**TERM – WINTER 2023 **

| **Course & Section Code:** | **DBS501S1A** |
| --- | --- |
| **Course Name:** | **Procedure Using Oracles PL/SQL** |
| **Course Start & End Dates:** |  |
| **Q&A / Virtual Office Hour:** | **6:30 PM – 9:30 PM** |
| **Instructor Name & Email:** | **Ersan Cam: ersan.cam@senecacollege.ca** |

Lab#5 (Variables, Loops , IF THEN ELSE)

**Due Feb 20th Midnight**

**Tasks (All the tasks marks are equal)**

**Below Exercise 1 and 2 are paper base exercise meaning you answer on the paper first without executing the code.. Then you can write and test the code**

Text, letter

Description automatically generated

Text

Description automatically generated

**Preparation for Task 3**

Run this Create Table as SELECT command to create replica version of employees table

**CREATE TABLE emp**

**AS**

**SELECT \* FROM employees;**

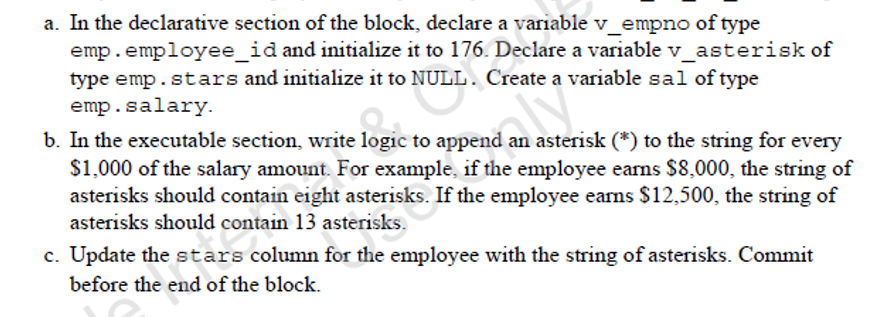
Then add one more column to this new table. New column name is stars (VARCHAR2 - 50 digit) . Run below command

**ALTER TABLE emp ADD stars VARCHAR2(50);**

SELECT emp table to see stars columns as empty.

**Task #3:**

Write PL/SQL anonyms block to fill newly added columns for each employee\_id entered by end user from screen (use &emp\_id to test one particular employee\_id)



**Hint :** Use round(salary /1000) formula to find how many starts will be added to stars column.

For example, let’s say you will test 107

*SELECT round(salary/1000)*

*FROM EMPLOYEES*

*WHERE employee\_id =* ***&empid*** *;*

107 employee id’s salary is 4830 so round will find 5 stars.

Now your job is to use one type of LOOPS and loop thru 5 times in this case and each loop iteration

UPDATE emp

SET stars = stars + ‘\*’

WHERE employee\_id= v\_\_empid ; ( or WHERE employee\_id=107)

Each time LOOP thru it will add one extra stars (\*) eventually it will be 5 star

Once your PL/SQL complete executing, run SELECT statement to show the screen shot

***SELECT \* FROM EMP WHERE employee\_id=107;***

**Task #4:**

Develop a PL/SQL block that would go through all the employees from EMPLOYEES table who work in department\_id= 90 (there is suppose to be 3 employee who work in Department 90)

IN BEGIN END END SECTION;

Execute 3 separate of SELECT statement one after the other. Each SELECT will pull first\_name , salary of each employee and store them in local variables.

SELECT LOAD INTO v\_firstname1, v\_salary1 WHERE employee\_id=100

Then second

SELECT first\_name, salary INTO v\_firstname2, v\_salary2 ….. WHERE employee\_id=101

Then third

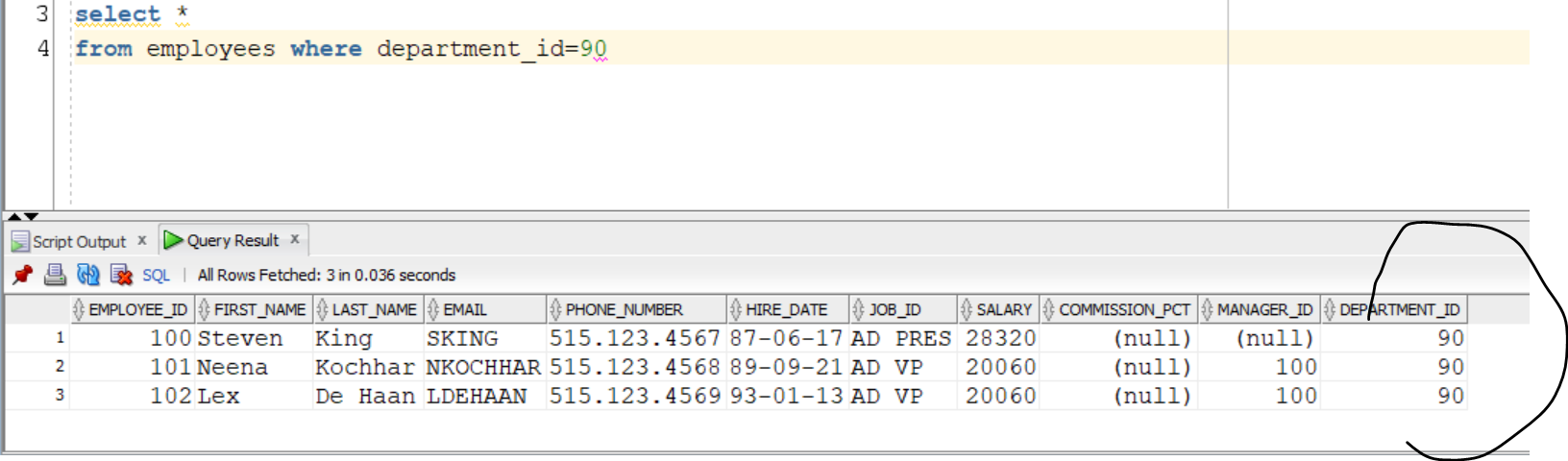
SELECT, SELECT first\_name, salary INTO v\_firstname3, v\_salary3 ….. WHERE employee\_id=102

You must now have 6 local variables. DECLARE all of them accordingly in DECLARE section and use them in BEGIN and END.

Then add another SELECT statement in BEGIN-END to find company average salary

SELECT avg(salary) INTO v\_avgsal FROM EMPLOYEES;

!!! Do not forget to create v\_avgsal local variable in DECLARE section.



Now after loading all those info to local variables,

Use IF THEN ELSE to PRINT these messages on the screen based on compare results

**Hint: You can have one big IF ELSIF ELSIF END or you can have 3 separate IF THEN ELSE END IF statetement . Up to you**

**IF v\_salary1 > v\_avgsal THEN --( note** Employee\_id = 100 salary 24000 )

‘Employee ‘ || v\_firstname1 || ‘salary ‘ || v\_salary1 || ‘ is more than average salary ‘

Or Else print (Display)

‘Employee ‘ || v\_firstname1 || ‘salary ‘ || ‘is less average salary ‘

**IF v\_salary2 > v\_avgsal THEN --( note** Employee\_id = 101 salary 17000 )

‘Employee ‘ || v\_firstname2 || ‘salary ‘ || v\_Salary2|| ‘is more than average salary ‘

Or Else print (Display)

‘Employee ‘ || v\_firstname2 || ‘salary ‘ || ‘is less than average salary ‘

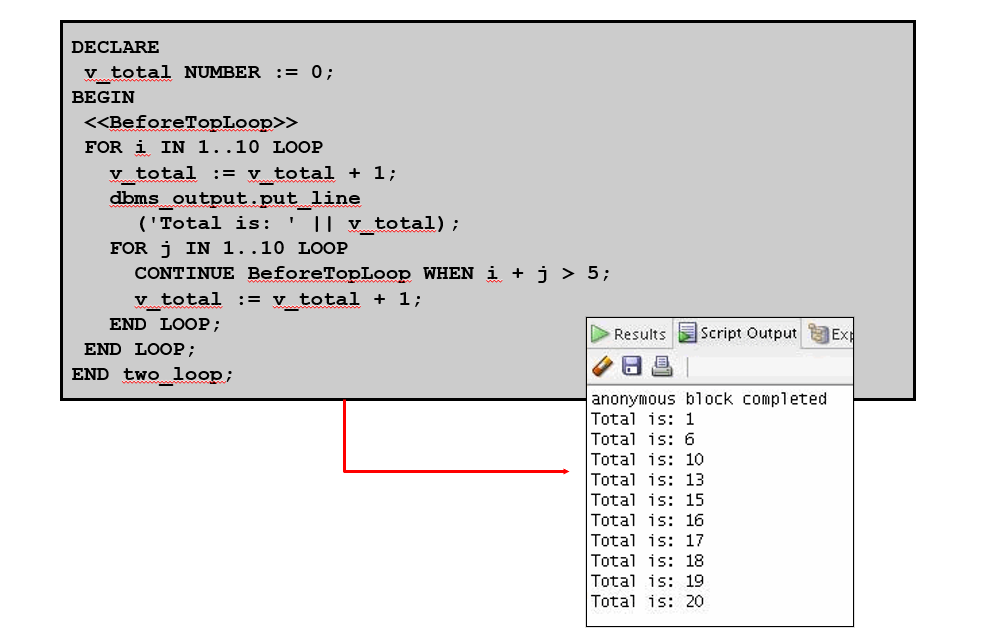
**IF v\_salary3 > v\_avgsal THEN --( note** Employee\_id = 101 salary 17000 )

DBMS\_OUTPUT ‘Employee ‘ || v\_firstname3 || ‘salary ‘ || v\_salary3 || ‘is more than average salary ‘

Or Else print (Display)

DBMS\_OUTPUT ‘Employee ‘ || v\_firstname3 || ‘salary ‘ || || v\_salary3 || ‘is less than average salary ‘

5) Investigate below code and explain how CONTINUE would impact the DBMS OUTPUT printing options. Go with each i iterations from 1 to 10 and also go with internal child FOR LOOP j from 1 to 10



DECLARE

v\_total NUMBER := 0;

BEGIN

<<BeforeTopLoop>>

FOR i IN 1..10 LOOP

v\_total := v\_total + 1;

dbms\_output.put\_line

('Total is: ' || v\_total);

FOR j IN 1..10 LOOP

CONTINUE BeforeTopLoop WHEN i + j > 5;

v\_total := v\_total + 1;

END LOOP;

END LOOP;

END two\_loop;