+.00000001;

3. Consider the following high-level system requirement: Department of Computer Science and Engineering (CSE) of IUT has got a Merit Scholarship (MS) fund for only 2nd year students of Software Engineering. 15 (CO1) (PO1)

Entities needed:

```
Students(ID, Name, Program, Year, CGPA)
Misconducts(StudentID, date-time, description)
StudentTransctions(StudentID,date-time,Amount (paid))
```

Policy for scholarship distribution: The initial eligibility for MS is:

- Student must have a CGPA of 3.5 or above
- He/She should not be involved in any misconduct as recorded in Misconducts entity (a student with higher CGPA but having misconduct record is automatically disqualified)
- Student with higher CGPA will be given preference

The total MS amount for each year is dynamic (changes over time). So, there is no assurance how many students will avail it. In other words it is distributed as long as fund is available. Whenever a student is selected and fund is available an appropriate transaction is made in the `StudentTransctions` entity.

Your task is to create a PL/SQL function as follows:

Input: Total MS amount, per student amount
Output: Number of students who received SC, number of students who were initially selected but did not receive MS.
Algorithm: Disburse Merit Scholarship as per the given policy.

Solution:

Use explicit cursor : ensure initial selection is done correctly such as:

```
select ....
from ...
where prog='SWE' and year=2 and CGPA>=3.5 and SID NOT IN
 (select SID from misconducts)
 order by CGPA desc;
```

Now open cursor (better to use cursor for loop) and use simple logic to initiate transaction.

For return values: use return statement for the first one (use an internal counter and return it), for the second return value use OUT p.m. use explicitcursor%rowcount-rowprocessed (as first return value)