```
-- ====================================Task 1: Compute and Print the Bonus Amount
DECLARE
    emp_id NUMBER := &emp_id;
    salary NUMBER;
    bonus NUMBER;
BEGIN
    SELECT salary INTO salary FROM hr.employees WHERE employee_id = emp_id;

    IF salary IS NULL THEN
        bonus := 0;
    ELSIF salary < 1000 THEN
        bonus := salary * 0.10;
    ELSIF salary BETWEEN 1000 AND 1500 THEN
        bonus := salary * 0.15;
    ELSE
        bonus := salary * 0.20;
    END IF;

    DBMS_OUTPUT.PUT_LINE('Bonus amount for employee ' || emp_id || ' is: ' || bonus);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No employee found with ID: ' || emp_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
```

Script Output ×

Task completed in 2.677 seconds

```
END;

PL/SQL procedure successfully completed.
```

```
--=======================================Task 2: Update Salary if Commission is Nu
DECLARE
    emp_id NUMBER := &emp_id;
    commission NUMBER;
BEGIN
    SELECT commission_pct INTO commission
    FROM hr.employees
    WHERE employee_id = emp_id;

    IF commission IS NULL THEN
        UPDATE hr.employees
        SET salary = salary + NVL(commission, 0)
        WHERE employee_id = emp_id;
        DBMS_OUTPUT.PUT_LINE('Salary updated for employee ' || emp_id);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Commission is not null; no update performed.');
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No employee found with employee ID ' || emp_id);
END;
```

Script Output ×

Task completed in 2.807 seconds

```
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No employee found with employee ID ' || emp_id);
END;

PL/SQL procedure successfully completed.
```

```
--============================Task 3: Get Department Name for Employee in Dept 30
DECLARE
    dept_name VARCHAR2(100);
BEGIN
    SELECT department_name INTO dept_name
    FROM hr.departments
    WHERE department_id = 30;

    DBMS_OUTPUT.PUT_LINE('Department Name for Dept 30: ' || dept_name);
END;
```

Script Output

Task completed in 0.032 seconds

```
PL/SQL procedure successfully completed.
```

```
--============================= Task 4: Find Job Nature in Dept 20 (Argument)
DECLARE
    job_type VARCHAR2(100);
    dept_id NUMBER := 20;
BEGIN
    SELECT job_id INTO job_type
    FROM hr.employees
    WHERE department_id = dept_id
    AND ROWNUM = 1;

    DBMS_OUTPUT.PUT_LINE('Job Nature in Dept 20: ' || job_type);
END;
```

Script Output

Task completed in 0.035 seconds

```
PL/SQL procedure successfully completed.
```

```
--========================== Task 5: Find Salary in Dept 20 (Argument)
DECLARE
    emp_salary NUMBER;
    dept_id NUMBER := 20;
BEGIN
    SELECT salary INTO emp_salary
    FROM hr.employees
    WHERE department_id = dept_id
    AND ROWNUM = 1;

    DBMS_OUTPUT.PUT_LINE('Employee Salary in Dept 20: ' || emp_salary);
END;
```

Script Output x

Task completed in 0.033 seconds

PL/SQL procedure successfully completed.

```
--========================== Task 6: Update Salary with 10% Increase
GRANT UPDATE ON hr.employees TO homeuser;
CREATE OR REPLACE PROCEDURE update_salary(emp_id IN NUMBER) IS
BEGIN
    UPDATE hr.employees
    SET salary = salary * 1.10
    WHERE employee_id = emp_id;

    DBMS_OUTPUT.PUT_LINE('Salary updated with 10% increase for employee ' || emp_id);
END;
```

Script Output x

Task completed in 0.07 seconds

```
Grant succeeded.


Procedure UPDATE_SALARY compiled

LINE/COL  ERROR
--------- -------------------------------------------------------------
3/5       PL/SQL: SQL Statement ignored
3/15      PL/SQL: ORA-01031: insufficient privileges
Errors: check compiler log
```

```
--========================== Task 7: Add Rs.1000 to Salaries Above 5000 in Specific Dept
GRANT UPDATE ON hr.employees TO homeuser;
CREATE OR REPLACE PROCEDURE add_bonus_to_dept(dept_id IN NUMBER) IS
BEGIN
    UPDATE hr.employees
    SET salary = salary + 1000
    WHERE salary > 5000 AND department_id = dept_id;

    DBMS_OUTPUT.PUT_LINE('Added Rs.1000 to salaries in Dept ' || dept_id || ' where salary > 5000');
END;
```

Script Output

Task completed in 0.063 seconds

```
Grant succeeded.


Procedure ADD_BONUS_TO_DEPT compiled

LINE/COL  ERROR
--------- -----------------------------------------------------------------
3/5       PL/SQL: SQL Statement ignored
3/15      PL/SQL: ORA-01031: insufficient privileges
Errors: check compiler log
```

```
--==========================Task 8: Create Views
--==================== a. Display Each Designation and Number of Employees
CREATE VIEW hr.emp_designation_count AS
SELECT job_id, COUNT(*) AS num_employees
FROM hr.employees
GROUP BY job_id;
-- ==================b. Display Employee Details (Excluding 'King')
CREATE VIEW hr.emp_details_excluding_king AS
SELECT e.employee_id, e.first_name, e.last_name, e.department_id, d.department_name
FROM hr.employees e
JOIN hr.departments d ON e.department_id = d.department_id
WHERE e.last_name != 'King';
--==================== c. Display Employee Details
CREATE VIEW hr.emp_details AS
SELECT e.employee_id, e.first_name, e.last_name, e.department_id, d.department_name
FROM hr.employees e
JOIN hr.departments d ON e.department_id = d.department_id;
```

Script Output

Task completed in 0.038 seconds

```
View HR.EMP_DESIGNATION_COUNT created.
```

```
-- ==================b. Display Employee Details (Excluding 'King')
CREATE VIEW hr.emp_details_excluding_king AS
SELECT e.employee_id, e.first_name, e.last_name, e.department_id, d.department_name
FROM hr.employees e
JOIN hr.departments d ON e.department_id = d.department_id
WHERE e.last_name != 'King';
```

Script Output

Task completed in 0.048 seconds

```
*Action:

View HR.EMP DETAILS EXCLUDING KING created.
```

```
--==================== c. Display Employee Details
CREATE VIEW hr.emp_details AS
SELECT e.employee_id, e.first_name, e.last_name, e.department_id, d.department_name
FROM hr.employees e
JOIN hr.departments d ON e.department_id = d.department_id;
```

Script Output

Task completed in 0.038 seconds

```
View HR.EMP_DETAILS created.
```

```
--===================Task 9: Add Two Inputs and Show Output=======================
DECLARE
    num1 NUMBER := &num1;
    num2 NUMBER := &num2;
    sum_result NUMBER;
BEGIN
    sum_result := num1 + num2;
    DBMS_OUTPUT.PUT_LINE('The sum is: ' || sum_result);
END;
```

Script Output

Task completed in 3.881 seconds

```
END;

PL/SQL procedure successfully completed.
```

```
--================Task 10: Sum of Numbers Between Two Boundaries
DECLARE
    lower_bound NUMBER := &lower_bound;
    upper_bound NUMBER := &upper_bound;
    sum_result NUMBER := 0;
BEGIN
    FOR i IN lower_bound .. upper_bound LOOP
        sum_result := sum_result + i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('The sum of numbers between ' || lower_bound
END;
/
```

Script Output

Task completed in 3.948 seconds

```
END;

PL/SQL procedure successfully completed.
```

```
--================ Task 11: Retrieve Employee Name, Hiredate, and Department Name
DECLARE
    emp_id NUMBER := &emp_id;
    emp_name VARCHAR2(100);
    hire_date DATE;
    dept_name VARCHAR2(100);
BEGIN
    BEGIN
        SELECT (e.first_name || ' ' || e.last_name), e.hire_date,
               (SELECT department_name FROM hr.departments WHERE department_id = e.department_id)
        INTO emp_name, hire_date, dept_name
        FROM hr.employees e
        WHERE e.employee_id = emp_id;

        DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_name);
        DBMS_OUTPUT.PUT_LINE('Hire Date: ' || hire_date);
        DBMS_OUTPUT.PUT_LINE('Department Name: ' || dept_name);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('No employee found with ID ' || emp_id);
    END;
END;
/
```

Script Output

Task completed in 2.315 seconds

```
END;

PL/SQL procedure successfully completed.
```

```
--================Task 12: Check if Number is Palindrome
DECLARE
    num NUMBER := &num;
    reverse_num NUMBER := 0;
    temp NUMBER;
BEGIN
    temp := num;
    WHILE temp > 0 LOOP
        reverse_num := (reverse_num * 10) + MOD(temp, 10);
        temp := FLOOR(temp / 10);
    END LOOP;

    IF num = reverse_num THEN
        DBMS_OUTPUT.PUT_LINE(num || ' is a palindrome.');
    ELSE
        DBMS_OUTPUT.PUT_LINE(num || ' is not a palindrome.');
    END IF;
END;
```

Script Output

Task completed in 2.731 seconds

```
END;

PL/SQL procedure successfully completed.
```

```
--====================Task 13: Insert into Employee and Department Tables
DECLARE
    emp_id NUMBER := &emp_id;
    emp_name VARCHAR2(100) := '&emp_name';
    last_name VARCHAR2(100) := '&last_name';
    dept_id NUMBER := &dept_id;
    dept_name VARCHAR2(100) := '&dept_name';
    salary NUMBER := &salary;
    job_id VARCHAR2(10) := '&job_id';
BEGIN
    INSERT INTO hr.departments (department_id, department_name)
    VALUES (dept_id, dept_name);

    INSERT INTO hr.employees (employee_id, first_name, last_name, salary, department_id, job_id)
    VALUES (emp_id, emp_name, last_name, salary, dept_id, job_id);

    DBMS_OUTPUT.PUT_LINE('Inserted employee and department data successfully.');
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Error: Duplicate value for a unique constraint.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
```

Script Output

Task completed in 7.915 seconds

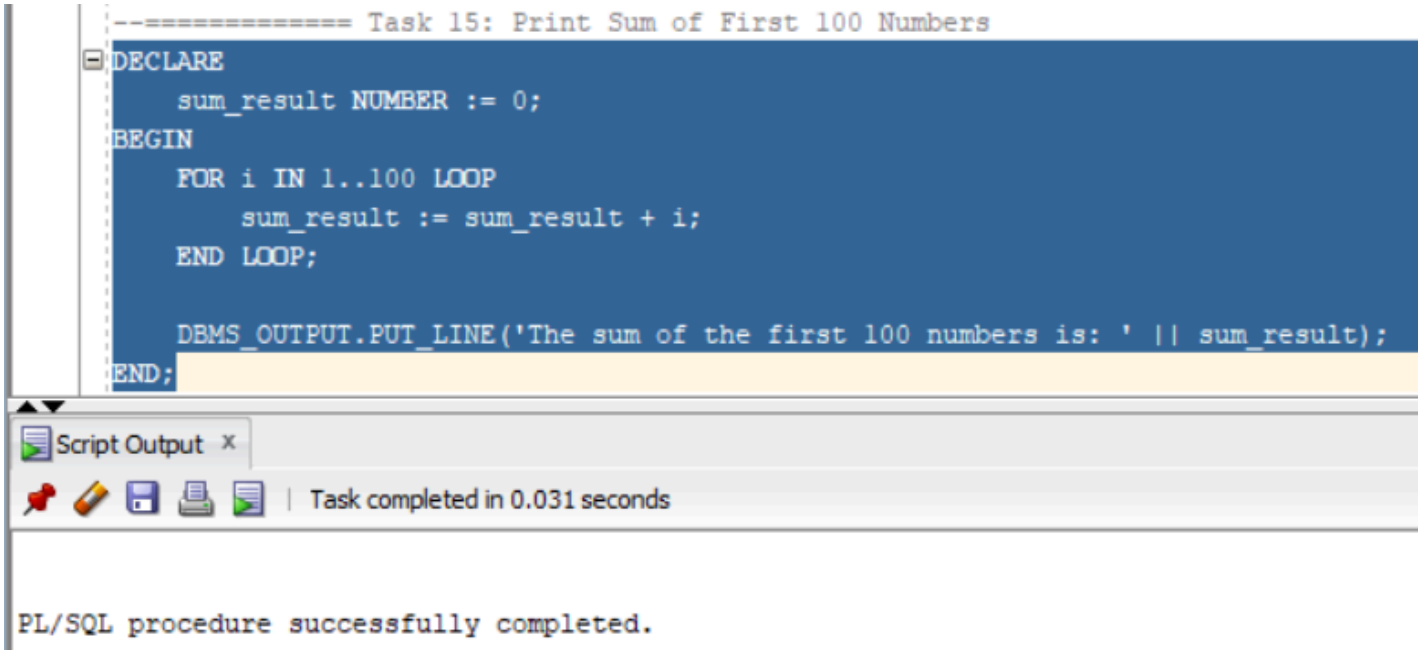
```
END;

PL/SQL procedure successfully completed.
```

```
--==========Task 14: Find First Employee with Salary Over $2500 Higher than Employee 90
DECLARE
    emp_id NUMBER := 90;
    found BOOLEAN := FALSE;
    emp_salary NUMBER;
BEGIN
    FOR emp IN (SELECT employee_id, salary FROM hr.employees WHERE salary > 2500 ORDER BY manager_id NULLS FIRST) LOOP
        IF NOT found THEN
            DBMS_OUTPUT.PUT_LINE('First employee with salary over $2500 in chain: ' || emp.employee_id);
            found := TRUE;
        END IF;
    END LOOP;
END;
```

Script Output

Task completed in 0.031 seconds

```
PL/SQL procedure successfully completed.
```


-------------- Task 15: Print Sum of First 100 Numbers
DECLARE
sum_result NUMBER := 0;
BEGIN
FOR i IN 1..100 LOOP
sum_result := sum_result + i;
END LOOP;
DBMS_OUTPUT.PUT_LINE('The sum of the first 100 numbers is: ' || sum_result);
END;
Script Output
Task completed in 0.031 seconds
PL/SQL procedure successfully completed.