Name:Mennatallah Abdelhamid

SQL and PL/SQL Labs

**PL/SQL (Day2):**

|  |  |
| --- | --- |
| *1* | *Create a procedure* ***PRINT\_EMP\_LEVEL*** *with a cursor to* ***print*** *each* ***employee name*** *besides his* ***level***  *Employee level based on his salary:*   * *above 4000 are rich* * *below 2500 are poor* * *and others are normal.* |
|  | ***create or replace procedure print\_emp\_level***  ***is***  ***cursor cur\_print is***  ***select last\_name ,salary***  ***from employees ;***  ***begin***  ***for i in cur\_print***  ***loop***  ***if i.salary> 4000 then***  ***DBMS\_OUTPUT.PUT\_LINE(i.last\_name||' is rich');***  ***elsif i.salary<255 then***  ***DBMS\_OUTPUT.PUT\_LINE(i.last\_name||' is poor');***  ***else***  ***DBMS\_OUTPUT.PUT\_LINE(i.last\_name||' is normal');***  ***end if;***  ***end loop;***  ***end;***  ***exec print\_emp\_level***  ***Mourgos is rich***  ***Nayer is normal***  ***Mikkilineni is normal***  ***Landry is normal***  ***Markle is normal***  ***Bissot is normal***  ***Atkinson is normal***  ***Marlow is normal*** |
| *2* | *Create a function* ***DISPLAY\_EMP\_INFO*** *with a* ***number*** *parameter* ***p\_id****,*  *and a* ***varchar*** *variable* ***v\_name*** *that selects the name of employee number* ***p\_id*** *into* ***v\_name****.*  *if no employees with this number, return (****no employees of this number****).* |
|  | ***create or replace function DISPLAY\_EMP\_INFO(p\_id number)***  ***return varchar2***  ***is***  ***v\_name employees.last\_name%type;***  ***begin***  ***select last\_name***  ***into v\_name***  ***from employees***  ***where employee\_id=p\_id;***  ***return v\_name;***  ***exception***  ***when no\_data\_found then***  ***return ('no employees of this number');***  ***end;***  ***SQL> exec :x:=display\_emp\_info(1);***  ***PL/SQL procedure successfully completed.***  ***SQL> print x***  ***X***  ***--------------------------------------------------------------------------------***  ***no employees of this number***  ***SQL> exec :x:=display\_emp\_info(100);***  ***PL/SQL procedure successfully completed.***  ***SQL> print x***  ***X***  ***--------------------------------------------------------------------------------***  ***King*** |
| *4* | *Create a procedure* ***DELETE\_DEPT*** *with a number parameter* ***p\_dno*** *that delete the department with this number, if the department has employees working in it, display a message (“****you should delete the employees of this department first****”). Hint (the non-predefined exception number is –2292).* |
|  | ***create or replace procedure delete\_dept(p\_dno number)***  ***is***  ***flag EXCEPTION;***  ***PRAGMA EXCEPTION\_INIT(flag,-2292);***  ***begin***  ***delete from departments***  ***where department\_id=p\_dno;***  ***exception***  ***WHEN flag THEN***  ***DBMS\_OUTPUT.PUT\_LINE('you should delete the employees of this department first');***  ***end;***  ***SQL> exec delete\_dept(20)***  ***you should delete the employees of this department first*** |
| *5* | *Changes to data are allowed on tables only during normal office working days Sunday through Thursday.*  *Create a stored procedure called* ***SECURE\_DML*** *that prevents the DML statement from executing outside of normal working days, returning the message: “****You may make changes only during working days only****.”* |
|  | ***create or replace procedure secure\_dml***  ***is***  ***begin***  ***IF (TO\_CHAR(SYSDATE,'DY') IN ('SAT','FRI')) then***  ***RAISE\_APPLICATION\_ERROR(-20500, 'You may make changes only during working days only');***  ***end if;***  ***end;*** |
| *6* | *a) Create a statement trigger* ***SECURE\_JOBS*** *on the JOBS table that calls the above procedure.* |
|  | ***create or replace trigger secure\_job***  ***BEFORE INSERT OR UPDATE OR DELETE ON jobs***  ***begin secure\_dml;***  ***end;***  ***update jobs***  ***set job\_id=500***  ***SQL> /***  ***update jobs***  ***\****  ***ERROR at line 1:***  ***ORA-20500: You may make changes only during working days only*** |
|  | *b) Test the procedure by temporarily modifying the days in the procedure and attempting to insert a new record into the JOBS table. After testing, reset the procedure days as specified before.* |
|  | *insert into jobs*  *values(700,'marketing',50,80)*  *SQL> /*  *insert into jobs*  *\**  *ERROR at line 1:*  *ORA-20500: You may make changes only during working days only* |
| *7* | *Employees should receive an automatic increase in salary if the minimum salary for a job is increased.*  *Implement this requirement through a trigger on the JOBS table.*   1. *Create a stored procedure named* ***UPD\_EMP\_SAL***   *This procedure accepts two parameters: the* ***job ID*** *for which salary has to be updated and the* ***new minimum salary*** *for this job ID.*  *The procedure update the* ***salary*** *amount of the given employee, if his salary less than the given minimum salary, then his salary will equal the given minimum salary*  *This procedure is executed from the trigger on the JOBS table.* |
|  | *create or replace procedure upd\_emp\_sal(job varchar2,min\_sal number)*  *is*  *begin*  *update employees*  *set salary=min\_sal*  *where job\_id=job*  *and salary<min\_sal;*  *end;*  */* |
|  | 1. *Create a row trigger named* ***check\_min\_sal*** *on the* ***JOBS*** *table after update the field* ***min\_salary***   *The trigger invokes the procedure* ***UPD\_EMP\_SAL*** *when the minimum salary in the JOBS table is updated for a specified job ID.* |
|  | *create or replace trigger check\_min\_sal*  *after update of min\_salary ON jobs*  *for each row*  *begin*  *upd\_emp\_sal(:new.job\_id,:old.min\_salary);*  *end;* |
|  | *c) Increase the minimum salary for the programmer job from 4,000 to 5,000* |
|  | *1 update jobs*  *2 set min\_salary=5000*  *3\* where lower(job\_id)='it\_prog'*  *SQL> /*  *1 row updated.* |
|  | *d) Employee Lorentz (employee ID 107) had a salary of less than 4,500. Verify that her salary has been increased to the new minimum of 5,000.* |
|  | *select salary*  *from employees*  *where employee\_id=107*  *SALARY*  *----------*  *5000* |
| *2* | *In a loop, use a cursor to retrieve the* ***department number*** *and the* ***department name*** *from the* ***DEPARTMENTS*** *table for those departments whose* ***DEPARTMENT\_ID*** *is less than 100.*  *Pass the* ***department number*** *to another cursor to retrieve from the* ***EMPLOYEES*** *table the details of* ***employee last name, job, hire date, and salary*** *of those employees whose* ***EMPLOYEE\_ID*** *is less than 120 and who work in that department.* |
|  |  |