**TERM – SUMMER 2023**

| **Course & Section Code:** | **DBS501** |
| --- | --- |
| **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#10 (Triggers)

**Task1:**

USE LargeCo\_ORA\_Simplified script to build database…

USE **LG\_XXXXX** tables for Task 1 and 2

1. **Write a trigger to update the customer balance when an invoice is deleted from LGINVOICE table. Name the trigger trg\_updatecustbalance**

Hint: In Trigger body, update LGCUSTOMER for (WHERE) cust\_code = : old.cust\_code

Cust\_balance must be equal to cust\_balance- old.inv\_total

|  |
| --- |
| create or replace trigger trg\_updatecustbalance  after delete on lginvoice  for each row  begin  update lgcustomer  set cust\_balance = cust\_balance - :old.inv\_total  where cust\_code = :old.cust\_code;  end;    Test case  Before deleting invoice |

**Task#2**

1. **Create a trigger named trg\_line\_prod that will automatically update the product quantity on hand (**P\_QOH ) in PRODUCT TABLE **for each product sold after a new LINE row is added(inserted) to LGLINE table.**

Hint: (just high level coding. Do not quote on me )

In Trigger body, update LGPRODUCT for (WHERE) Prod\_SKU = :NEW.Prod\_SKU;

And calculate PROD\_QOH = PROD\_QOH – new quantity

|  |
| --- |
| create or replace trigger trg\_line\_prod  after insert on lgline  for each row  begin  update lgproduct  set prod\_qoh = prod\_qoh - :new.line\_qty  where prod\_sku = :new.prod\_sku;  end;    Test case    Inserted inv\_num 4000 with qty of 10    Prod\_qoh reduced to 20 |

**Task#3: (Use HR\_XXX tables)**

**Preparation for Task #3**

Please build this procedure and insert a new record with this procedure

**Step1b:**

CREATE OR REPLACE PROCEDURE new\_job

( p\_jobid IN jobs.job\_id%TYPE, p\_title IN jobs.job\_title%TYPE, v\_minsal IN jobs.min\_salary%TYPE)

IS

v\_maxsal jobs.max\_salary%TYPE := 2 \* v\_minsal;

BEGIN

INSERT INTO jobs(job\_id, job\_title, min\_salary, max\_salary) VALUES (p\_jobid, p\_title, v\_minsal, v\_maxsal);

DBMS\_OUTPUT.PUT\_LINE ('New row added to JOBS table:'); DBMS\_OUTPUT.PUT\_LINE (p\_jobid || ' ' || p\_title ||' '|| v\_minsal || ' ' || v\_maxsal);

END new\_job;

|  |
| --- |
| create or replace PROCEDURE new\_job  ( p\_jobid IN hr\_jobs.job\_id%TYPE, p\_title IN hr\_jobs.job\_title%TYPE, v\_minsal IN hr\_jobs.min\_salary%TYPE)  IS  v\_maxsal hr\_jobs.max\_salary%TYPE := 2 \* v\_minsal;  BEGIN  INSERT INTO hr\_jobs(job\_id, job\_title, min\_salary, max\_salary) VALUES (p\_jobid, p\_title, v\_minsal, v\_maxsal);  DBMS\_OUTPUT.PUT\_LINE ('New row added to JOBS table:'); DBMS\_OUTPUT.PUT\_LINE (p\_jobid || ' ' || p\_title ||' '|| v\_minsal || ' ' || v\_maxsal);  END new\_job; |

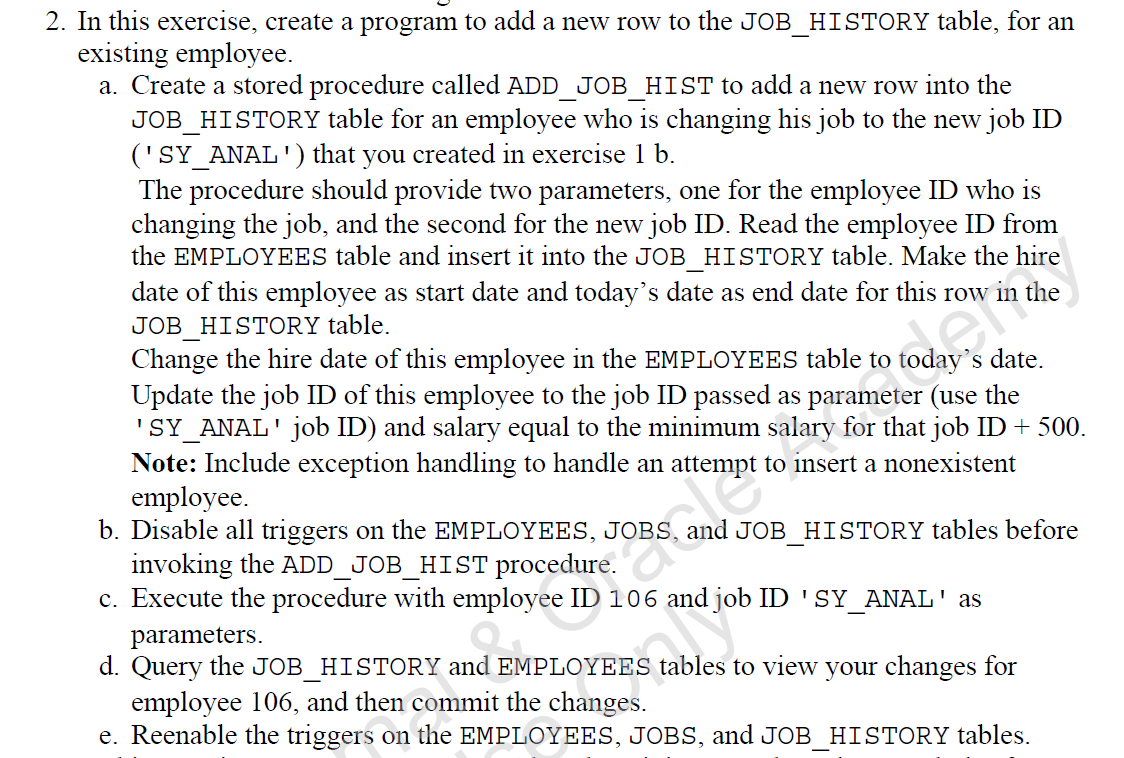
**Step1b: to execute procedure to insert new record.**

Please replace single quote ‘ ‘ from Word to your SQL Developer

EXECUTE new\_job (‘SY\_ANAL’, ‘System Analyst’,6000)

This will create a new JOB\_ID in JOBS table. So you can use this for Task 2

|  |
| --- |
|  |



|  |
| --- |
| a)  create or replace procedure add\_job\_hist  (p\_empid hr\_employees.employee\_id%type,  p\_jobid hr\_jobs.job\_id%type) as  v\_min\_salary hr\_jobs.min\_salary%type;  v\_employee\_count number(1);  begin  select count(\*) into v\_employee\_count from hr\_employees where employee\_id = p\_empid;  if v\_employee\_count = 0 then  raise\_application\_error(-20001, 'Invalid employee id.');  end if;  select min\_salary into v\_min\_salary from hr\_jobs where job\_id = p\_jobid;  insert into hr\_job\_history(employee\_id, start\_date, end\_date, job\_id, department\_id)  values(p\_empid, (select hire\_date from hr\_employees where employee\_id = p\_empid), SYSDATE, p\_jobid,  (select department\_id from hr\_employees where employee\_id = p\_empid));  update hr\_employees  set hire\_date = sysdate,  job\_id = p\_jobid,  salary = v\_min\_salary + 500  where employee\_id = p\_empid;  end add\_job\_hist;    B)    c)    d)      e) |

**Commands for 2-b (given for you)**

ALTER TABLE employees DISABLE ALL TRIGGERS;

ALTER TABLE jobs DISABLE ALL TRIGGERS;

ALTER TABLE job\_history DISABLE ALL TRIGGERS;

**Commands for 2-e (Given for you)**

ALTER TABLE employees DISABLE ALL TRIGGERS;

ALTER TABLE jobs DISABLE ALL TRIGGERS;

ALTER TABLE job\_history DISABLE ALL TRIGGERS;