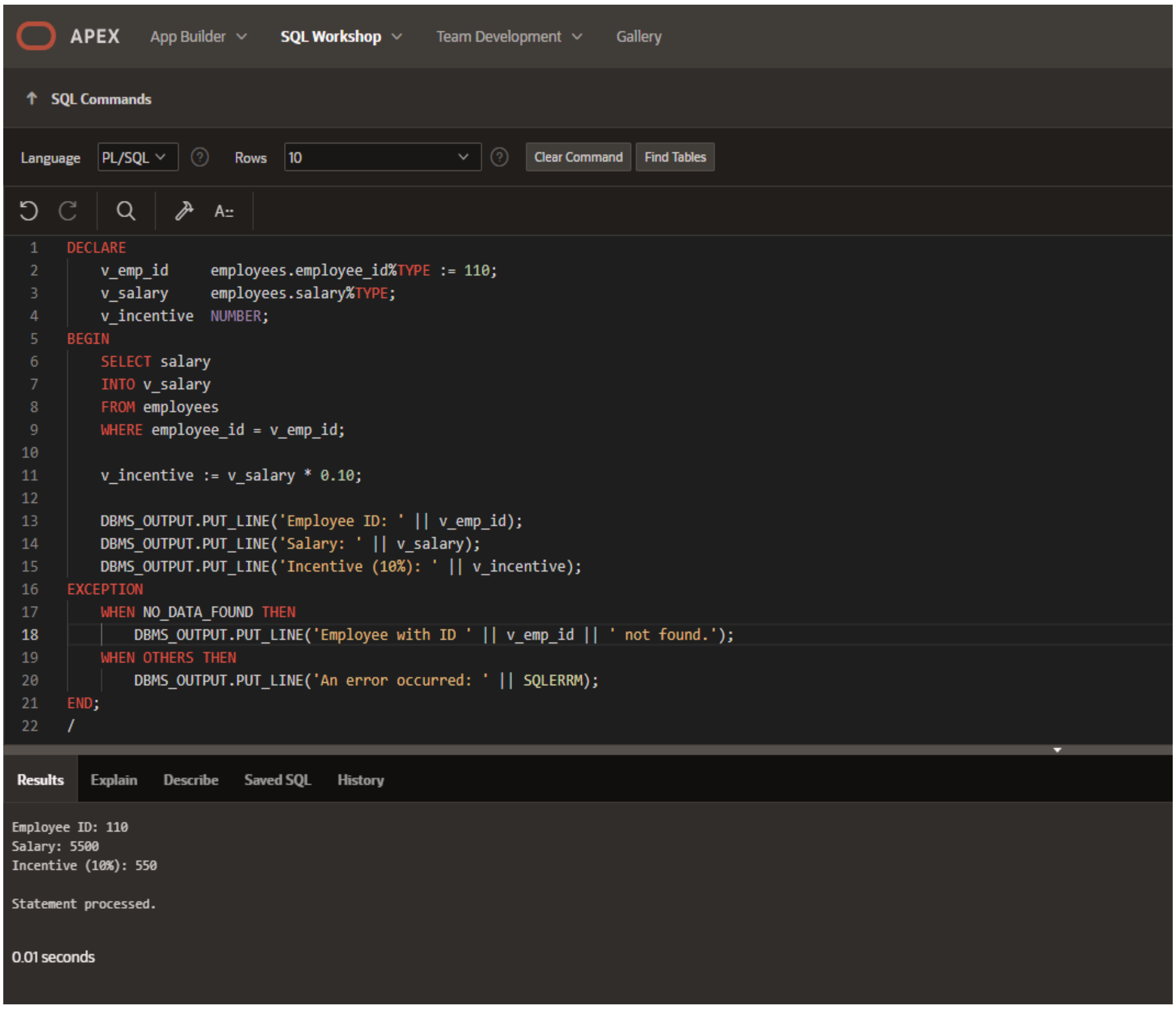
PL/SQL
CONTROL STRUCTURES

Name: Vedhasree S
Register Number: 240701580
Department: CSE

PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.



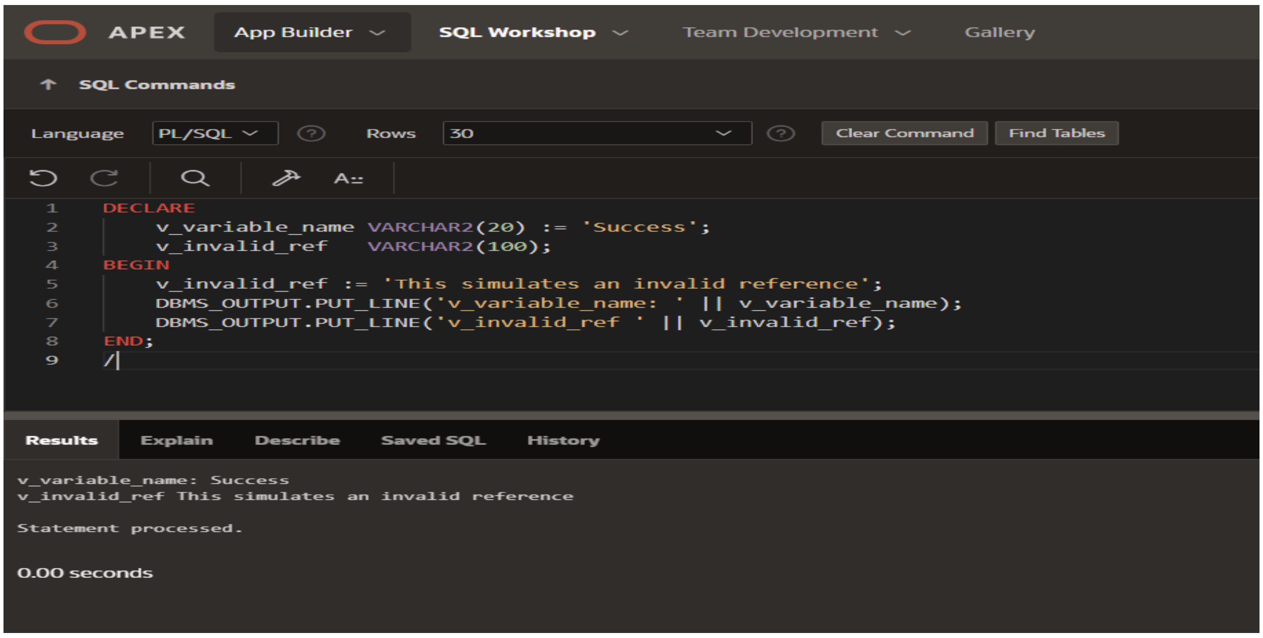
Language PL/SQL Rows 10

```
DECLARE
    v_emp_id     employees.employee_id%TYPE := 110;
    v_salary     employees.salary%TYPE;
    v_incentive  NUMBER;
BEGIN
    SELECT salary
    INTO v_salary
    FROM employees
    WHERE employee_id = v_emp_id;

    v_incentive := v_salary * 0.10;

    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_emp_id);
    DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);
    DBMS_OUTPUT.PUT_LINE('Incentive (10%): ' || v_incentive);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Employee with ID ' || v_emp_id || ' not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

```
Employee ID: 110
Salary: 5500
Incentive (10%): 550

Statement processed.

0.01 seconds
```

PROGRAM 2

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.



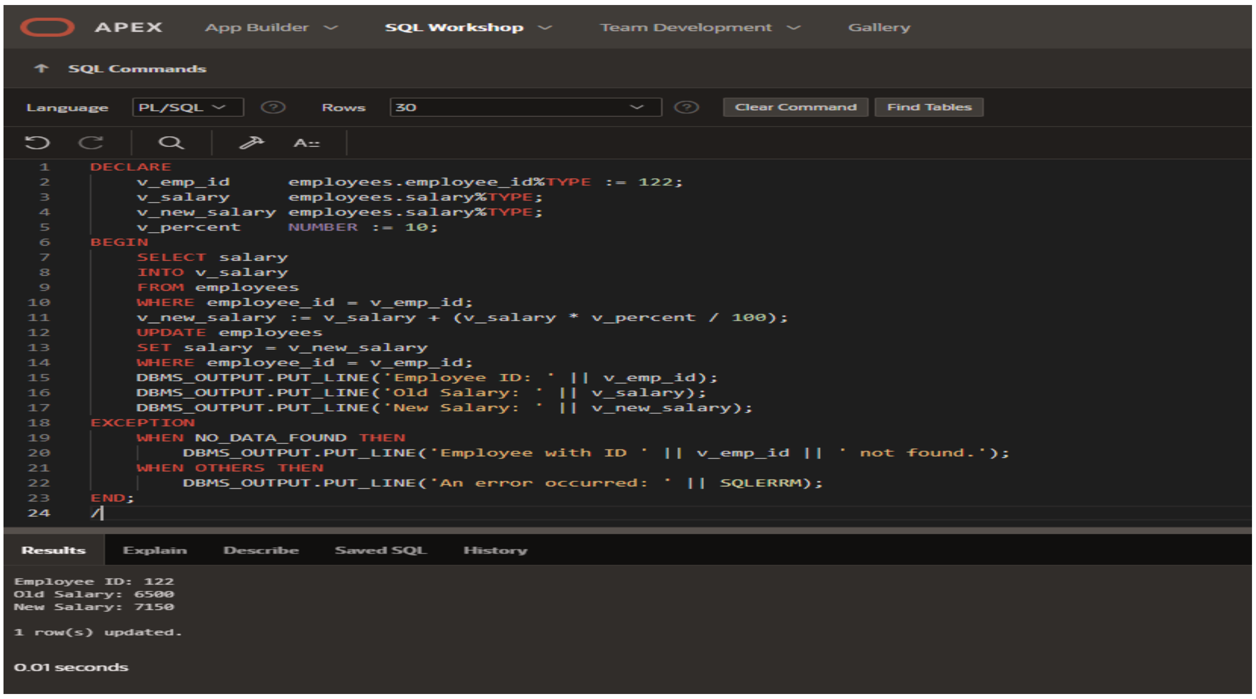
```
DECLARE
    v_variable_name VARCHAR2(20) := 'Success';
    v_invalid_ref   VARCHAR2(100);
BEGIN
    v_invalid_ref := 'This simulates an invalid reference';
    DBMS_OUTPUT.PUT_LINE('v_variable_name: ' || v_variable_name);
    DBMS_OUTPUT.PUT_LINE('v_invalid_ref ' || v_invalid_ref);
END;
/
```

```
v_variable_name: Success
v_invalid_ref This simulates an invalid reference

Statement processed.

0.00 seconds
```

PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122.

Sample table: employees

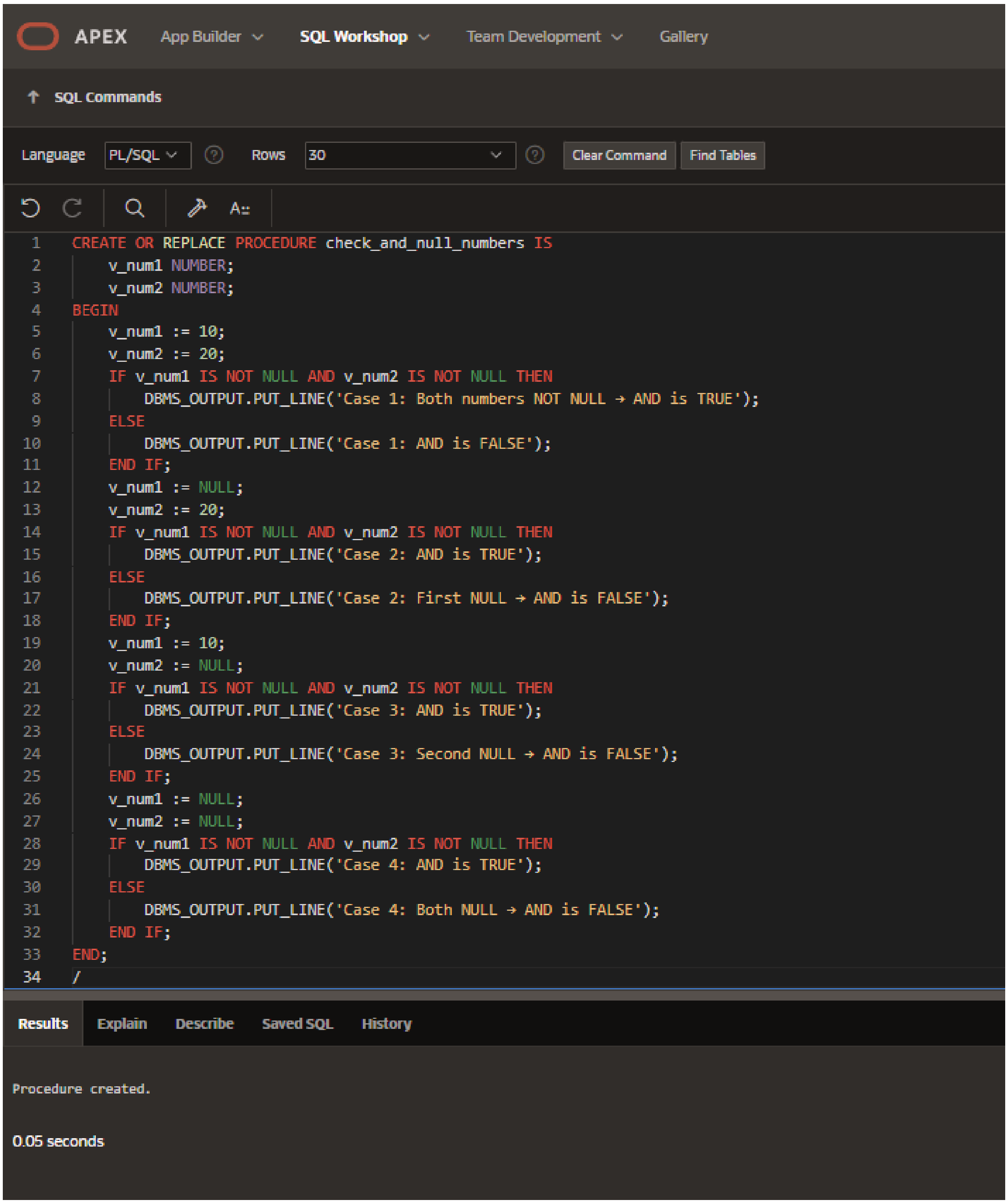


```
DECLARE
    v_emp_id     employees.employee_id%TYPE := 122;
    v_salary     employees.salary%TYPE;
    v_new_salary employees.salary%TYPE;
    v_percent    NUMBER := 10;
BEGIN
    SELECT salary
    INTO v_salary
    FROM employees
    WHERE employee_id = v_emp_id;
    v_new_salary := v_salary + (v_salary * v_percent / 100);
    UPDATE employees
    SET salary = v_new_salary
    WHERE employee_id = v_emp_id;
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_emp_id);
    DBMS_OUTPUT.PUT_LINE('Old Salary: ' || v_salary);
    DBMS_OUTPUT.PUT_LINE('New Salary: ' || v_new_salary);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Employee with ID ' || v_emp_id || ' not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

```
Employee ID: 122
Old Salary: 6500
New Salary: 7150

1 row(s) updated.

0.01 seconds
```

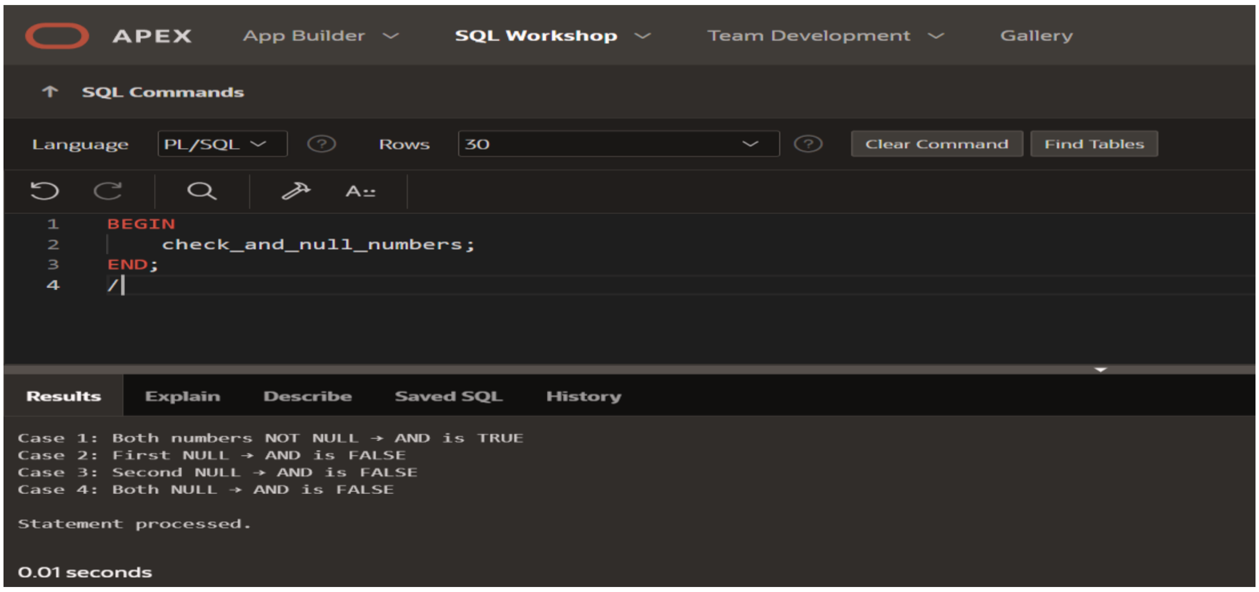
PROGRAM 4

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.



```sql
CREATE OR REPLACE PROCEDURE check_and_null_numbers IS
    v_num1 NUMBER;
    v_num2 NUMBER;
BEGIN
    v_num1 := 10;
    v_num2 := 20;
    IF v_num1 IS NOT NULL AND v_num2 IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Case 1: Both numbers NOT NULL → AND is TRUE');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Case 1: AND is FALSE');
    END IF;
    v_num1 := NULL;
    v_num2 := 20;
    IF v_num1 IS NOT NULL AND v_num2 IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Case 2: AND is TRUE');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Case 2: First NULL → AND is FALSE');
    END IF;
    v_num1 := 10;
    v_num2 := NULL;
    IF v_num1 IS NOT NULL AND v_num2 IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Case 3: AND is TRUE');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Case 3: Second NULL → AND is FALSE');
    END IF;
    v_num1 := NULL;
    v_num2 := NULL;
    IF v_num1 IS NOT NULL AND v_num2 IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Case 4: AND is TRUE');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Case 4: Both NULL → AND is FALSE');
    END IF;
END;
/
```

```
Procedure created.

0.05 seconds
```

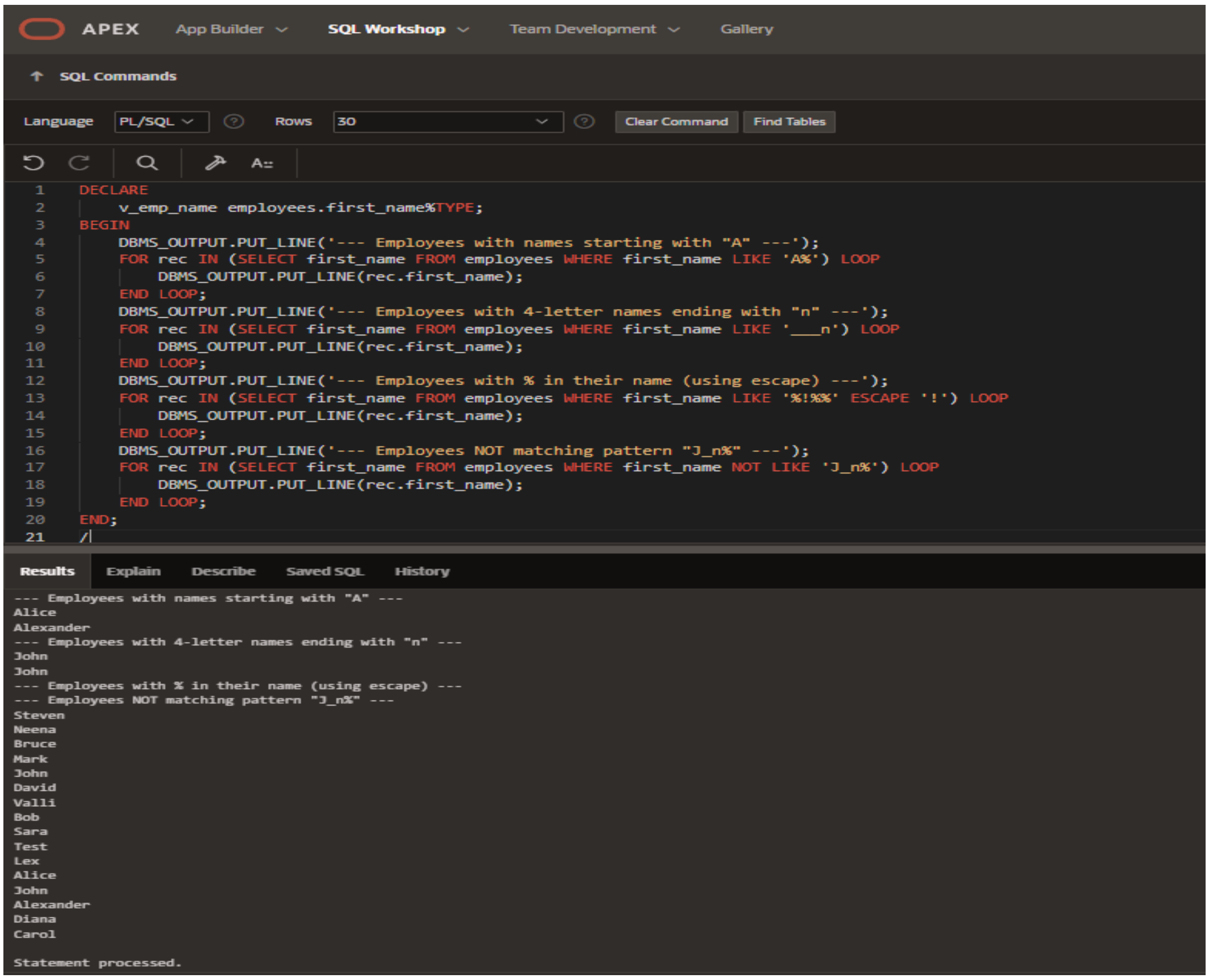


```
BEGIN
    check_and_null_numbers;
END;
/
```

```
Case 1: Both numbers NOT NULL → AND is TRUE
Case 2: First NULL → AND is FALSE
Case 3: Second NULL → AND is FALSE
Case 4: Both NULL → AND is FALSE

Statement processed.

0.01 seconds
```

PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

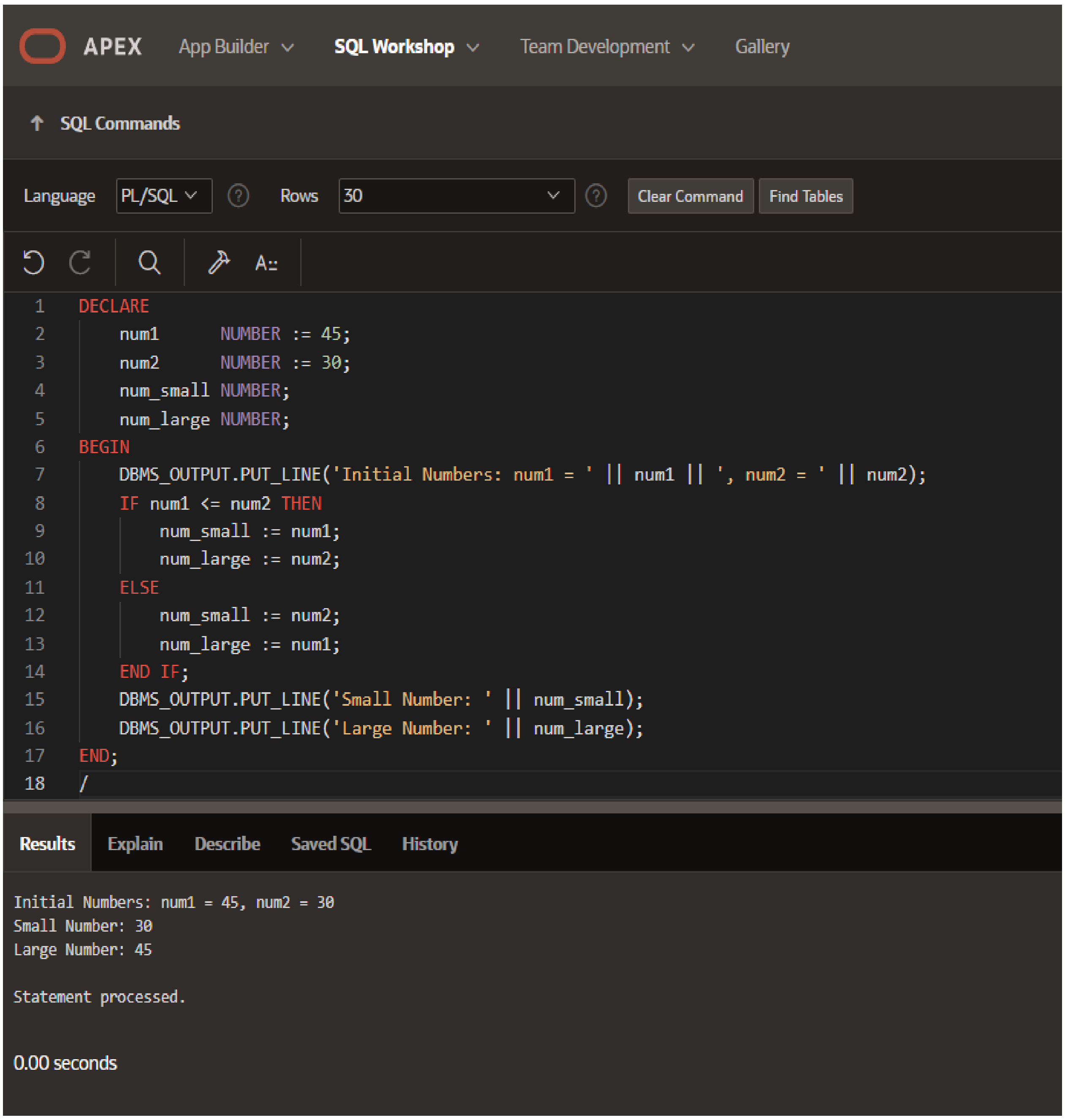


```
DECLARE
    v_emp_name employees.first_name%TYPE;
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Employees with names starting with "A" ---');
    FOR rec IN (SELECT first_name FROM employees WHERE first_name LIKE 'A%') LOOP
        DBMS_OUTPUT.PUT_LINE(rec.first_name);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('--- Employees with 4-letter names ending with "n" ---');
    FOR rec IN (SELECT first_name FROM employees WHERE first_name LIKE '___n') LOOP
        DBMS_OUTPUT.PUT_LINE(rec.first_name);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('--- Employees with % in their name (using escape) ---');
    FOR rec IN (SELECT first_name FROM employees WHERE first_name LIKE '%!%%' ESCAPE '!') LOOP
        DBMS_OUTPUT.PUT_LINE(rec.first_name);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('--- Employees NOT matching pattern "J_n%" ---');
    FOR rec IN (SELECT first_name FROM employees WHERE first_name NOT LIKE 'J_n%') LOOP
        DBMS_OUTPUT.PUT_LINE(rec.first_name);
    END LOOP;
END;
/
```

```
--- Employees with names starting with "A" ---
Alice
Alexander
--- Employees with 4-letter names ending with "n" ---
John
John
--- Employees with % in their name (using escape) ---
--- Employees NOT matching pattern "J_n%" ---
Steven
Neena
Bruce
Mark
John
David
Valli
Bob
Sara
Test
Lex
Alice
John
Alexander
Diana
Carol

Statement processed.
```

PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.



Language PL/SQL

Rows 30

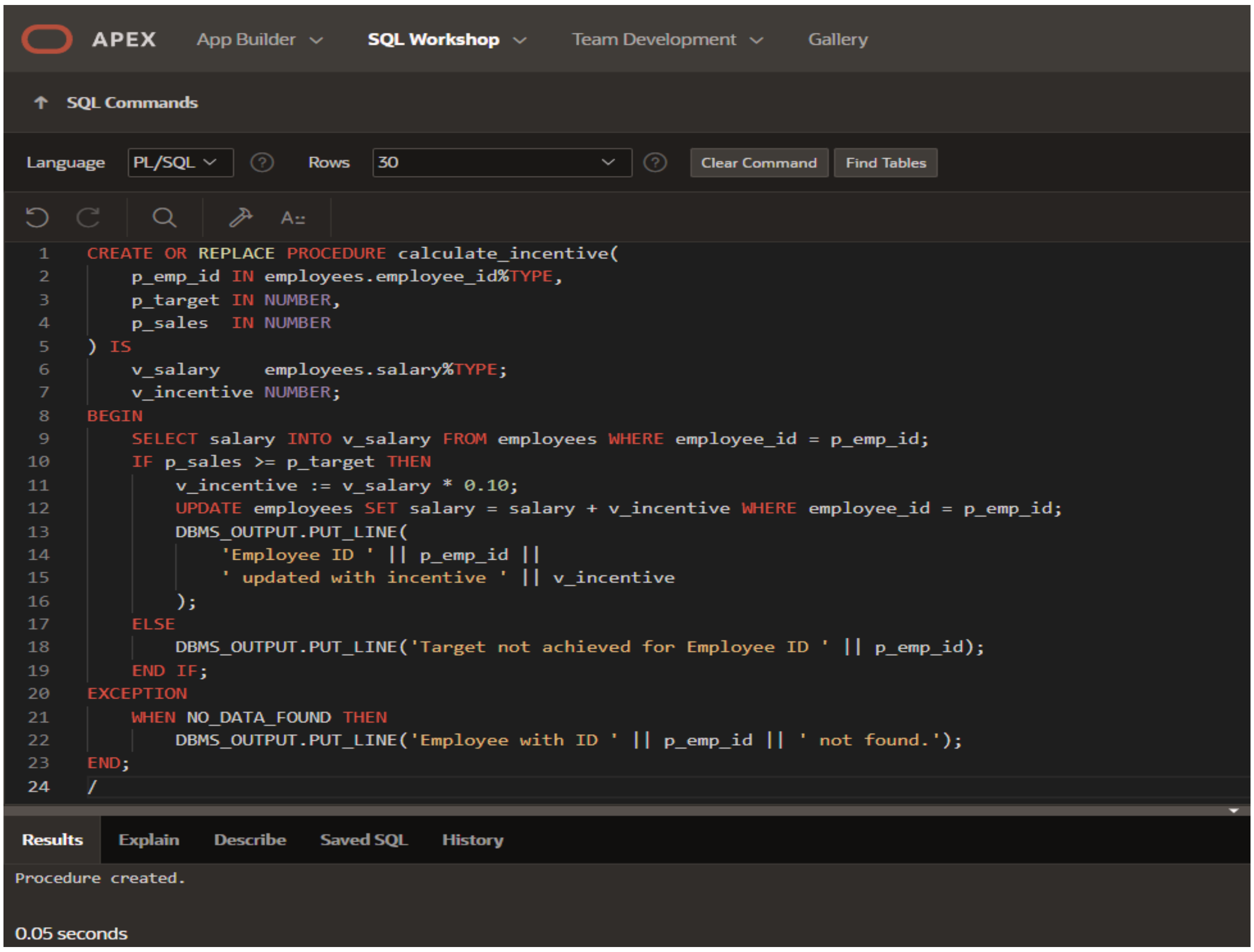
```
DECLARE
    num1      NUMBER := 45;
    num2      NUMBER := 30;
    num_small NUMBER;
    num_large NUMBER;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Initial Numbers: num1 = ' || num1 || ', num2 = ' || num2);
    IF num1 <= num2 THEN
        num_small := num1;
        num_large := num2;
    ELSE
        num_small := num2;
        num_large := num1;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Small Number: ' || num_small);
    DBMS_OUTPUT.PUT_LINE('Large Number: ' || num_large);
END;
/
```

```
Initial Numbers: num1 = 45, num2 = 30
Small Number: 30
Large Number: 45

Statement processed.
```

0.00 seconds

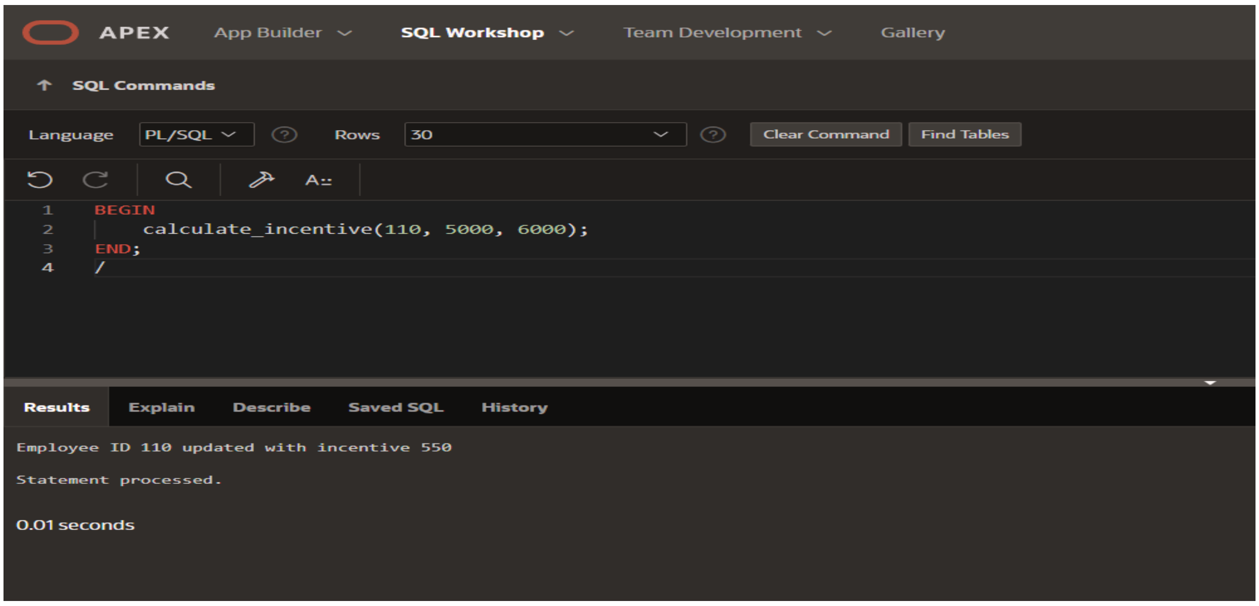
PROGRAM 7

Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.



```
CREATE OR REPLACE PROCEDURE calculate_incentive(
    p_emp_id IN employees.employee_id%TYPE,
    p_target IN NUMBER,
    p_sales  IN NUMBER
) IS
    v_salary    employees.salary%TYPE;
    v_incentive NUMBER;
BEGIN
    SELECT salary INTO v_salary FROM employees WHERE employee_id = p_emp_id;
    IF p_sales >= p_target THEN
        v_incentive := v_salary * 0.10;
        UPDATE employees SET salary = salary + v_incentive WHERE employee_id = p_emp_id;
        DBMS_OUTPUT.PUT_LINE(
            'Employee ID ' || p_emp_id ||
            ' updated with incentive ' || v_incentive
        );
    ELSE
        DBMS_OUTPUT.PUT_LINE('Target not achieved for Employee ID ' || p_emp_id);
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Employee with ID ' || p_emp_id || ' not found.');
END;
/
```

```
Procedure created.

0.05 seconds
```

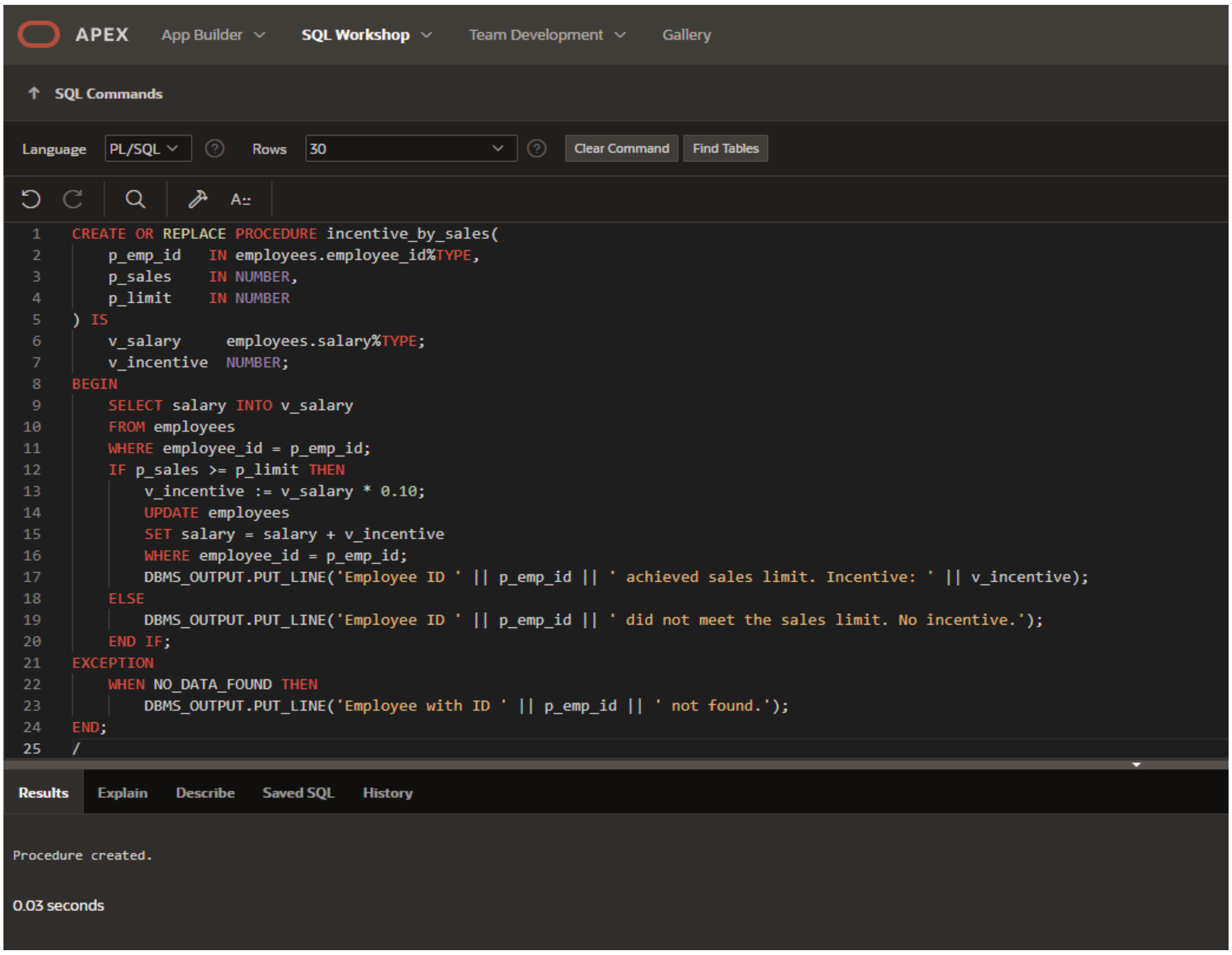


```
BEGIN
    calculate_incentive(110, 5000, 6000);
END;
/
```

```
Employee ID 110 updated with incentive 550

Statement processed.

0.01 seconds
```

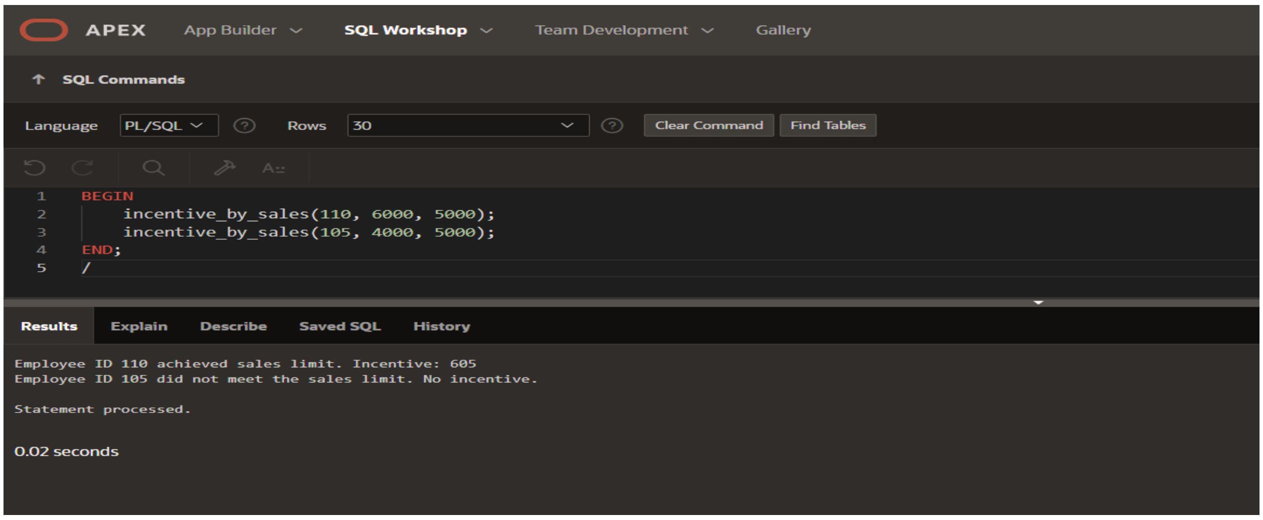
PROGRAM 8

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.



```
CREATE OR REPLACE PROCEDURE incentive_by_sales(
    p_emp_id    IN employees.employee_id%TYPE,
    p_sales     IN NUMBER,
    p_limit     IN NUMBER
) IS
    v_salary      employees.salary%TYPE;
    v_incentive   NUMBER;
BEGIN
    SELECT salary INTO v_salary
    FROM employees
    WHERE employee_id = p_emp_id;
    IF p_sales >= p_limit THEN
        v_incentive := v_salary * 0.10;
        UPDATE employees
        SET salary = salary + v_incentive
        WHERE employee_id = p_emp_id;
        DBMS_OUTPUT.PUT_LINE('Employee ID ' || p_emp_id || ' achieved sales limit. Incentive: ' || v_incentive);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Employee ID ' || p_emp_id || ' did not meet the sales limit. No incentive.');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Employee with ID ' || p_emp_id || ' not found.');
END;
/
```

```
Procedure created.

0.03 seconds
```

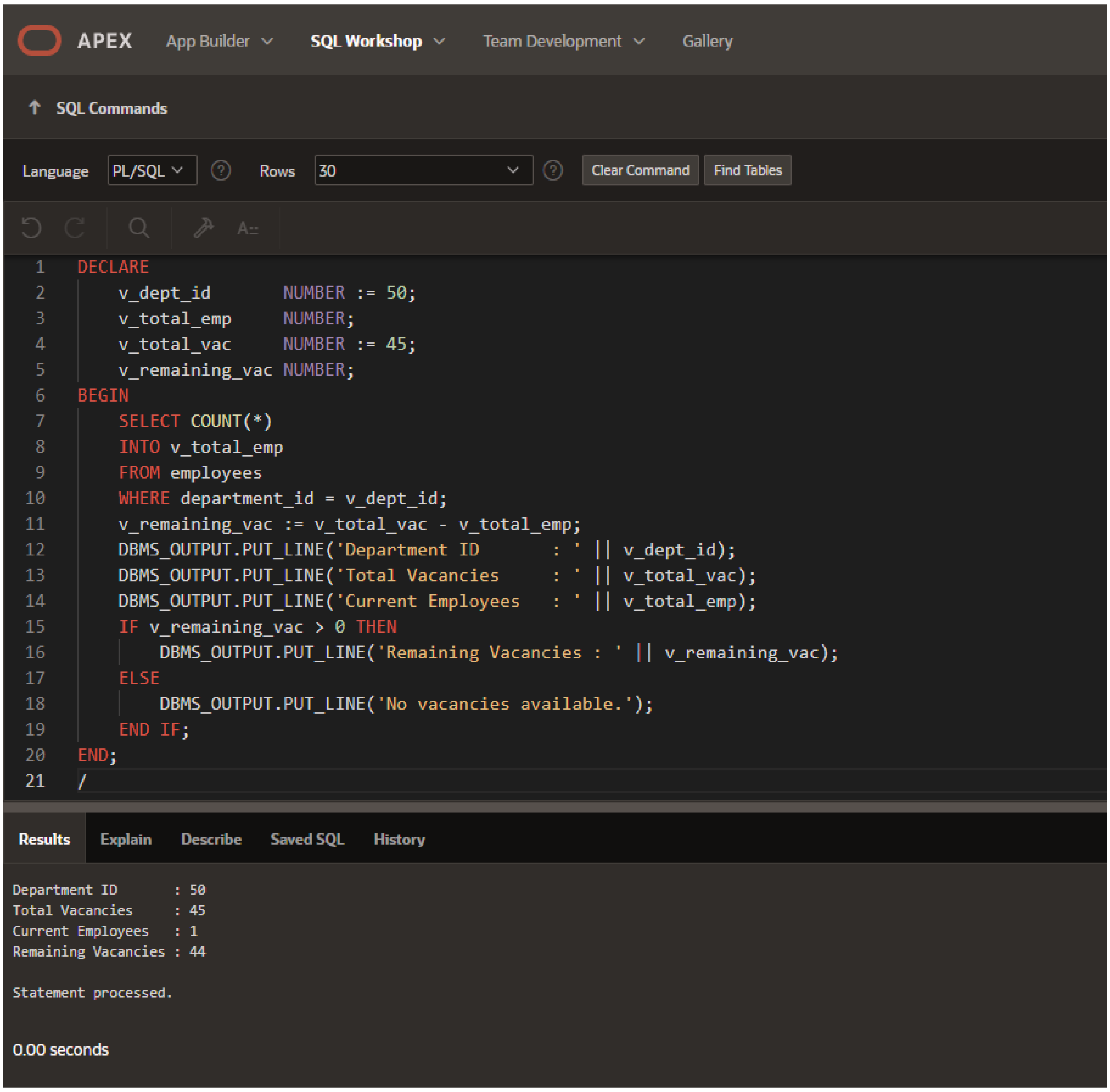


```
BEGIN
    incentive_by_sales(110, 6000, 5000);
    incentive_by_sales(105, 4000, 5000);
END;
/
```

```
Employee ID 110 achieved sales limit. Incentive: 605
Employee ID 105 did not meet the sales limit. No incentive.

Statement processed.

0.02 seconds
```

PROGRAM 9

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.



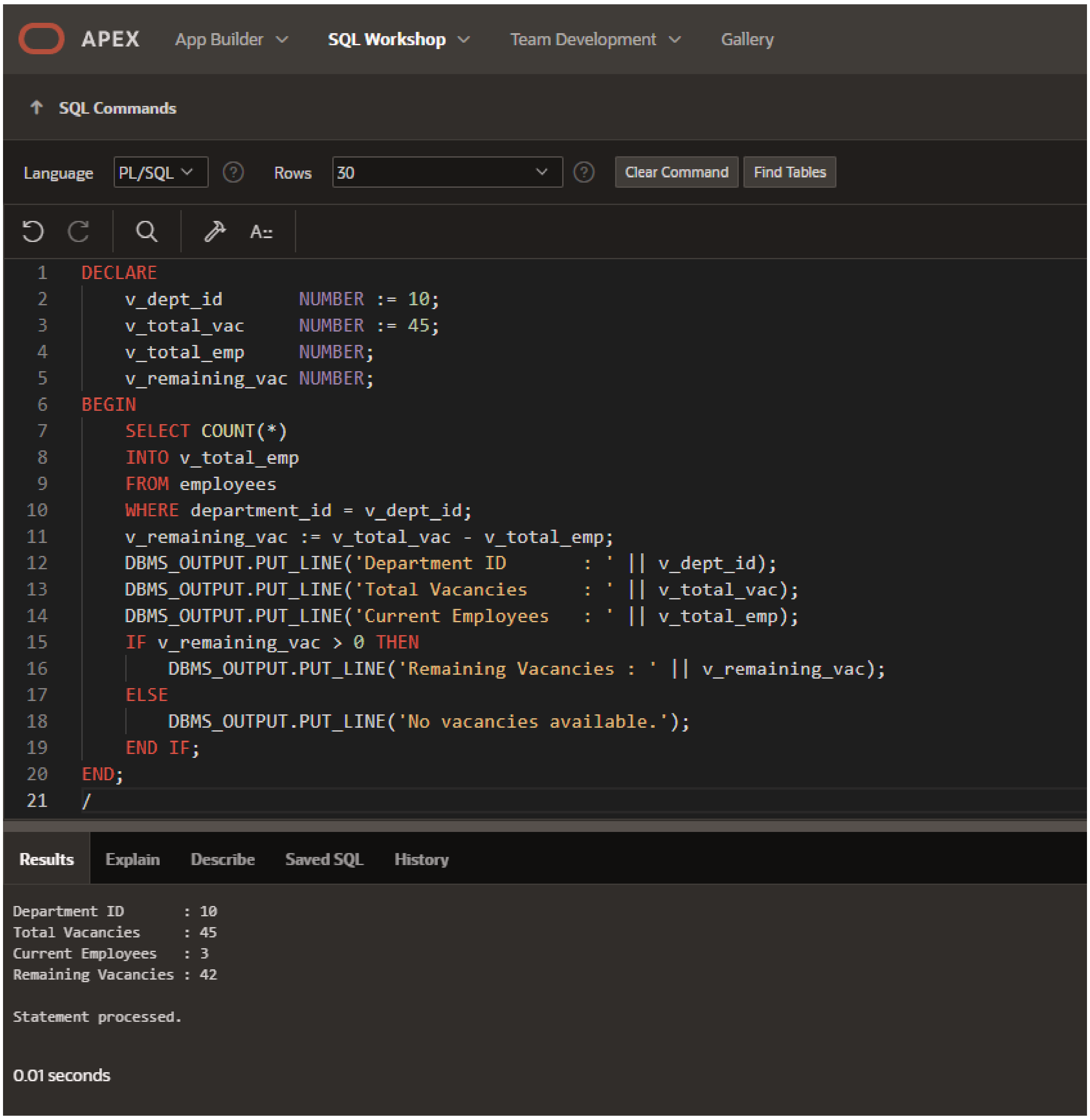
```
DECLARE
    v_dept_id       NUMBER := 50;
    v_total_emp     NUMBER;
    v_total_vac     NUMBER := 45;
    v_remaining_vac NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_total_emp
    FROM employees
    WHERE department_id = v_dept_id;
    v_remaining_vac := v_total_vac - v_total_emp;
    DBMS_OUTPUT.PUT_LINE('Department ID       : ' || v_dept_id);
    DBMS_OUTPUT.PUT_LINE('Total Vacancies     : ' || v_total_vac);
    DBMS_OUTPUT.PUT_LINE('Current Employees   : ' || v_total_emp);
    IF v_remaining_vac > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Remaining Vacancies : ' || v_remaining_vac);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No vacancies available.');
    END IF;
END;
/
```

```
Department ID       : 50
Total Vacancies     : 45
Current Employees   : 1
Remaining Vacancies : 44

Statement processed.
```

0.00 seconds

PROGRAM 10

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.



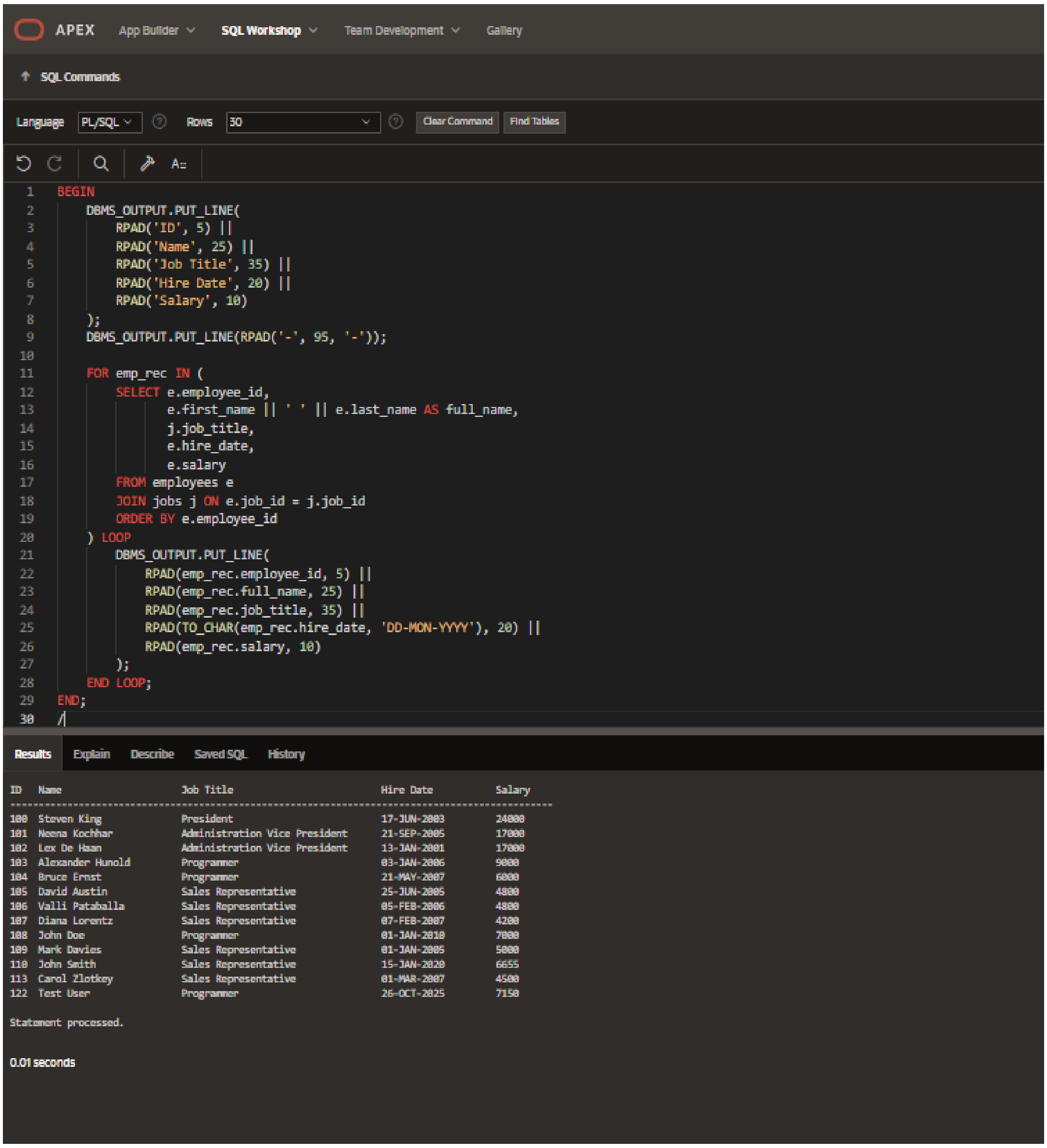
```
DECLARE
    v_dept_id       NUMBER := 10;
    v_total_vac     NUMBER := 45;
    v_total_emp     NUMBER;
    v_remaining_vac NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_total_emp
    FROM employees
    WHERE department_id = v_dept_id;
    v_remaining_vac := v_total_vac - v_total_emp;
    DBMS_OUTPUT.PUT_LINE('Department ID       : ' || v_dept_id);
    DBMS_OUTPUT.PUT_LINE('Total Vacancies     : ' || v_total_vac);
    DBMS_OUTPUT.PUT_LINE('Current Employees   : ' || v_total_emp);
    IF v_remaining_vac > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Remaining Vacancies : ' || v_remaining_vac);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No vacancies available.');
    END IF;
END;
/
```

```
Department ID       : 10
Total Vacancies     : 45
Current Employees   : 3
Remaining Vacancies : 42

Statement processed.
```

0.01 seconds

PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees.

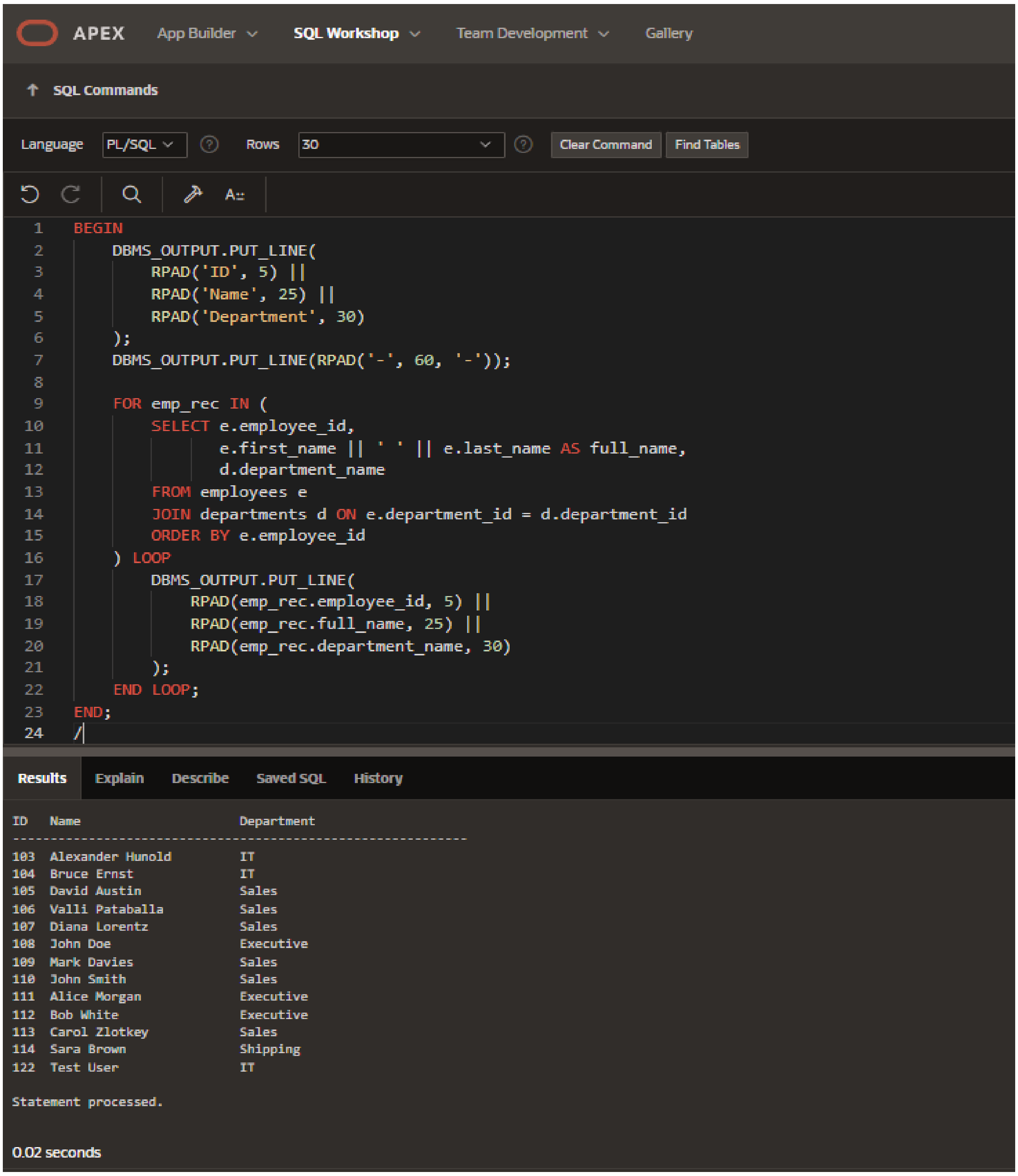


```sql
BEGIN
    DBMS_OUTPUT.PUT_LINE(
        RPAD('ID', 5) ||
        RPAD('Name', 25) ||
        RPAD('Job Title', 35) ||
        RPAD('Hire Date', 20) ||
        RPAD('Salary', 10)
    );
    DBMS_OUTPUT.PUT_LINE(RPAD('-', 95, '-'));

    FOR emp_rec IN (
        SELECT e.employee_id,
               e.first_name || ' ' || e.last_name AS full_name,
               j.job_title,
               e.hire_date,
               e.salary
        FROM employees e
        JOIN jobs j ON e.job_id = j.job_id
        ORDER BY e.employee_id
    ) LOOP
        DBMS_OUTPUT.PUT_LINE(
            RPAD(emp_rec.employee_id, 5) ||
            RPAD(emp_rec.full_name, 25) ||
            RPAD(emp_rec.job_title, 35) ||
            RPAD(TO_CHAR(emp_rec.hire_date, 'DD-MON-YYYY'), 20) ||
            RPAD(emp_rec.salary, 10)
        );
    END LOOP;
END;
/
```

```
ID   Name                     Job Title                          Hire Date           Salary
-----------------------------------------------------------------------------------------------
100  Steven King              President                          17-JUN-2003         24000
101  Neena Kochhar            Administration Vice President      21-SEP-2005         17000
102  Lex De Haan              Administration Vice President      13-JAN-2001         17000
103  Alexander Hunold         Programmer                         03-JAN-2006         9000
104  Bruce Ernst              Programmer                         21-MAY-2007         6000
105  David Austin             Sales Representative               25-JUN-2005         4800
106  Valli Pataballa          Sales Representative               05-FEB-2006         4800
107  Diana Lorentz            Sales Representative               07-FEB-2007         4200
108  John Doe                 Programmer                         01-JAN-2010         7000
109  Mark Davies              Sales Representative               01-JAN-2005         5000
110  John Smith               Sales Representative               15-JAN-2020         6655
113  Carol Zlotkey            Sales Representative               01-MAR-2007         4500
122  Test User                Programmer                         26-OCT-2025         7150

Statement processed.

0.01 seconds
```

PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department names of all employees.



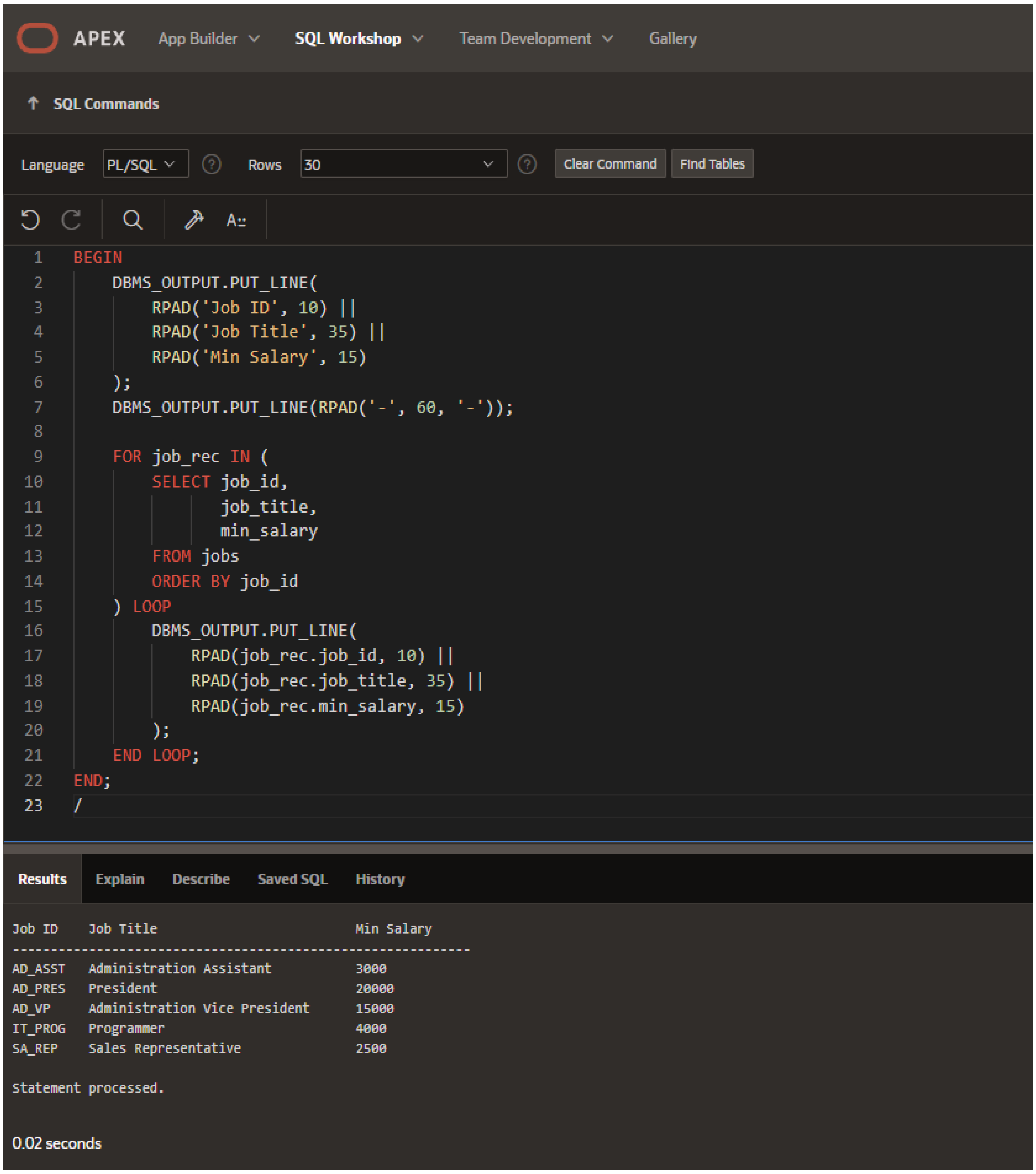
Language PL/SQL Rows 30

```sql
BEGIN
    DBMS_OUTPUT.PUT_LINE(
        RPAD('ID', 5) ||
        RPAD('Name', 25) ||
        RPAD('Department', 30)
    );
    DBMS_OUTPUT.PUT_LINE(RPAD('-', 60, '-'));

    FOR emp_rec IN (
        SELECT e.employee_id,
               e.first_name || ' ' || e.last_name AS full_name,
               d.department_name
        FROM employees e
        JOIN departments d ON e.department_id = d.department_id
        ORDER BY e.employee_id
    ) LOOP
        DBMS_OUTPUT.PUT_LINE(
            RPAD(emp_rec.employee_id, 5) ||
            RPAD(emp_rec.full_name, 25) ||
            RPAD(emp_rec.department_name, 30)
        );
    END LOOP;
END;
/
```

```
ID   Name                     Department
------------------------------------------------------------
103  Alexander Hunold         IT
104  Bruce Ernst              IT
105  David Austin             Sales
106  Valli Pataballa          Sales
107  Diana Lorentz            Sales
108  John Doe                 Executive
109  Mark Davies              Sales
110  John Smith               Sales
111  Alice Morgan             Executive
112  Bob White                Executive
113  Carol Zlotkey            Sales
114  Sara Brown               Shipping
122  Test User                IT

Statement processed.

0.02 seconds
```

PROGRAM 13

Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.



Language PL/SQL   Rows 30

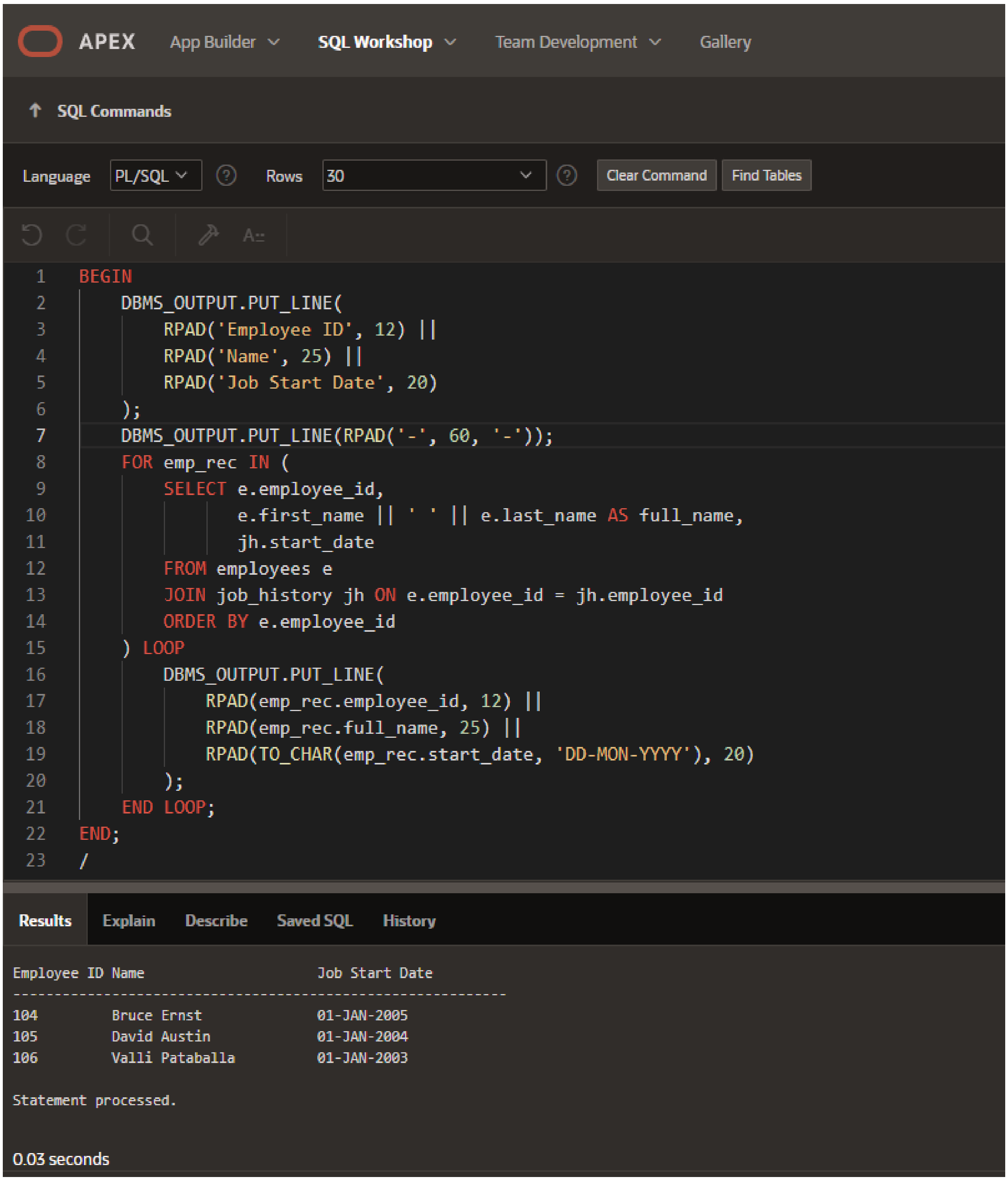
```sql
BEGIN
    DBMS_OUTPUT.PUT_LINE(
        RPAD('Job ID', 10) ||
        RPAD('Job Title', 35) ||
        RPAD('Min Salary', 15)
    );
    DBMS_OUTPUT.PUT_LINE(RPAD('-', 60, '-'));

    FOR job_rec IN (
        SELECT job_id,
               job_title,
               min_salary
        FROM jobs
        ORDER BY job_id
    ) LOOP
        DBMS_OUTPUT.PUT_LINE(
            RPAD(job_rec.job_id, 10) ||
            RPAD(job_rec.job_title, 35) ||
            RPAD(job_rec.min_salary, 15)
        );
    END LOOP;
END;
/
```

```
Job ID    Job Title                          Min Salary
------------------------------------------------------------
AD_ASST   Administration Assistant           3000
AD_PRES   President                          20000
AD_VP     Administration Vice President      15000
IT_PROG   Programmer                         4000
SA_REP    Sales Representative               2500

Statement processed.
```

0.02 seconds

PROGRAM 14

Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

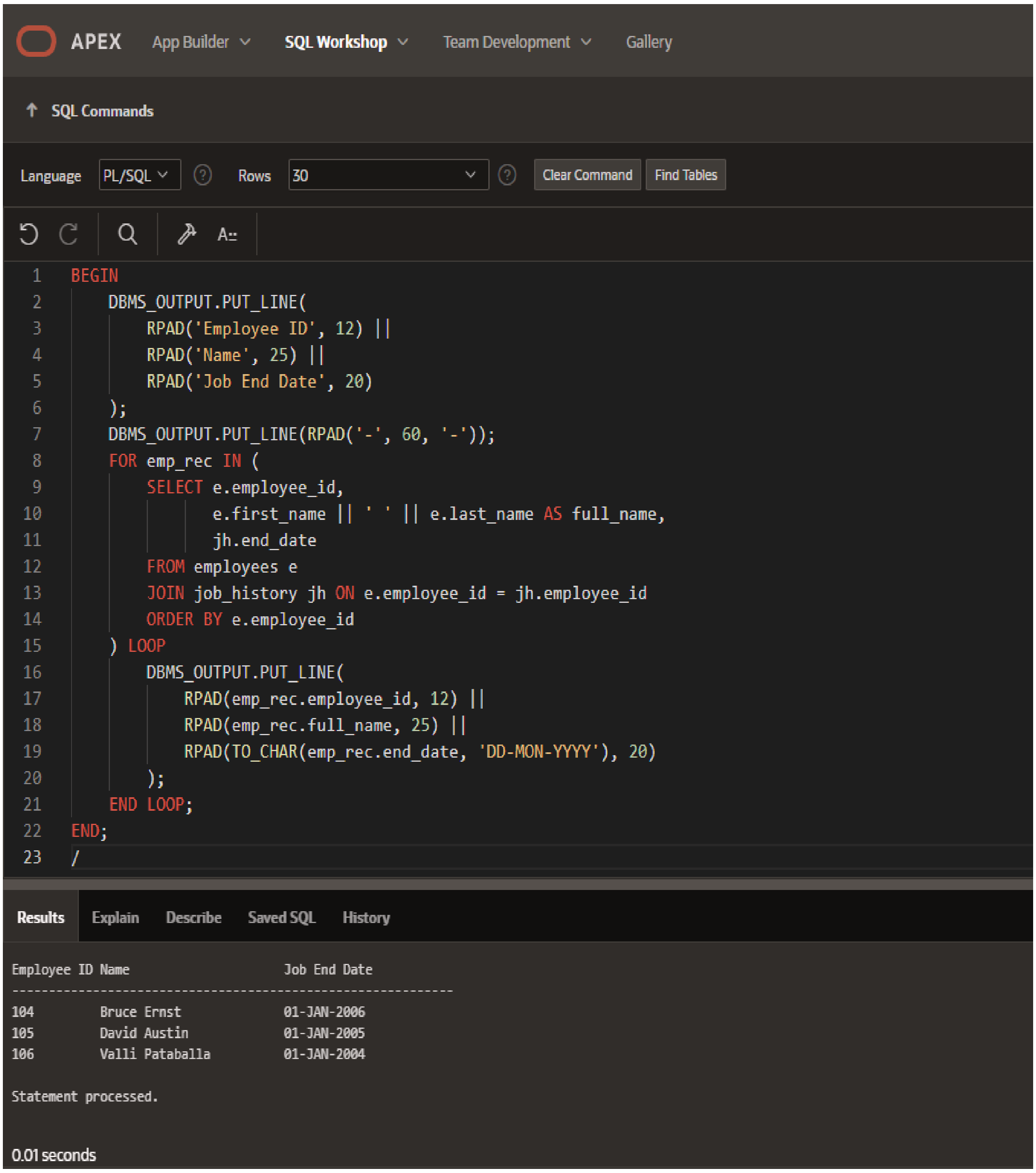


```
BEGIN
    DBMS_OUTPUT.PUT_LINE(
        RPAD('Employee ID', 12) ||
        RPAD('Name', 25) ||
        RPAD('Job Start Date', 20)
    );
    DBMS_OUTPUT.PUT_LINE(RPAD('-', 60, '-'));
    FOR emp_rec IN (
        SELECT e.employee_id,
               e.first_name || ' ' || e.last_name AS full_name,
               jh.start_date
        FROM employees e
        JOIN job_history jh ON e.employee_id = jh.employee_id
        ORDER BY e.employee_id
    ) LOOP
        DBMS_OUTPUT.PUT_LINE(
            RPAD(emp_rec.employee_id, 12) ||
            RPAD(emp_rec.full_name, 25) ||
            RPAD(TO_CHAR(emp_rec.start_date, 'DD-MON-YYYY'), 20)
        );
    END LOOP;
END;
/
```

```
Employee ID Name                     Job Start Date
------------------------------------------------------------
104         Bruce Ernst              01-JAN-2005
105         David Austin             01-JAN-2004
106         Valli Pataballa          01-JAN-2003

Statement processed.

0.03 seconds
```

PROGRAM 15

Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.



Language PL/SQL   Rows 30

```sql
BEGIN
    DBMS_OUTPUT.PUT_LINE(
        RPAD('Employee ID', 12) ||
        RPAD('Name', 25) ||
        RPAD('Job End Date', 20)
    );
    DBMS_OUTPUT.PUT_LINE(RPAD('-', 60, '-'));
    FOR emp_rec IN (
        SELECT e.employee_id,
               e.first_name || ' ' || e.last_name AS full_name,
               jh.end_date
        FROM employees e
        JOIN job_history jh ON e.employee_id = jh.employee_id
        ORDER BY e.employee_id
    ) LOOP
        DBMS_OUTPUT.PUT_LINE(
            RPAD(emp_rec.employee_id, 12) ||
            RPAD(emp_rec.full_name, 25) ||
            RPAD(TO_CHAR(emp_rec.end_date, 'DD-MON-YYYY'), 20)
        );
    END LOOP;
END;
/
```

```
Employee ID Name                     Job End Date
------------------------------------------------------------
104         Bruce Ernst              01-JAN-2006
105         David Austin             01-JAN-2005
106         Valli Pataballa          01-JAN-2004

Statement processed.
```

0.01 seconds