

SCHOOL OF ADVANCED SCIENCES

(SAS)

**Assignment - 5**

Program: M.Sc. DATA SCIENCE

Course: Database Management Systems Lab

Submitted by: SOUMYADEEP GANGULY Course code: PMDS506P

Register number: 24MDT0082

1. **Write a PL/SQL block that computes the total salary of the employees in the employee table. (use for loop)**

DECLARE

s NUMBER := 0;

BEGIN

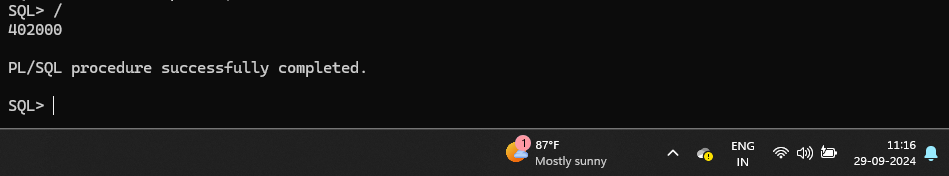
FOR i IN (SELECT \* FROM employee) LOOP

s := s + i.salary;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(s);

END;



1. **Write a PL/SQL block that checks if an employee's salary is greater than or equal to 70,000. If it is, print "High salary". Otherwise, print "Regular salary". (use if else)**

BEGIN

FOR i IN (SELECT \* FROM employee) LOOP

if i.salary >= 70000 then

dbms\_output.put\_line('High Salary');

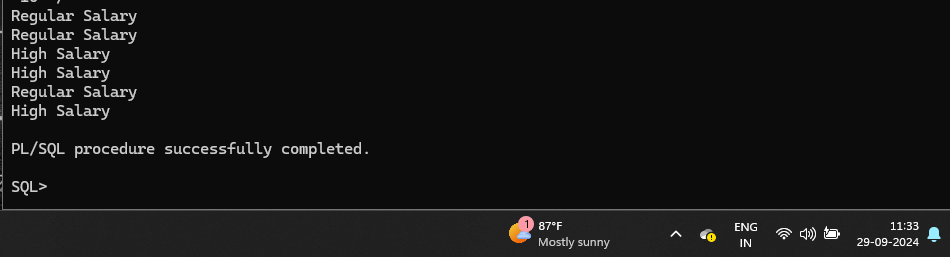
else

dbms\_output.put\_line('Regular Salary');

end if;

END LOOP;

END;

****

1. **Write a PL/SQL block that uses a FOR loop to display the salaries of employees in the 'IT' department. (for loop)**

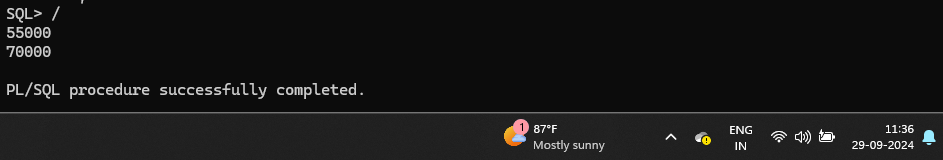
BEGIN

FOR i IN (SELECT \* FROM employee where department='IT') LOOP

dbms\_output.put\_line(i.salary);

END LOOP;

END;



1. **Write a PL/SQL block that increases the salary of all employees in IT department by 5% and prints the updated salaries. (for loop)**

DECLARE

sal NUMBER :=0;

BEGIN

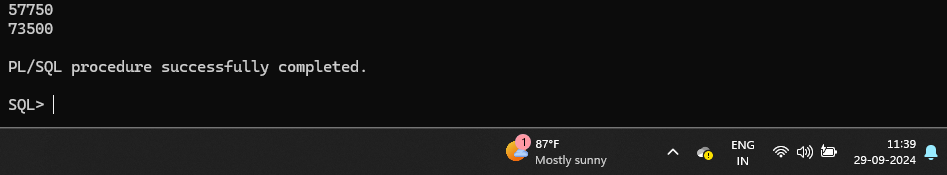
FOR i IN (SELECT \* FROM employee where department='IT') LOOP

sal := i.salary + i.salary \* 0.05;

dbms\_output.put\_line(sal);

END LOOP;

END;



1. **Categorize employees into 'High', 'Medium', or 'Low' salary brackets based on their salary. (Use case expression)**

DECLARE

sal varchar(10);

BEGIN

FOR i IN (SELECT \* FROM employee) LOOP

sal :=

case

when i.salary >= 72000 then 'High'

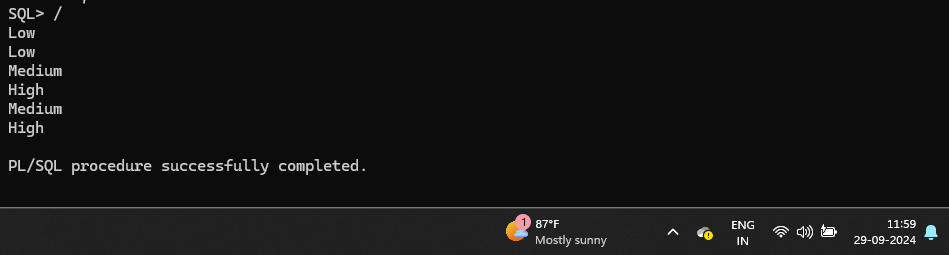
when i.salary >= 65000 and i.salary <= 70000 then 'Medium'

else 'Low'

end;

dbms\_output.put\_line(sal);

END LOOP;

END;

1. **Create a SQL query that uses a CASE expression to display different messages based on the department of employees. Display 'Welcome to HR' for employees in the 'HR' department, 'Tech Team' for employees in the 'IT' department, and 'Finance Team' for employees in the 'Finance' department.**

DECLARE

sal varchar(50);

BEGIN

FOR i IN (SELECT \* FROM employee) LOOP

sal :=

case i.department

when 'HR' then 'Welcome to HR'

when 'IT' then 'Welcome to Tech Team'

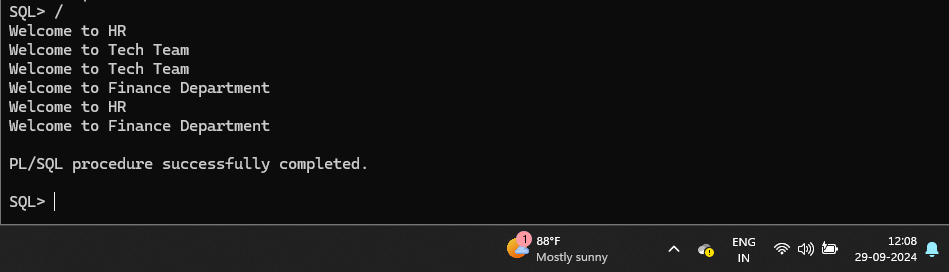
else 'Welcome to Finance Department'

end;

dbms\_output.put\_line(sal);

END LOOP;

END;



1. **Create a PL/SQL procedure increase\_salary that takes an emp\_id and a percentage increase and updates the salary of the employee by the given percentage.**

CREATE OR REPLACE PROCEDURE increase\_salary

(e\_id IN employee.emp\_id%TYPE,

percent\_increase IN NUMBER)

IS

current\_sal employee.salary%TYPE;

new\_salary employee.salary%TYPE;

BEGIN

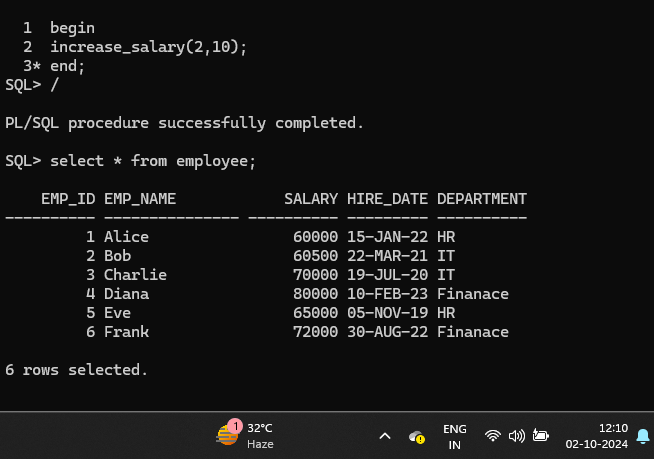
SELECT salary INTO current\_sal FROM employee WHERE emp\_id = e\_id;

new\_salary := current\_sal + current\_sal \* (percent\_increase / 100);

UPDATE employee SET salary = new\_salary WHERE emp\_id = e\_id;

END increase\_salary;

/



1. **Write a PL/SQL function get\_employee\_department that takes an emp\_id and returns the department of the employee.**

CREATE OR REPLACE FUNCTION get\_employee\_department

(e\_id NUMBER)

RETURN employee.department%TYPE

IS

emp\_dept employee.department%TYPE;

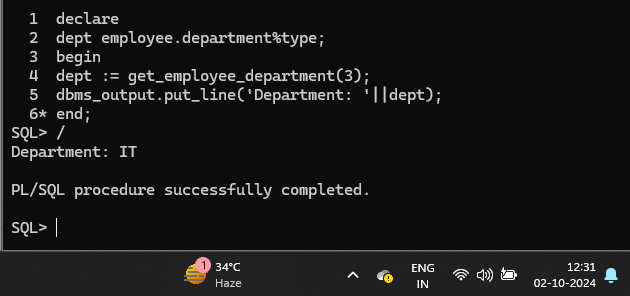
BEGIN

SELECT department INTO emp\_dept FROM employee WHERE emp\_id = e\_id;

RETURN emp\_dept;

END;/

/



1. **Create a procedure update\_salary that takes an emp\_id and a new salary and updates the employee's salary in the employee’s table.**

create or replace procedure update\_salary

(e\_id in employee.emp\_id%type,

new\_sal in employee.salary%type)

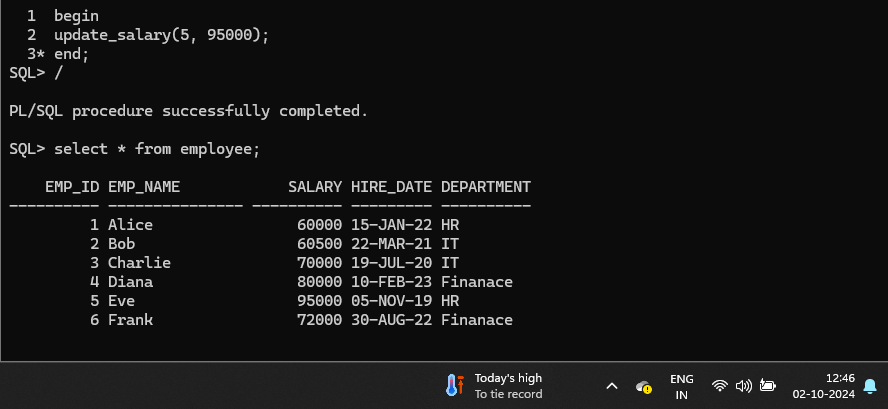
is

begin

update employee set salary= new\_sal where emp\_id = e\_id;

end update\_salary;

/



1. **Write a function calculate\_bonus that takes a salary and returns a bonus amount based on the following criteria: 10% of salary if the salary is above 70000. 5% of salary if the salary is between 60000 and 70000. 1% of salary if below 60000.**

CREATE OR REPLACE FUNCTION calculate\_bonus

(salary employee.salary%type)

RETURN employee.salary%type

IS

bonus employee.salary%type;

BEGIN

if salary > 70000 then

return salary+salary\*0.1;

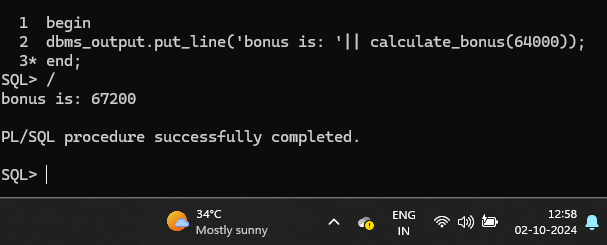
elsif salary >=60000 and salary <=70000 then

return salary+salary\*0.05;

else

return salary+salary\*0.01;

end if;



1. **Create a procedure promote\_employee that takes an emp\_id and updates the employee’s department to 'Management' if their salary is greater than or equal to 70000.**

CREATE OR REPLACE PROCEDURE promote\_employee

(e\_id IN employee.emp\_id%TYPE)

IS

current\_sal employee.salary%TYPE;

BEGIN

SELECT salary INTO current\_sal

FROM employee

WHERE emp\_id = e\_id;

IF current\_sal >= 70000 THEN

UPDATE employee SET department = 'Management' WHERE emp\_id = e\_id;

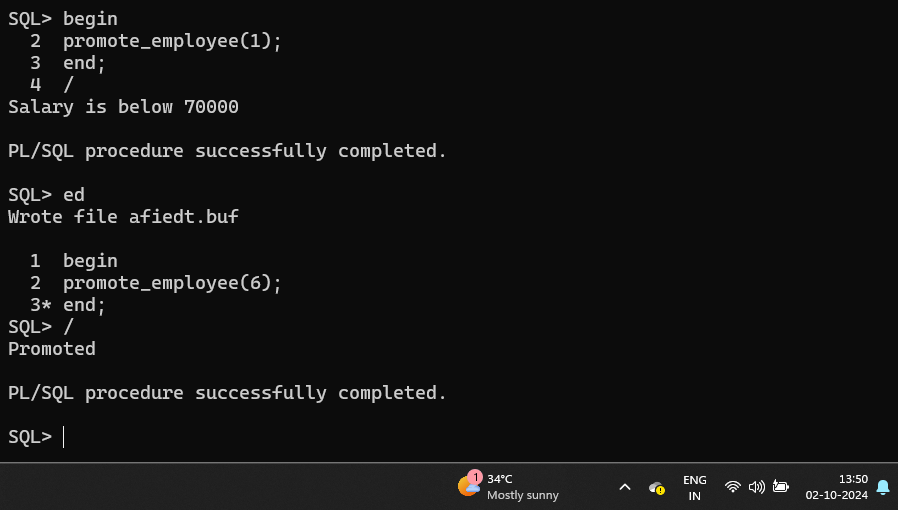
dbms\_output.put\_line('Promoted');

ELSE

dbms\_output.put\_line('Salary is below 70000');

END IF;

END;/



1. **Create a function get\_employee\_details that takes an emp\_id and returns the employee's name, salary, and hire date.**

CREATE OR REPLACE PROCEDURE get\_employee\_details

(e\_id IN employee.emp\_id%TYPE)

IS

emp\_name employee.emp\_name%TYPE;

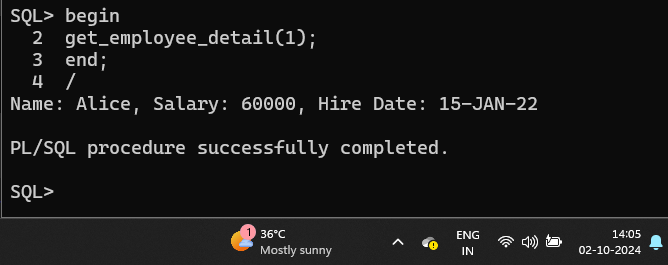
salary employee.salary%TYPE;

hire\_date employee.hire\_date%TYPE;

BEGIN

SELECT emp\_name, salary, hire\_date INTO emp\_name, salary, hire\_date FROM employee WHERE emp\_id = e\_id;

dbms\_output.put\_line('Name: ' || emp\_name || ', Salary: ' || salary || ', Hire Date: ' || hire\_date); END; /

****

1. **Write a function compare\_salaries that takes two emp\_ids and returns the name of the employee with the highest salary.**

CREATE OR REPLACE FUNCTION compare\_salaries

(e\_id1 employee.emp\_id%TYPE,

e\_id2 employee.emp\_id%TYPE)

RETURN employee.emp\_name%TYPE

IS

name1 employee.emp\_name%TYPE;

name2 employee.emp\_name%TYPE;

sal1 employee.salary%TYPE;

sal2 employee.salary%TYPE;

result\_name employee.emp\_name%TYPE;

BEGIN

SELECT emp\_name, salary

INTO name1, sal1

FROM employee

WHERE emp\_id = e\_id1;

SELECT emp\_name, salary

INTO name2, sal2

FROM employee

WHERE emp\_id = e\_id2;

IF sal1 > sal2 THEN

result\_name := name1;

dbms\_output.put\_line(name1 || ' has the higher salary.');

ELSE

result\_name := name2;

dbms\_output.put\_line(name2 || ' has the higher salary.');

END IF;

RETURN result\_name;

END;

**** /

1. **Create a function count\_employees\_in\_department that takes a department name and returns the number of employees in that department.**

CREATE OR REPLACE FUNCTION count\_employees\_in\_department

(dept employee.department%TYPE)

RETURN NUMBER

IS

cnt NUMBER;

BEGIN

SELECT COUNT(emp\_id)

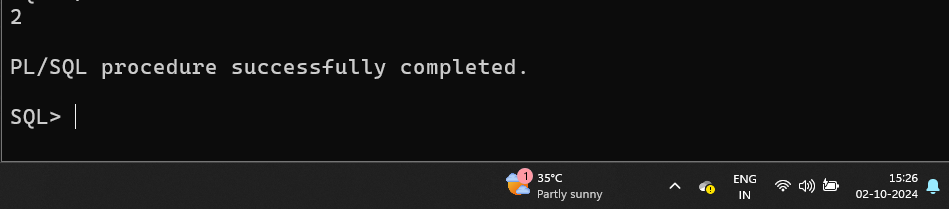
INTO cnt

FROM employee

WHERE department = dept;

RETURN cnt;

END;

****

1. **Write a procedure check\_hire\_date that takes an emp\_id and prints a message indicating if the employee was hired in on or after 2022.**

create or replace procedure check\_hire\_date

(e\_id employee.emp\_id%type)

is

h\_date date;

begin

select hire\_date into h\_date from employee where emp\_id = e\_id;

if h\_date >= to\_date('01-JAN-2022', 'DD-MON-YYYY') then

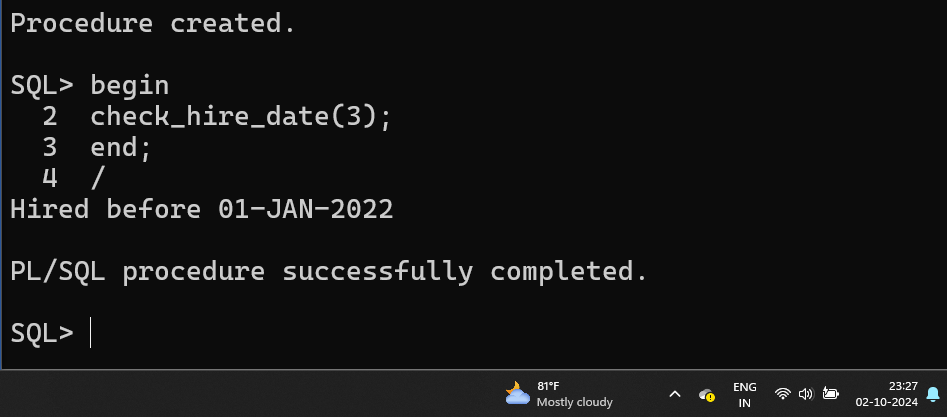
dbms\_output.put\_line('Hired after 01-JAN-2022');

else

dbms\_output.put\_line('Hired before 01-JAN-2022');

end if;

end;



1. **Create a procedure increase\_salary\_by\_department that takes a department name and a percentage increase and increases the salary of all employees in that department by the given percentage.**

CREATE OR REPLACE PROCEDURE increase\_salary\_by\_department (

dept IN employee.department%TYPE,

percentage IN NUMBER

)

AS

BEGIN

FOR i IN (SELECT employee\_id, salary FROM employee WHERE department = dept)

LOOP

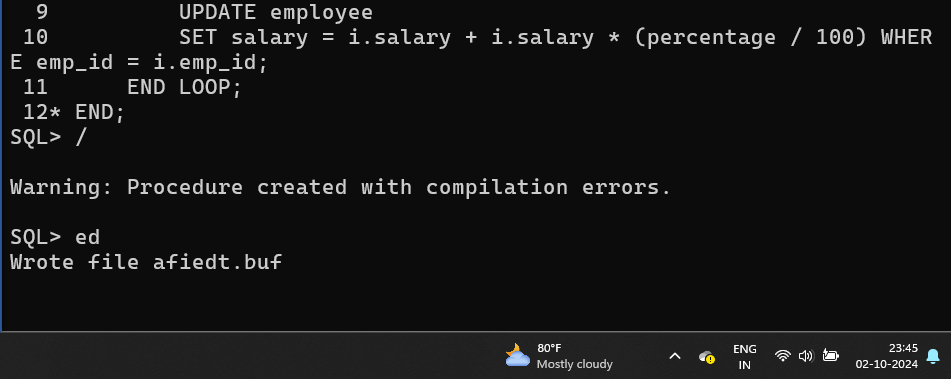
UPDATE employee

SET salary = i.salary + i.salary \* (percentage / 100) WHERE emp\_id = i.emp\_id;

END LOOP;

END;

/



1. **Write a function highest\_salary\_in\_department that takes a department name and returns the highest salary in that department. (using aggregate functions).**

CREATE OR REPLACE FUNCTION highest\_salary\_in\_department (

dept\_name IN employee.department%TYPE

)

RETURN NUMBER

IS

v\_highest\_salary NUMBER;

BEGIN

SELECT MAX(salary) INTO v\_highest\_salary FROM employee WHERE department = dept\_name;

RETURN v\_highest\_salary;

END;

/



1. **Write a function highest\_salary\_in\_department that takes a department name and returns the highest salary in that department. (Without using aggregate functions)**

CREATE OR REPLACE FUNCTION highest\_salary\_in\_department (

dept\_name IN employee.department%TYPE

)

RETURN NUMBER

IS

v\_highest\_salary employee.salary%TYPE := 0;

BEGIN

FOR emp\_rec IN (SELECT salary FROM employee WHERE department = dept\_name)

LOOP

IF emp\_rec.salary > v\_highest\_salary THEN

v\_highest\_salary := emp\_rec.salary;

END IF;

END LOOP;

RETURN v\_highest\_salary;

END;

/

