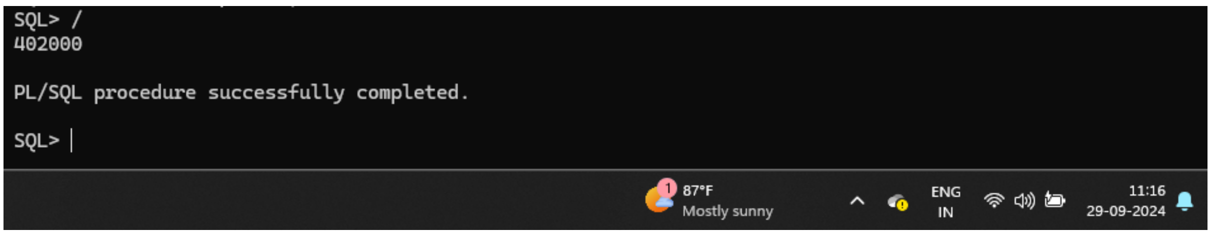
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

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;
```

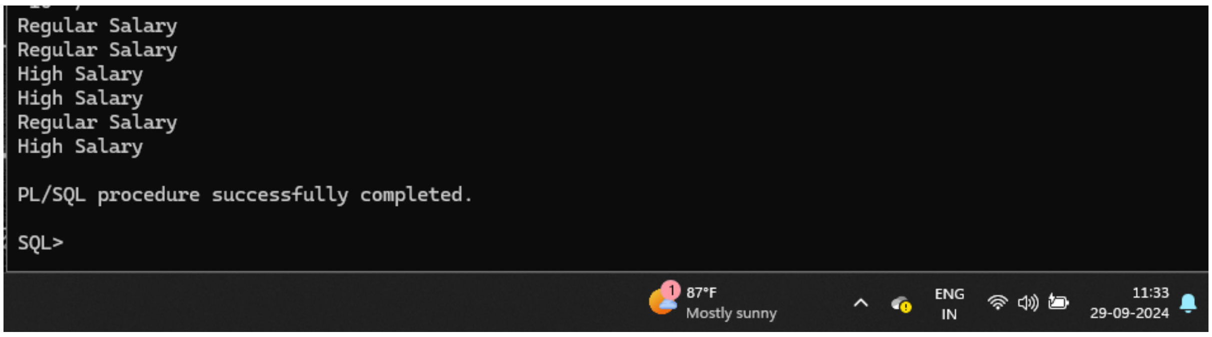


```
SQL> /
402000

PL/SQL procedure successfully completed.

SQL> |
```

87°F
Mostly sunny
ENG
IN
11:16
29-09-2024

2. **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;
```

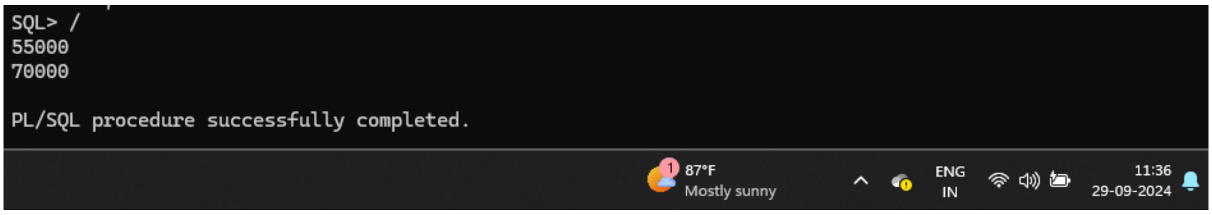


```
Regular Salary
Regular Salary
High Salary
High Salary
Regular Salary
High Salary

PL/SQL procedure successfully completed.

SQL>
```

87°F
Mostly sunny
ENG
IN
11:33
29-09-2024

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

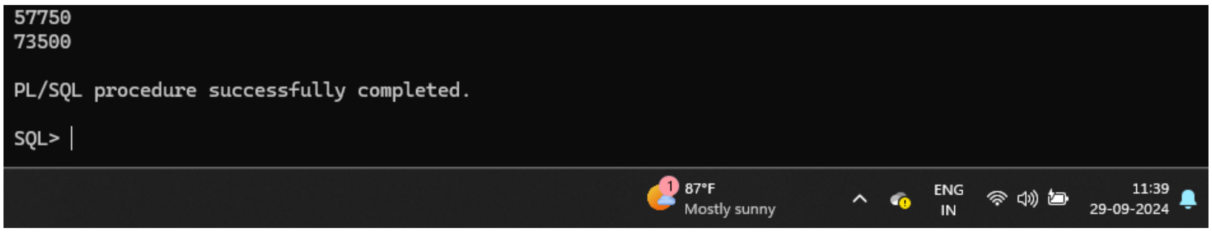
```
BEGIN
   FOR i IN (SELECT * FROM employee where department='IT') LOOP
          dbms_output.put_line(i.salary);
   END LOOP;
END;
```



```
SQL> /
55000
70000

PL/SQL procedure successfully completed.
```

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

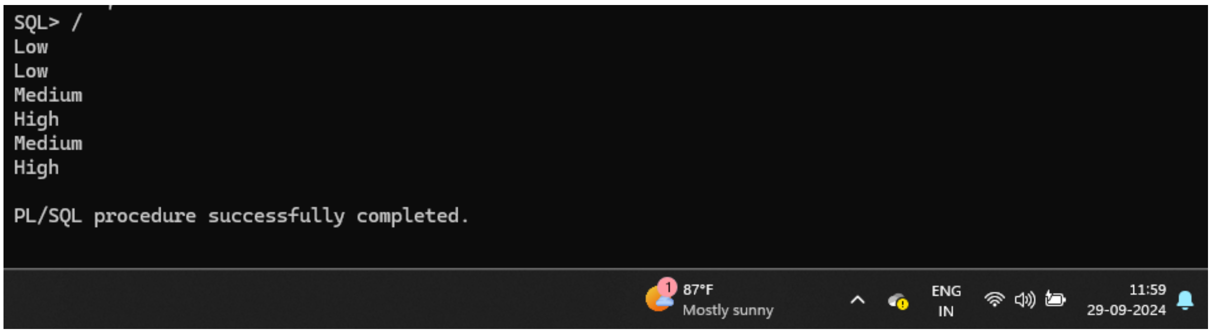
```
DECLARE
sal NUMBER :=0;
BEGIN
   FOR i IN (SELECT * FROM employee where department='IT') LOOP
          sal := i.salary + i.salary * 0.05;
          dbms_output.put_line(sal);
   END LOOP;
END;
```



```
57750
73500

PL/SQL procedure successfully completed.

SQL> |
```

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

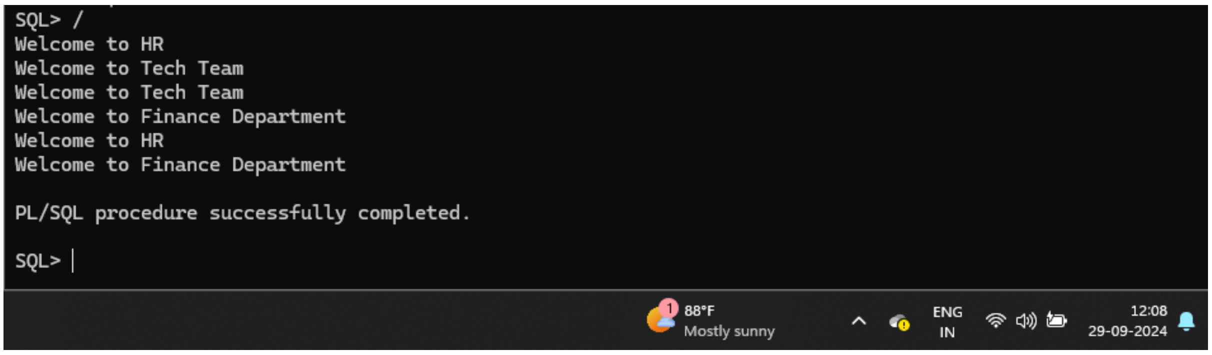
```
DECLARE
sal varchar(10);
BEGIN
  FOR i IN (SELECT * FROM employee) LOOP
       sal :=
       case
              when i.salary >= 72000 then 'High'
              when i.salary >= 65000 and i.salary <= 70000 then 'Medium'
              else 'Low'
       end;
       dbms_output.put_line(sal);
  END LOOP;
END;
```



```
SQL> /
Low
Low
Medium
High
Medium
High

PL/SQL procedure successfully completed.
```

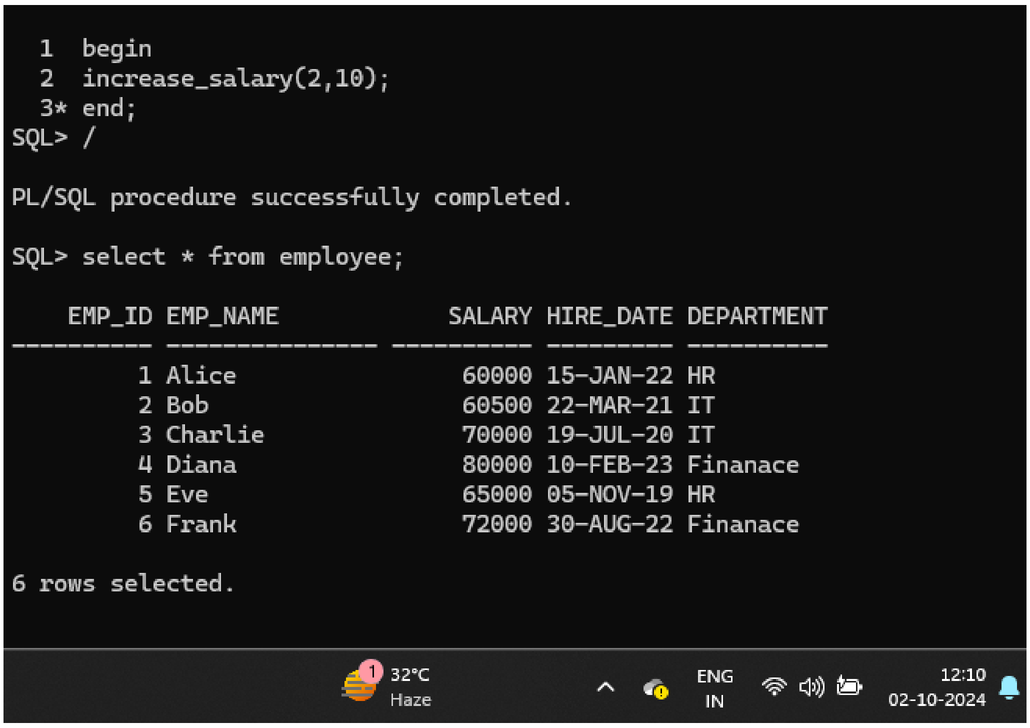
6. **Create a SQL query that uses a CASE expression to display different messages based on the department of employees. Display 'Welcome to HR' for employees in the 'HR' department, 'Tech Team' for employees in the 'IT' department, and 'Finance Team' for employees in the 'Finance' department.**

```
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;
```


```
SQL> /
Welcome to HR
Welcome to Tech Team
Welcome to Tech Team
Welcome to Finance Department
Welcome to HR
Welcome to Finance Department

PL/SQL procedure successfully completed.

SQL> |
```

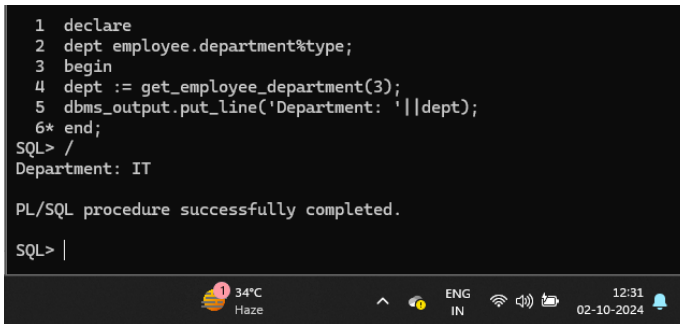
7. **Create a PL/SQL procedure increase_salary that takes an emp_id and a percentage increase and updates the salary of the employee by the given percentage.**

```
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  begin
  2  increase_salary(2,10);
  3* end;
SQL> /

PL/SQL procedure successfully completed.

SQL> select * from employee;

    EMP_ID EMP_NAME               SALARY HIRE_DATE DEPARTMENT
---------- --------------- ---------- --------- ----------
         1 Alice                 60000 15-JAN-22 HR
         2 Bob                   60500 22-MAR-21 IT
         3 Charlie               70000 19-JUL-20 IT
         4 Diana                 80000 10-FEB-23 Finanace
         5 Eve                   65000 05-NOV-19 HR
         6 Frank                 72000 30-AUG-22 Finanace

6 rows selected.
```

8. **Write a PL/SQL function get_employee_department that takes an emp_id and returns the department of the employee.**

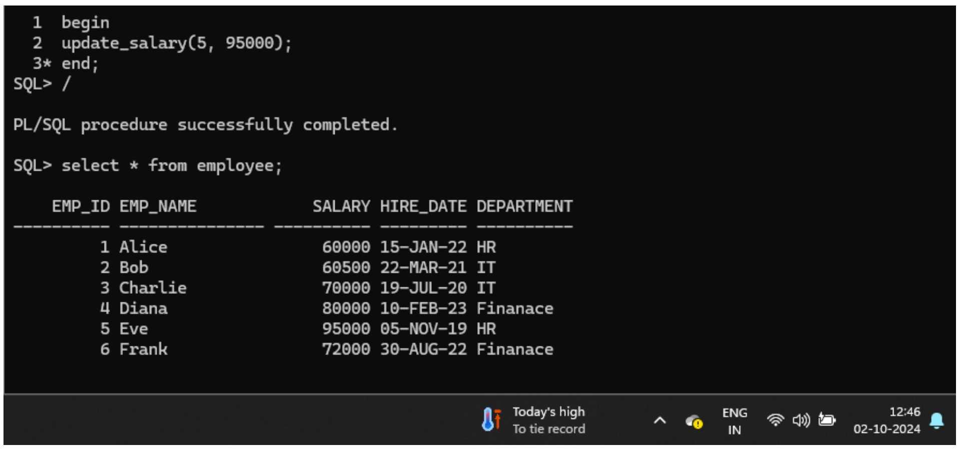
```
CREATE OR REPLACE FUNCTION get_employee_department
  (e_id NUMBER)
  RETURN employee.department%TYPE
IS
  emp_dept employee.department%TYPE;
BEGIN
  SELECT department INTO emp_dept FROM employee WHERE emp_id = e_id;
  RETURN emp_dept;
END;/
/
```



```
  1  declare
  2  dept employee.department%type;
  3  begin
  4  dept := get_employee_department(3);
  5  dbms_output.put_line('Department: '||dept);
  6* end;
SQL> /
Department: IT

PL/SQL procedure successfully completed.

SQL> |
```

9. **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.**

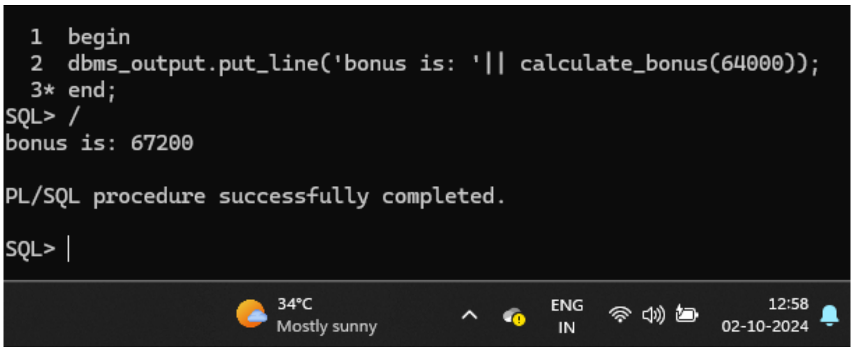
```
create or replace procedure update_salary
(e_id in employee.emp_id%type,
new_sal in employee.salary%type)
is
begin
update employee set salary= new_sal where emp_id = e_id;
end update_salary;
/
```



```
  1  begin
  2  update_salary(5, 95000);
  3* end;
SQL> /

PL/SQL procedure successfully completed.

SQL> select * from employee;

    EMP_ID EMP_NAME            SALARY HIRE_DATE DEPARTMENT
---------- --------------- ---------- --------- ----------
         1 Alice                60000 15-JAN-22 HR
         2 Bob                  60500 22-MAR-21 IT
         3 Charlie              70000 19-JUL-20 IT
         4 Diana                80000 10-FEB-23 Finanace
         5 Eve                  95000 05-NOV-19 HR
         6 Frank                72000 30-AUG-22 Finanace
```

10. 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.

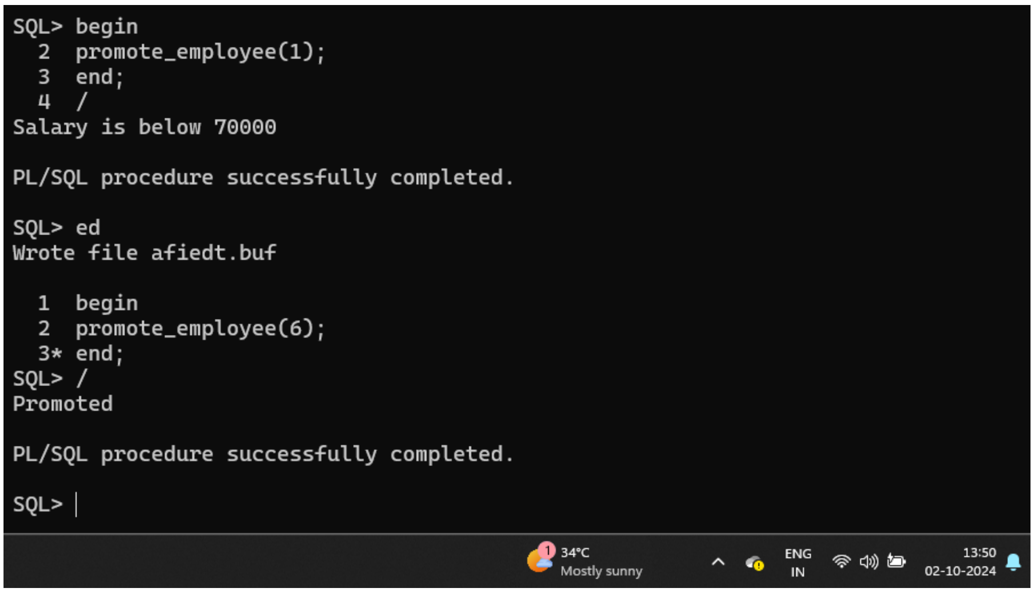
```
CREATE OR REPLACE FUNCTION calculate_bonus
   (salary employee.salary%type)
   RETURN employee.salary%type
  IS
   bonus employee.salary%type;
  BEGIN
   if salary > 70000 then
    return salary+salary*0.1;
   elsif salary >=60000 and salary <=70000 then
   return salary+salary*0.05;
  else
   return salary+salary*0.01;
  end if;
```



```
  1  begin
  2  dbms_output.put_line('bonus is: '|| calculate_bonus(64000));
  3* end;
SQL> /
bonus is: 67200

PL/SQL procedure successfully completed.

SQL> |
```

11. 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.

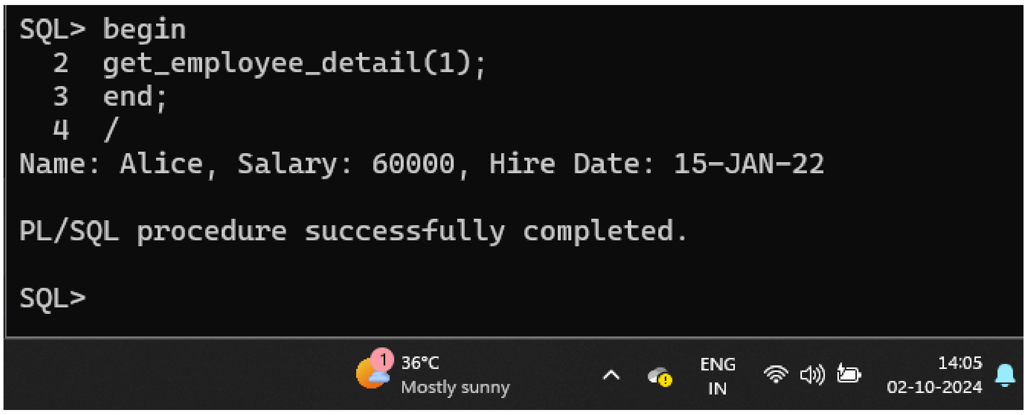
```
CREATE OR REPLACE PROCEDURE promote_employee
 (e_id IN employee.emp_id%TYPE)
IS
 current_sal employee.salary%TYPE;
BEGIN
 SELECT salary INTO current_sal
 FROM employee
 WHERE emp_id = e_id;
 IF current_sal >= 70000 THEN
        UPDATE employee SET department = 'Management' WHERE emp_id = e_id;
        dbms_output.put_line('Promoted');
 ELSE
  dbms_output.put_line('Salary is below 70000');
 END IF;
END;/
```



```
SQL> begin
  2  promote_employee(1);
  3  end;
  4  /
Salary is below 70000

PL/SQL procedure successfully completed.

SQL> ed
Wrote file afiedt.buf

  1  begin
  2  promote_employee(6);
  3* end;
SQL> /
Promoted

PL/SQL procedure successfully completed.

SQL> |
```

12. Create a function get_employee_details that takes an emp_id and returns the employee's name, salary, and hire date.

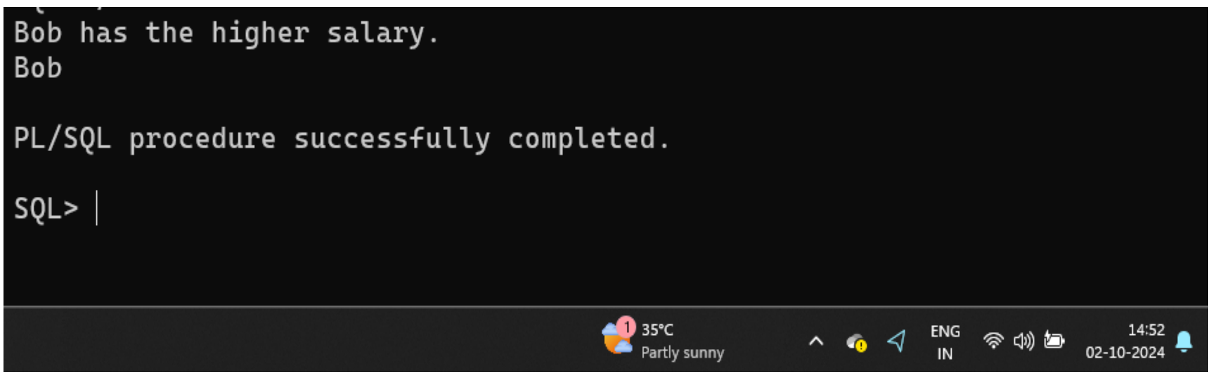
```
CREATE OR REPLACE PROCEDURE get_employee_details
(e_id IN employee.emp_id%TYPE)
IS
emp_name employee.emp_name%TYPE;
salary employee.salary%TYPE;
hire_date employee.hire_date%TYPE;
BEGIN
SELECT emp_name, salary, hire_date INTO emp_name, salary, hire_date FROM
employee WHERE emp_id = e_id;
dbms_output.put_line('Name: ' || emp_name || ', Salary: ' || salary || ', Hire Date: '
|| hire_date); END; /
```



```
SQL> begin
  2  get_employee_detail(1);
  3  end;
  4  /
Name: Alice, Salary: 60000, Hire Date: 15-JAN-22

PL/SQL procedure successfully completed.

SQL>
```

13. Write a function compare_salaries that takes two emp_ids and returns the name of the employee with the highest salary.

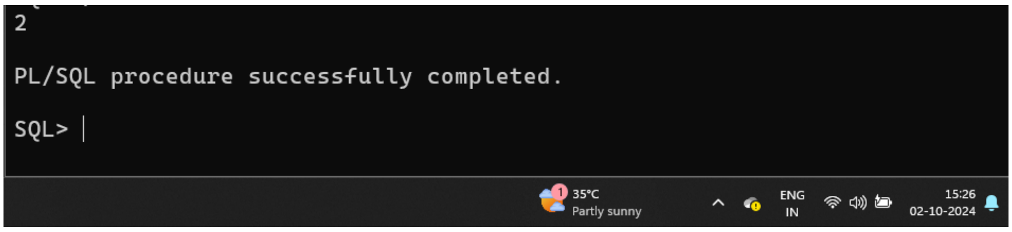
```
CREATE OR REPLACE FUNCTION compare_salaries
   (e_id1 employee.emp_id%TYPE,
    e_id2 employee.emp_id%TYPE)
   RETURN employee.emp_name%TYPE
  IS
   name1 employee.emp_name%TYPE;
   name2 employee.emp_name%TYPE;
   sal1 employee.salary%TYPE;
   sal2 employee.salary%TYPE;
   result_name employee.emp_name%TYPE;
  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;
      /
```



```
Bob has the higher salary.
Bob

PL/SQL procedure successfully completed.

SQL>
```

14. Create a function count_employees_in_department that takes a department name and returns the number of employees in that department.

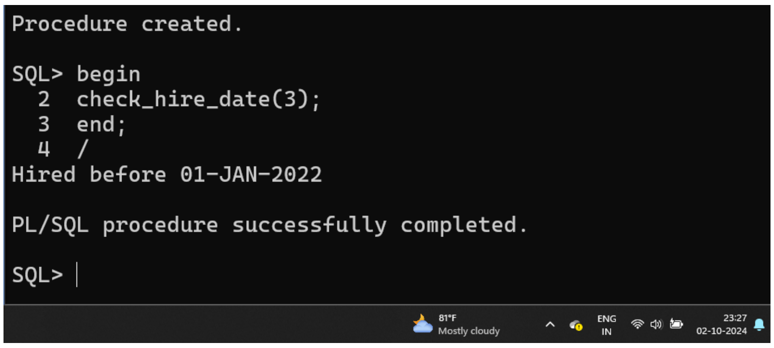
```
    CREATE OR REPLACE FUNCTION count_employees_in_department
      (dept employee.department%TYPE)
      RETURN NUMBER
     IS
      cnt NUMBER;
     BEGIN
      SELECT COUNT(emp_id)
     INTO cnt
      FROM employee
     WHERE department = dept;
     RETURN cnt;
   END;
```



```
2

PL/SQL procedure successfully completed.

SQL> |
```

15. 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.

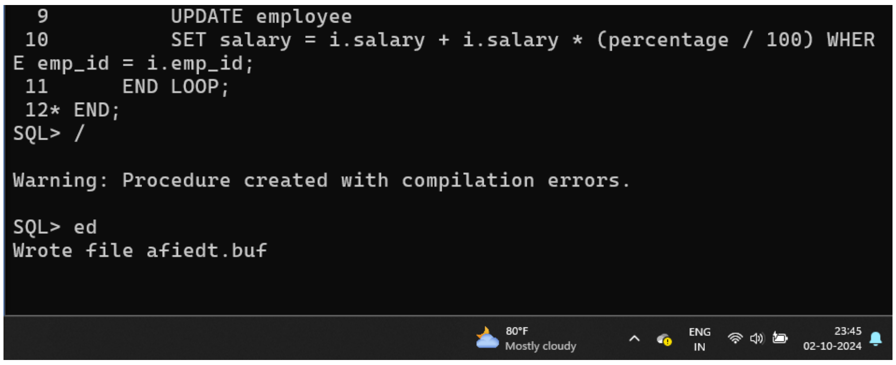
```
create or replace procedure check_hire_date
(e_id employee.emp_id%type)
is
h_date date;
begin
        select hire_date into h_date from employee where emp_id = e_id;
        if h_date >= to_date('01-JAN-2022', 'DD-MON-YYYY') then
            dbms_output.put_line('Hired after 01-JAN-2022');
        else
            dbms_output.put_line('Hired before 01-JAN-2022');
end if;
end;
```



```
Procedure created.

SQL> begin
  2  check_hire_date(3);
  3  end;
  4  /
Hired before 01-JAN-2022

PL/SQL procedure successfully completed.

SQL> |
```

16. 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.

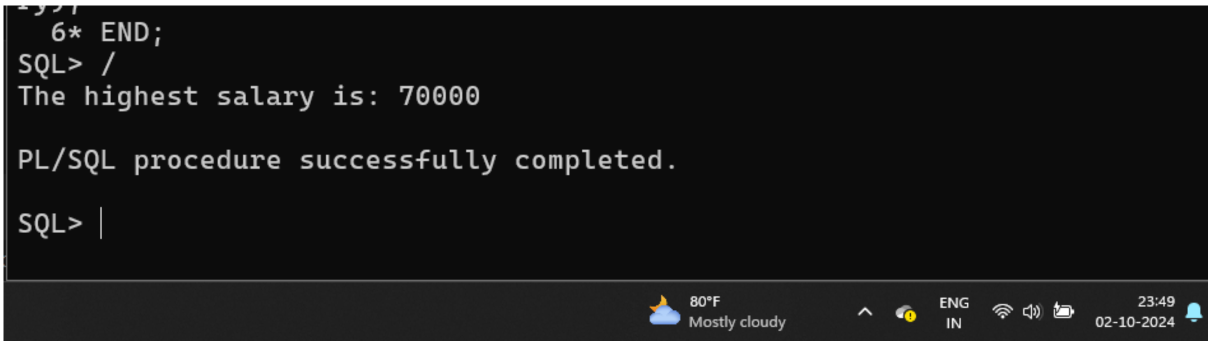
```
CREATE OR REPLACE PROCEDURE increase_salary_by_department (
   dept IN employee.department%TYPE,
   percentage IN NUMBER
)
AS
BEGIN
   FOR i IN (SELECT employee_id, salary FROM employee WHERE department = dept)
   LOOP
      UPDATE employee
      SET salary = i.salary + i.salary * (percentage / 100) WHERE emp_id = i.emp_id;
   END LOOP;
END;
/
```



```
            UPDATE employee
            SET salary = i.salary + i.salary * (percentage / 100) WHER
E emp_id = i.emp_id;
        END LOOP;
    END;
SQL> /

Warning: Procedure created with compilation errors.

SQL> ed
Wrote file afiedt.buf
```

17. Write a function highest_salary_in_department that takes a department name and returns the highest salary in that department. (using aggregate functions).

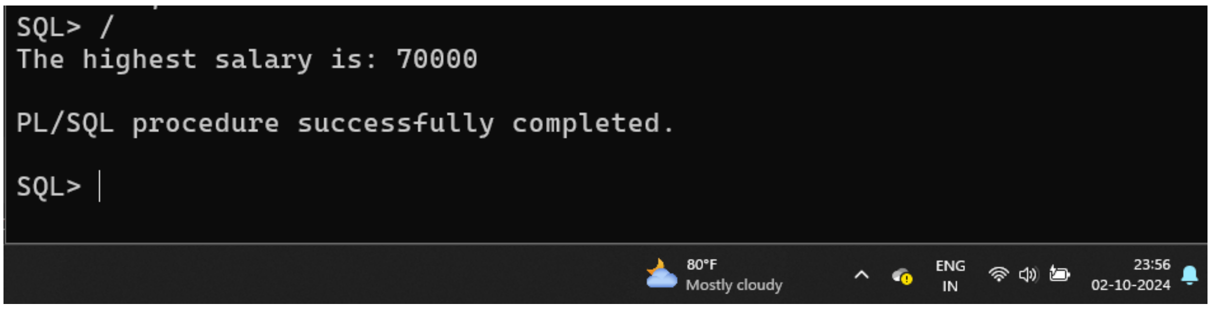
```
CREATE OR REPLACE FUNCTION highest_salary_in_department (
   dept_name IN employee.department%TYPE
)
RETURN NUMBER
IS
   v_highest_salary NUMBER;
BEGIN
   SELECT MAX(salary) INTO v_highest_salary FROM employee WHERE department =
dept_name;
   RETURN v_highest_salary;
END;
/
```



```
  6* END;
SQL> /
The highest salary is: 70000

PL/SQL procedure successfully completed.

SQL>
```

18. 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;
/
```



```
SQL> /
The highest salary is: 70000

PL/SQL procedure successfully completed.

SQL>
```