

# 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.

The screenshot displays the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' section is active, showing a PL/SQL block with line numbers 1 through 22. The block declares variables for employee ID, salary, and incentive, then calculates the incentive as 10% of the salary for employee ID 110. It uses DBMS\_OUTPUT.PUT\_LINE to display the results. An exception block handles 'NO\_DATA\_FOUND' and other errors. The 'Results' tab at the bottom shows the output: 'Employee ID: 110', 'Salary: 5500', 'Incentive (10%): 550', and 'Statement processed.' in 0.01 seconds.

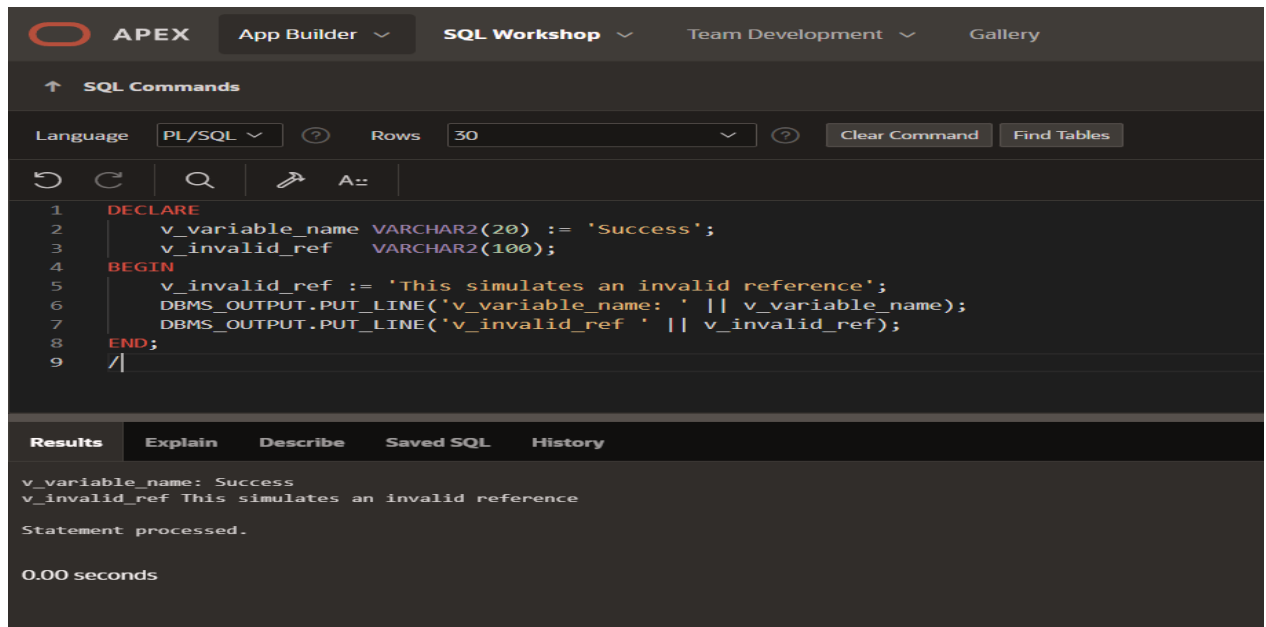
```
1 DECLARE
2     v_emp_id    employees.employee_id%TYPE := 110;
3     v_salary    employees.salary%TYPE;
4     v_incentive NUMBER;
5 BEGIN
6     SELECT salary
7     INTO v_salary
8     FROM employees
9     WHERE employee_id = v_emp_id;
10
11     v_incentive := v_salary * 0.10;
12
13     DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_emp_id);
14     DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);
15     DBMS_OUTPUT.PUT_LINE('Incentive (10%): ' || v_incentive);
16 EXCEPTION
17     WHEN NO_DATA_FOUND THEN
18         DBMS_OUTPUT.PUT_LINE('Employee with ID ' || v_emp_id || ' not found.');
```

Results Explain Describe Saved SQL History

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.



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' section is active, showing a PL/SQL block with the following code:

```
1 DECLARE
2   v_variable_name VARCHAR2(20) := 'Success';
3   v_invalid_ref   VARCHAR2(100);
4 BEGIN
5   v_invalid_ref := 'This simulates an invalid reference';
6   DBMS_OUTPUT.PUT_LINE('v_variable_name: ' || v_variable_name);
7   DBMS_OUTPUT.PUT_LINE('v_invalid_ref ' || v_invalid_ref);
8 END;
9 /
```

The 'Results' tab is selected, displaying the output of the execution:

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

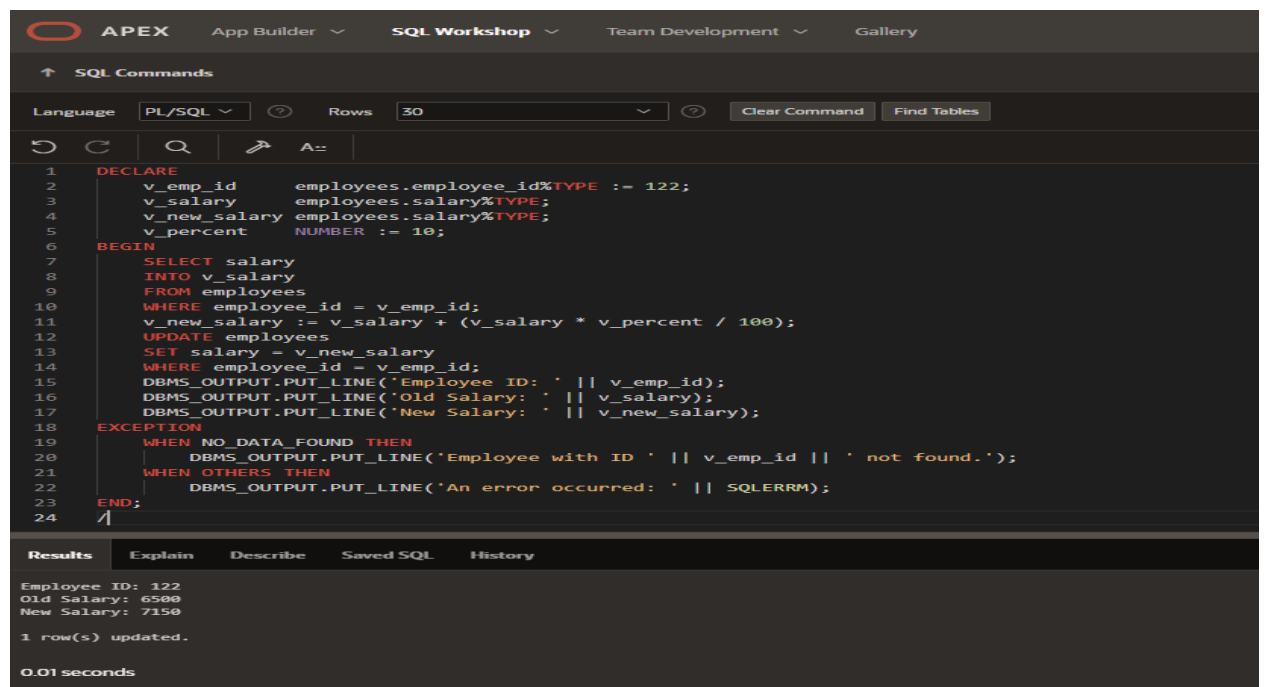
Statement processed.

0.00 seconds
```

## PROGRAM 3

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

Sample table: employees



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' section is active, showing a PL/SQL block with the following code:

```
1 DECLARE
2   v_emp_id      employees.employee_id%TYPE := 122;
3   v_salary      employees.salary%TYPE;
4   v_new_salary  employees.salary%TYPE;
5   v_percent     NUMBER := 10;
6 BEGIN
7   SELECT salary
8   INTO v_salary
9   FROM employees
10  WHERE employee_id = v_emp_id;
11  v_new_salary := v_salary + (v_salary * v_percent / 100);
12  UPDATE employees
13  SET salary = v_new_salary
14  WHERE employee_id = v_emp_id;
15  DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_emp_id);
16  DBMS_OUTPUT.PUT_LINE('Old Salary: ' || v_salary);
17  DBMS_OUTPUT.PUT_LINE('New Salary: ' || v_new_salary);
18 EXCEPTION
19  WHEN NO_DATA_FOUND THEN
20    DBMS_OUTPUT.PUT_LINE('Employee with ID ' || v_emp_id || ' not found.');
```

The 'Results' tab is selected, displaying the output of the execution:

```
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.

The screenshot displays the APEX SQL Workshop interface. At the top, there are navigation tabs: APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. Below the tabs, the 'SQL Commands' section is active, showing a PL/SQL block with 34 lines of code. The code defines a procedure named 'check\_and\_null\_numbers' that uses the 'IS [NOT] NULL' operator and the 'AND' operator to check the null status of two variables, v\_num1 and v\_num2. The procedure uses 'DBMS\_OUTPUT.PUT\_LINE' to output the results of these checks. The code is as follows:

```
1 CREATE OR REPLACE PROCEDURE check_and_null_numbers IS
2     v_num1 NUMBER;
3     v_num2 NUMBER;
4 BEGIN
5     v_num1 := 10;
6     v_num2 := 20;
7     IF v_num1 IS NOT NULL AND v_num2 IS NOT NULL THEN
8         DBMS_OUTPUT.PUT_LINE('Case 1: Both numbers NOT NULL → AND is TRUE');
9     ELSE
10        DBMS_OUTPUT.PUT_LINE('Case 1: AND is FALSE');
11    END IF;
12    v_num1 := NULL;
13    v_num2 := 20;
14    IF v_num1 IS NOT NULL AND v_num2 IS NOT NULL THEN
15        DBMS_OUTPUT.PUT_LINE('Case 2: AND is TRUE');
16    ELSE
17        DBMS_OUTPUT.PUT_LINE('Case 2: First NULL → AND is FALSE');
18    END IF;
19    v_num1 := 10;
20    v_num2 := NULL;
21    IF v_num1 IS NOT NULL AND v_num2 IS NOT NULL THEN
22        DBMS_OUTPUT.PUT_LINE('Case 3: AND is TRUE');
23    ELSE
24        DBMS_OUTPUT.PUT_LINE('Case 3: Second NULL → AND is FALSE');
25    END IF;
26    v_num1 := NULL;
27    v_num2 := NULL;
28    IF v_num1 IS NOT NULL AND v_num2 IS NOT NULL THEN
29        DBMS_OUTPUT.PUT_LINE('Case 4: AND is TRUE');
30    ELSE
31        DBMS_OUTPUT.PUT_LINE('Case 4: Both NULL → AND is FALSE');
32    END IF;
33 END;
34 /
```

Below the code editor, the 'Results' tab is selected, showing the output of the procedure execution. The output consists of four lines, each representing a case where the procedure was executed. The first line is 'Case 1: Both numbers NOT NULL → AND is TRUE'. The second line is 'Case 2: First NULL → AND is FALSE'. The third line is 'Case 3: Second NULL → AND is FALSE'. The fourth line is 'Case 4: Both NULL → AND is FALSE'. The execution time is shown as 0.05 seconds.

Results Explain Describe Saved SQL History

Procedure created.

0.05 seconds

**APEX** App Builder SQL Workshop Team Development Gallery

↑ SQL Commands

Language: PL/SQL Rows: 30 Clear Command Find Tables

```

1 BEGIN
2     check_and_null_numbers;
3 END;
4 /

```

**Results** Explain Describe Saved SQL History

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.

**APEX** App Builder SQL Workshop Team Development Gallery

↑ SQL Commands

Language: PL/SQL Rows: 30 Clear Command Find Tables

```

1 DECLARE
2     v_emp_name employees.first_name%TYPE;
3 BEGIN
4     DBMS_OUTPUT.PUT_LINE('--- Employees with names starting with "A" ---');
5     FOR rec IN (SELECT first_name FROM employees WHERE first_name LIKE 'A%') LOOP
6         DBMS_OUTPUT.PUT_LINE(rec.first_name);
7     END LOOP;
8     DBMS_OUTPUT.PUT_LINE('--- Employees with 4-letter names ending with "n" ---');
9     FOR rec IN (SELECT first_name FROM employees WHERE first_name LIKE '____n') LOOP
10        DBMS_OUTPUT.PUT_LINE(rec.first_name);
11    END LOOP;
12    DBMS_OUTPUT.PUT_LINE('--- Employees with % in their name (using escape) ---');
13    FOR rec IN (SELECT first_name FROM employees WHERE first_name LIKE '%!%' ESCAPE '!') LOOP
14        DBMS_OUTPUT.PUT_LINE(rec.first_name);
15    END LOOP;
16    DBMS_OUTPUT.PUT_LINE('--- Employees NOT matching pattern "J_n%" ---');
17    FOR rec IN (SELECT first_name FROM employees WHERE first_name NOT LIKE 'J_n%') LOOP
18        DBMS_OUTPUT.PUT_LINE(rec.first_name);
19    END LOOP;
20 END;
21 /

```

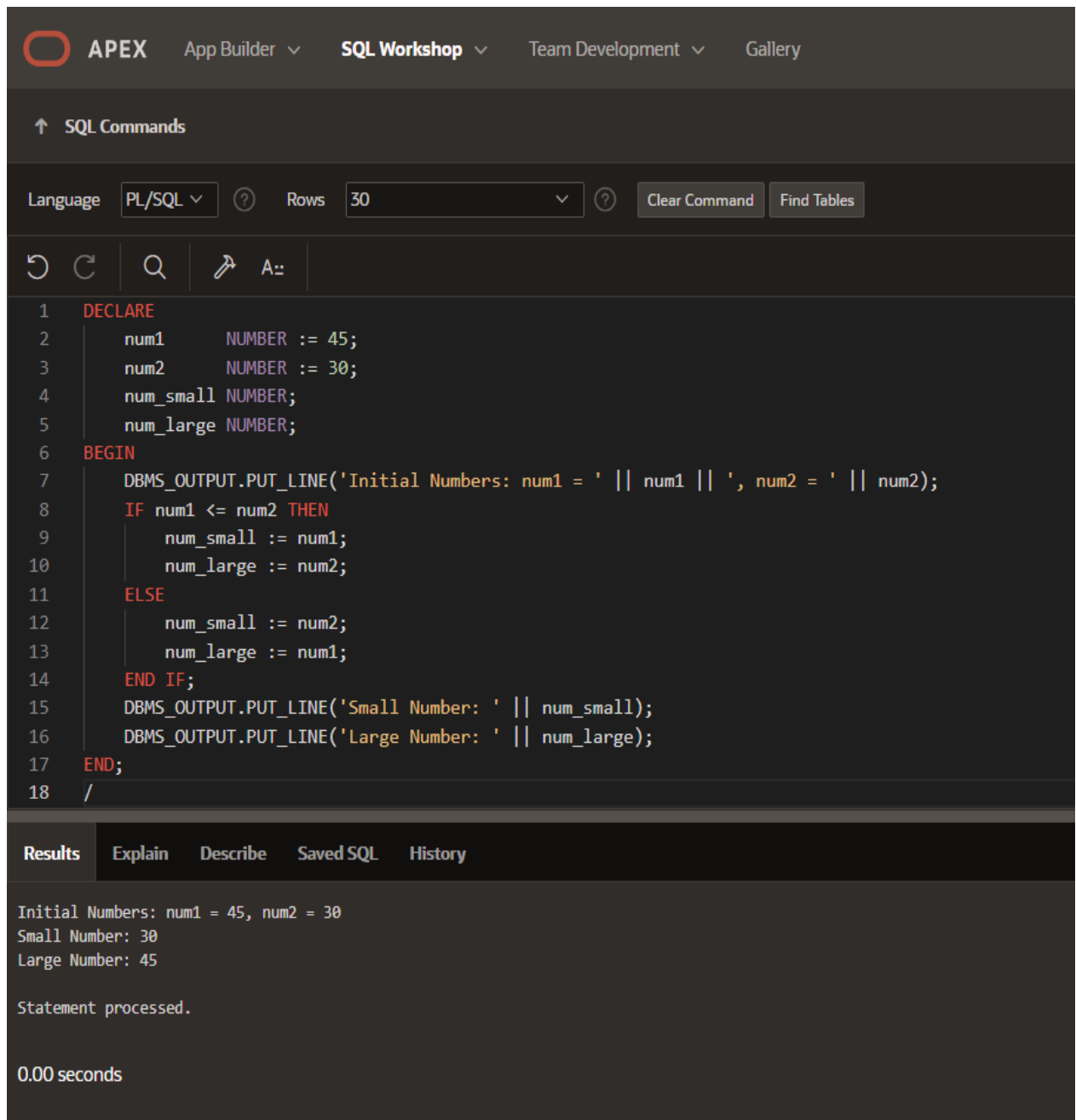
**Results** Explain Describe Saved SQL History

--- 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.



The screenshot displays the APEX SQL Workshop interface. At the top, there are navigation links: APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. Below these, the 'SQL Commands' section is active. The 'Language' is set to 'PL/SQL' and 'Rows' is set to '30'. There are buttons for 'Clear Command' and 'Find Tables'. The main area contains a PL/SQL program with line numbers 1 through 18. The program declares two variables, num1 and num2, and two output variables, num\_small and num\_large. It then uses an IF-THEN-ELSE statement to compare num1 and num2, assigning the smaller value to num\_small and the larger value to num\_large. Finally, it outputs the results using DBMS\_OUTPUT.PUT\_LINE. The 'Results' tab at the bottom shows the execution output: 'Initial Numbers: num1 = 45, num2 = 30', 'Small Number: 30', 'Large Number: 45', 'Statement processed.', and '0.00 seconds'.

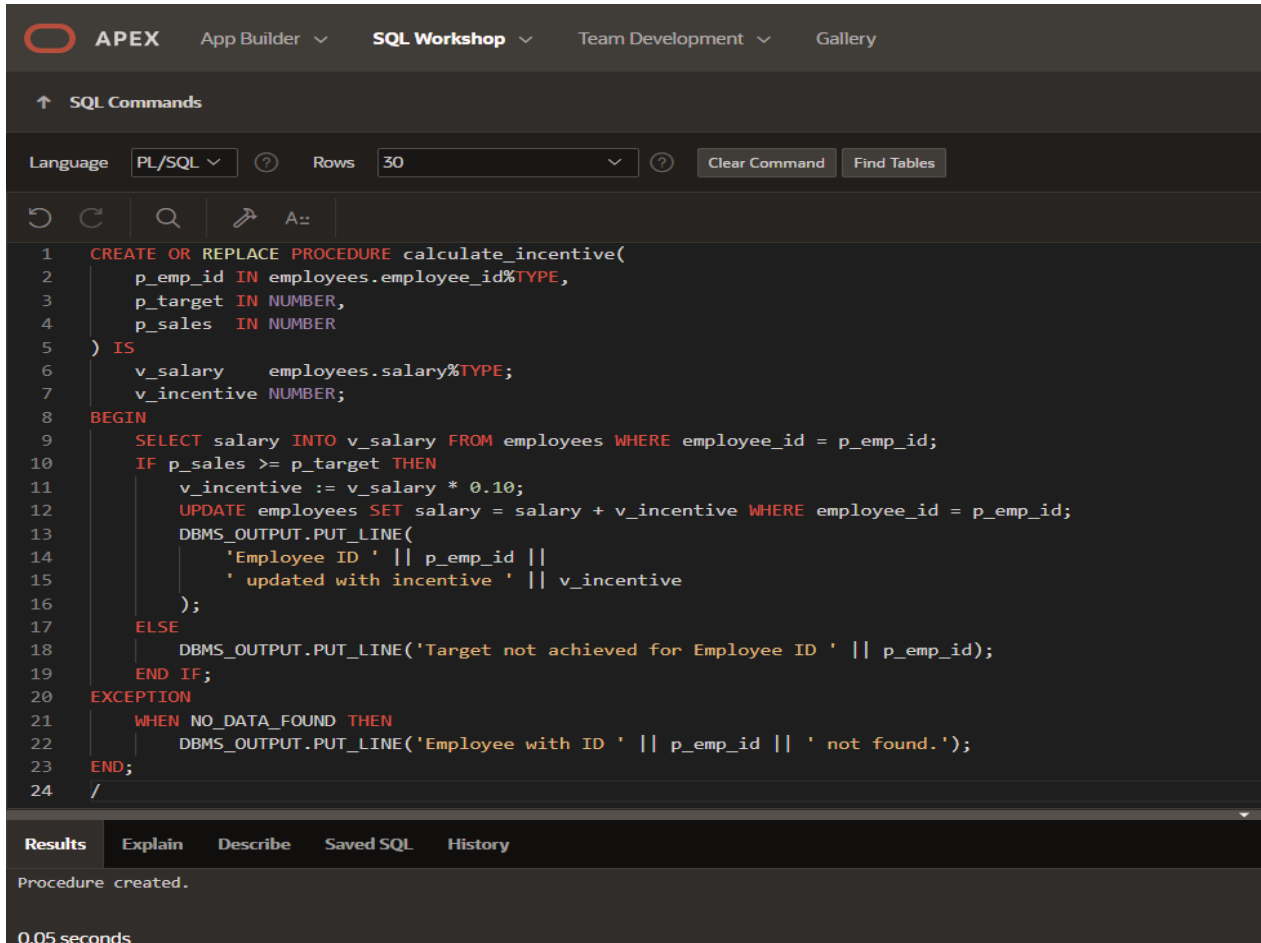
```
1 DECLARE
2     num1      NUMBER := 45;
3     num2      NUMBER := 30;
4     num_small NUMBER;
5     num_large NUMBER;
6 BEGIN
7     DBMS_OUTPUT.PUT_LINE('Initial Numbers: num1 = ' || num1 || ', num2 = ' || num2);
8     IF num1 <= num2 THEN
9         num_small := num1;
10        num_large := num2;
11    ELSE
12        num_small := num2;
13        num_large := num1;
14    END IF;
15    DBMS_OUTPUT.PUT_LINE('Small Number: ' || num_small);
16    DBMS_OUTPUT.PUT_LINE('Large Number: ' || num_large);
17 END;
18 /
```

**Results** Explain Describe Saved SQL History

Initial Numbers: num1 = 45, num2 = 30  
Small Number: 30  
Large Number: 45  
Statement processed.  
0.00 seconds

## PROGRAM 7

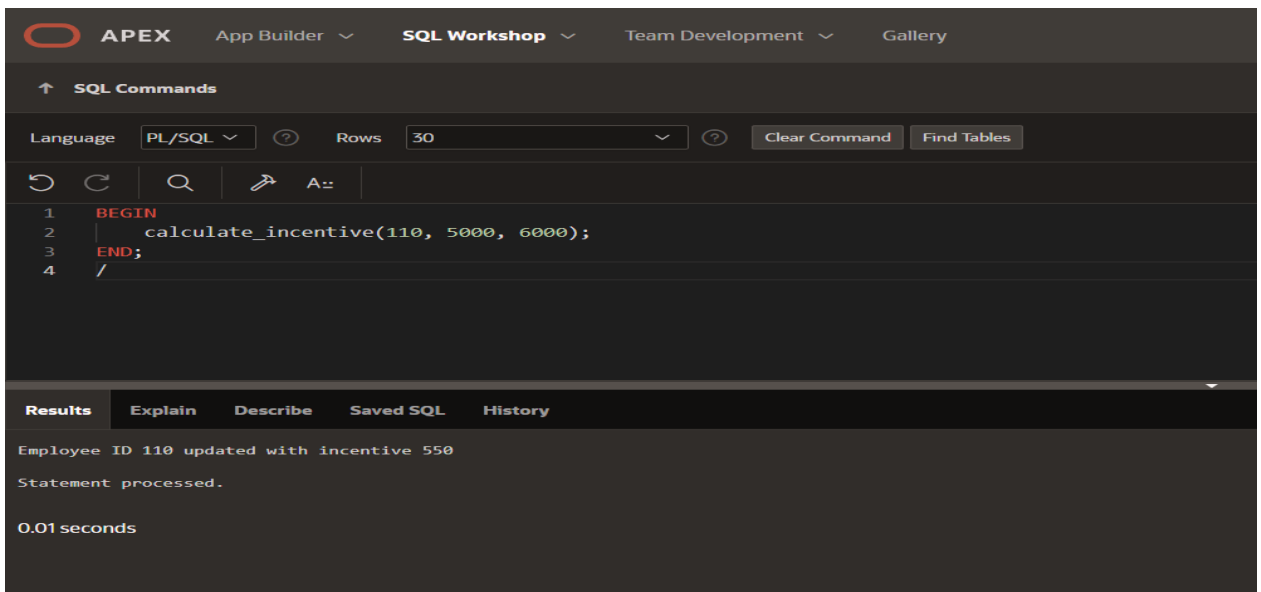
Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below the navigation bar, the 'SQL Commands' section is active. The 'Language' dropdown is set to 'PL/SQL', and the 'Rows' dropdown is set to '30'. The 'Clear Command' and 'Find Tables' buttons are visible. The main editor area contains the following PL/SQL code:

```
1 CREATE OR REPLACE PROCEDURE calculate_incentive(  
2     p_emp_id IN employees.employee_id%TYPE,  
3     p_target IN NUMBER,  
4     p_sales IN NUMBER  
5 ) IS  
6     v_salary employees.salary%TYPE;  
7     v_incentive NUMBER;  
8 BEGIN  
9     SELECT salary INTO v_salary FROM employees WHERE employee_id = p_emp_id;  
10    IF p_sales >= p_target THEN  
11        v_incentive := v_salary * 0.10;  
12        UPDATE employees SET salary = salary + v_incentive WHERE employee_id = p_emp_id;  
13        DBMS_OUTPUT.PUT_LINE(  
14            'Employee ID ' || p_emp_id ||  
15            ' updated with incentive ' || v_incentive  
16        );  
17    ELSE  
18        DBMS_OUTPUT.PUT_LINE('Target not achieved for Employee ID ' || p_emp_id);  
19    END IF;  
20 EXCEPTION  
21    WHEN NO_DATA_FOUND THEN  
22        DBMS_OUTPUT.PUT_LINE('Employee with ID ' || p_emp_id || ' not found.');
```

The 'Results' tab is selected, showing the message 'Procedure created.' and the execution time '0.05 seconds'.



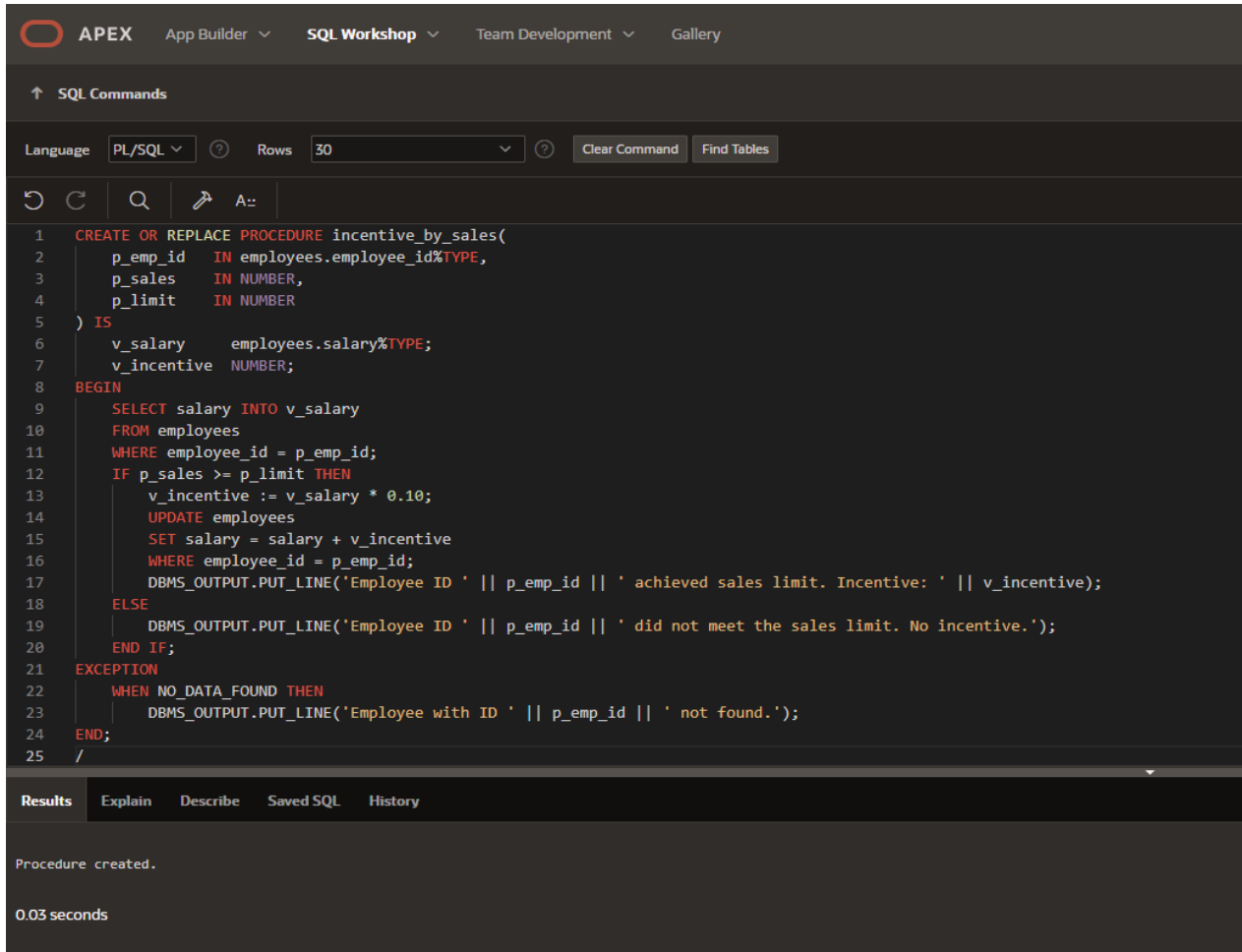
The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below the navigation bar, the 'SQL Commands' section is active. The 'Language' dropdown is set to 'PL/SQL', and the 'Rows' dropdown is set to '30'. The 'Clear Command' and 'Find Tables' buttons are visible. The main editor area contains the following PL/SQL code:

```
1 BEGIN  
2     calculate_incentive(110, 5000, 6000);  
3 END;  
4 /
```

The 'Results' tab is selected, showing the message 'Employee ID 110 updated with incentive 550' and 'Statement processed.' and the execution time '0.01 seconds'.

## PROGRAM 8

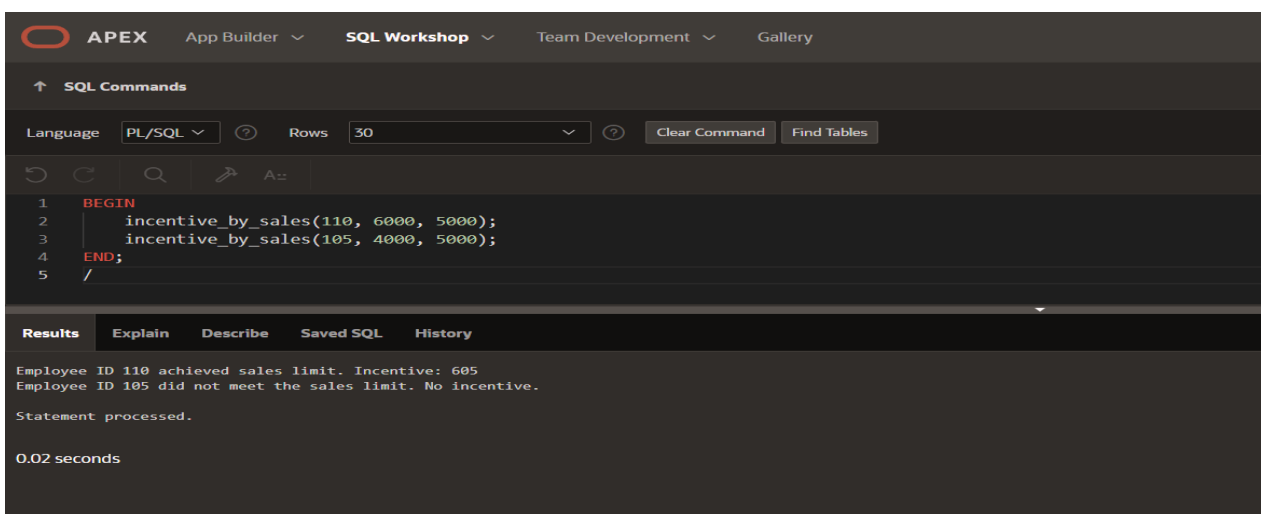
Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below the navigation bar, the 'SQL Commands' section is active. The 'Language' is set to 'PL/SQL', 'Rows' is set to '30', and there are buttons for 'Clear Command' and 'Find Tables'. The main editor displays the following PL/SQL code:

```
1 CREATE OR REPLACE PROCEDURE incentive_by_sales(  
2     p_emp_id IN employees.employee_id%TYPE,  
3     p_sales IN NUMBER,  
4     p_limit IN NUMBER  
5 ) IS  
6     v_salary employees.salary%TYPE;  
7     v_incentive NUMBER;  
8 BEGIN  
9     SELECT salary INTO v_salary  
10    FROM employees  
11   WHERE employee_id = p_emp_id;  
12   IF p_sales >= p_limit THEN  
13       v_incentive := v_salary * 0.10;  
14       UPDATE employees  
15          SET salary = salary + v_incentive  
16         WHERE employee_id = p_emp_id;  
17       DBMS_OUTPUT.PUT_LINE('Employee ID ' || p_emp_id || ' achieved sales limit. Incentive: ' || v_incentive);  
18   ELSE  
19       DBMS_OUTPUT.PUT_LINE('Employee ID ' || p_emp_id || ' did not meet the sales limit. No incentive.');20   END IF;  
21 EXCEPTION  
22   WHEN NO_DATA_FOUND THEN  
23       DBMS_OUTPUT.PUT_LINE('Employee with ID ' || p_emp_id || ' not found.');24 END;  
25 /
```

Below the code editor, the 'Results' tab is selected, showing the message 'Procedure created.' and the execution time '0.03 seconds'.



The screenshot shows the APEX SQL Workshop interface with the 'SQL Commands' section active. The 'Language' is set to 'PL/SQL', 'Rows' is set to '30', and there are buttons for 'Clear Command' and 'Find Tables'. The main editor displays the following PL/SQL code:

```
1 BEGIN  
2     incentive_by_sales(110, 6000, 5000);  
3     incentive_by_sales(105, 4000, 5000);  
4 END;  
5 /
```

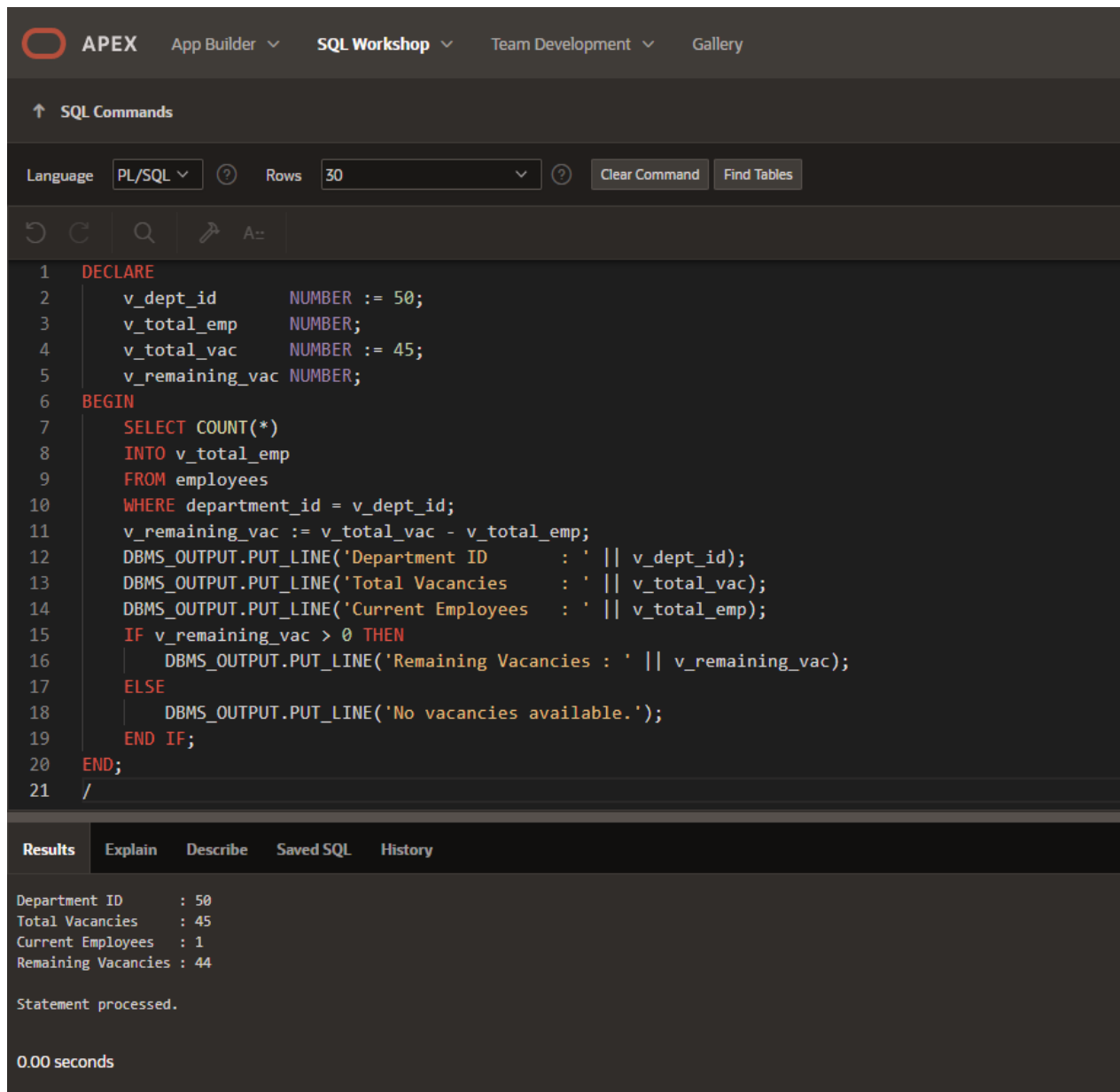
Below the code editor, the 'Results' tab is selected, showing the output of the procedure execution:

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

The message 'Statement processed.' is displayed, along with the execution time '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.



The screenshot displays the APEX SQL Workshop interface. At the top, there are navigation tabs: APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. Below the tabs, the 'SQL Commands' section is visible. It includes a 'Language' dropdown set to 'PL/SQL', a 'Rows' dropdown set to '30', and buttons for 'Clear Command' and 'Find Tables'. The main area shows a PL/SQL program with the following code:

```
1 DECLARE
2     v_dept_id      NUMBER := 50;
3     v_total_emp     NUMBER;
4     v_total_vac     NUMBER := 45;
5     v_remaining_vac NUMBER;
6 BEGIN
7     SELECT COUNT(*)
8     INTO v_total_emp
9     FROM employees
10    WHERE department_id = v_dept_id;
11    v_remaining_vac := v_total_vac - v_total_emp;
12    DBMS_OUTPUT.PUT_LINE('Department ID      : ' || v_dept_id);
13    DBMS_OUTPUT.PUT_LINE('Total Vacancies   : ' || v_total_vac);
14    DBMS_OUTPUT.PUT_LINE('Current Employees : ' || v_total_emp);
15    IF v_remaining_vac > 0 THEN
16        DBMS_OUTPUT.PUT_LINE('Remaining Vacancies : ' || v_remaining_vac);
17    ELSE
18        DBMS_OUTPUT.PUT_LINE('No vacancies available.');
```

The 'Results' tab is selected, showing the output of the program:

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

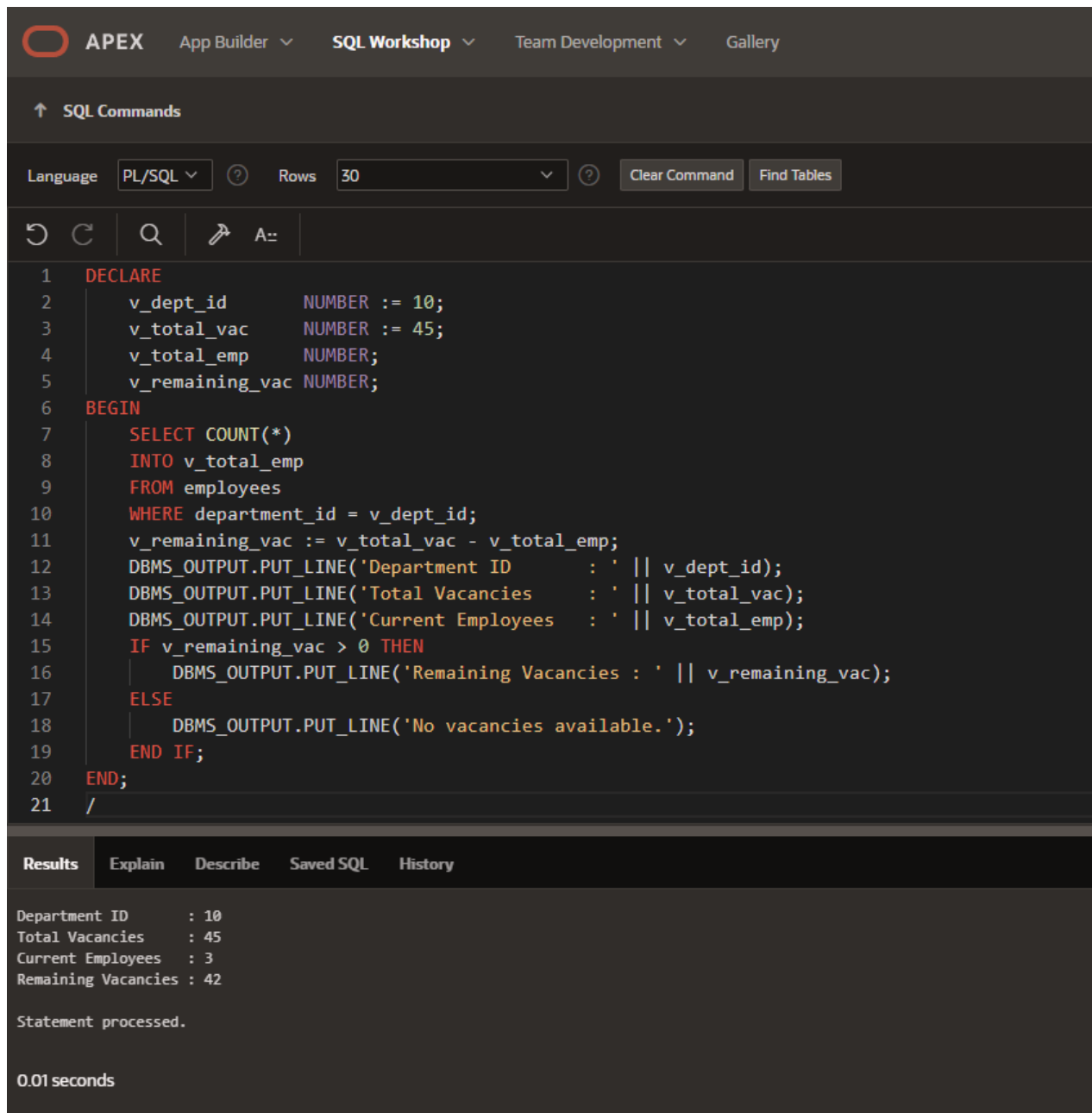
Statement processed.

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.



The screenshot displays the APEX SQL Workshop interface. The top navigation bar includes the APEX logo and links to App Builder, SQL Workshop, Team Development, and Gallery. The main area is titled "SQL Commands" and features a toolbar with options for Language (set to PL/SQL), Rows (set to 30), and buttons for Clear Command and Find Tables. Below the toolbar is a code editor with a PL/SQL program. The program declares variables for department ID, total vacancies, total employees, and remaining vacancies. It then calculates the total employees for department 10 and determines the remaining vacancies by subtracting the total employees from the total vacancies. The results are displayed using DBMS\_OUTPUT.PUT\_LINE. The bottom section shows the execution results, including the output of the program and the time taken to process the statement.

```
1 DECLARE
2     v_dept_id      NUMBER := 10;
3     v_total_vac     NUMBER := 45;
4     v_total_emp     NUMBER;
5     v_remaining_vac NUMBER;
6 BEGIN
7     SELECT COUNT(*)
8     INTO v_total_emp
9     FROM employees
10    WHERE department_id = v_dept_id;
11     v_remaining_vac := v_total_vac - v_total_emp;
12     DBMS_OUTPUT.PUT_LINE('Department ID      : ' || v_dept_id);
13     DBMS_OUTPUT.PUT_LINE('Total Vacancies      : ' || v_total_vac);
14     DBMS_OUTPUT.PUT_LINE('Current Employees    : ' || v_total_emp);
15     IF v_remaining_vac > 0 THEN
16         DBMS_OUTPUT.PUT_LINE('Remaining Vacancies : ' || v_remaining_vac);
17     ELSE
18         DBMS_OUTPUT.PUT_LINE('No vacancies available.');
```

**Results**   Explain   Describe   Saved SQL   History

```
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.

The screenshot displays the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section shows the 'Language' set to 'PL/SQL', 'Rows' set to '30', and buttons for 'Clear Command' and 'Find Tables'. The main editor contains a PL/SQL program that uses a cursor to loop through all employees, displaying their details in a formatted output.

```
1 BEGIN
2   DBMS_OUTPUT.PUT_LINE(
3     RPAD('ID', 5) ||
4     RPAD('Name', 25) ||
5     RPAD('Job Title', 35) ||
6     RPAD('Hire Date', 20) ||
7     RPAD('Salary', 10)
8   );
9   DBMS_OUTPUT.PUT_LINE(RPAD('-', 95, '-'));
10
11  FOR emp_rec IN (
12    SELECT e.employee_id,
13           e.first_name || ' ' || e.last_name AS full_name,
14           j.job_title,
15           e.hire_date,
16           e.salary
17    FROM employees e
18    JOIN jobs j ON e.job_id = j.job_id
19    ORDER BY e.employee_id
20  ) LOOP
21    DBMS_OUTPUT.PUT_LINE(
22      RPAD(emp_rec.employee_id, 5) ||
23      RPAD(emp_rec.full_name, 25) ||
24      RPAD(emp_rec.job_title, 35) ||
25      RPAD(TO_CHAR(emp_rec.hire_date, 'DD-MON-YYYY'), 20) ||
26      RPAD(emp_rec.salary, 10)
27    );
28  END LOOP;
29 END;
30 /
```

The 'Results' section shows the output of the program, displaying a table with columns: ID, Name, Job Title, Hire Date, and Salary. The data is sorted by employee ID.

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.

The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this is a 'SQL Commands' section with a 'Language' dropdown set to 'PL/SQL', a 'Rows' limit of '30', and buttons for 'Clear Command' and 'Find Tables'. The main editor displays a PL/SQL program that uses a loop to iterate through all employees, formatting their IDs, names, and department names for output. The 'Results' tab at the bottom shows the output of the program, displaying a list of employees with their IDs, names, and departments. The statement was processed successfully in 0.02 seconds.

```
1 BEGIN
2   DBMS_OUTPUT.PUT_LINE(
3     RPAD('ID', 5) ||
4     RPAD('Name', 25) ||
5     RPAD('Department', 30)
6   );
7   DBMS_OUTPUT.PUT_LINE(RPAD('-', 60, '-'));
8
9   FOR emp_rec IN (
10    SELECT e.employee_id,
11           e.first_name || ' ' || e.last_name AS full_name,
12           d.department_name
13    FROM employees e
14    JOIN departments d ON e.department_id = d.department_id
15    ORDER BY e.employee_id
16  ) LOOP
17    DBMS_OUTPUT.PUT_LINE(
18      RPAD(emp_rec.employee_id, 5) ||
19      RPAD(emp_rec.full_name, 25) ||
20      RPAD(emp_rec.department_name, 30)
21    );
22  END LOOP;
23 END;
24 /
```

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.

The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' section is active, showing a PL/SQL program. The program is as follows:

```
1 BEGIN
2     DBMS_OUTPUT.PUT_LINE(
3         RPAD('Job ID', 10) ||
4         RPAD('Job Title', 35) ||
5         RPAD('Min Salary', 15)
6     );
7     DBMS_OUTPUT.PUT_LINE(RPAD('-', 60, '-'));
8
9     FOR job_rec IN (
10        SELECT job_id,
11               job_title,
12               min_salary
13        FROM jobs
14        ORDER BY job_id
15    ) LOOP
16        DBMS_OUTPUT.PUT_LINE(
17            RPAD(job_rec.job_id, 10) ||
18            RPAD(job_rec.job_title, 35) ||
19            RPAD(job_rec.min_salary, 15)
20        );
21    END LOOP;
22 END;
23 /
```

The 'Results' tab is selected, displaying the output of the program. The output is a table with three columns: Job ID, Job Title, and Min Salary. The data is as follows:

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.

The screenshot shows the APEX SQL Workshop interface. At the top, there are navigation tabs: APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. Below the tabs, there's a section for SQL Commands with a language dropdown set to PL/SQL and a rows limit of 30. The main area contains a PL/SQL program that uses DBMS\_OUTPUT.PUT\_LINE to display employee IDs, names, and job history start dates. The program is executed, and the results are shown in a table below. The table has three columns: Employee ID, Name, and Job Start Date. The results show three employees: 104 (Bruce Ernst, 01-JAN-2005), 105 (David Austin, 01-JAN-2004), and 106 (Valli Pataballa, 01-JAN-2003). Below the table, it says 'Statement processed.' and '0.03 seconds'.

```
1 BEGIN
2     DBMS_OUTPUT.PUT_LINE(
3         RPAD('Employee ID', 12) ||
4         RPAD('Name', 25) ||
5         RPAD('Job Start Date', 20)
6     );
7     DBMS_OUTPUT.PUT_LINE(RPAD('-', 60, '-'));
8     FOR emp_rec IN (
9         SELECT e.employee_id,
10            e.first_name || ' ' || e.last_name AS full_name,
11            jh.start_date
12        FROM employees e
13        JOIN job_history jh ON e.employee_id = jh.employee_id
14        ORDER BY e.employee_id
15    ) LOOP
16        DBMS_OUTPUT.PUT_LINE(
17            RPAD(emp_rec.employee_id, 12) ||
18            RPAD(emp_rec.full_name, 25) ||
19            RPAD(TO_CHAR(emp_rec.start_date, 'DD-MON-YYYY'), 20)
20        );
21    END LOOP;
22 END;
23 /
```

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.

The screenshot shows the APEX SQL Workshop interface. At the top, there are navigation tabs: APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. Below the tabs, there's a section for SQL Commands. The Language is set to PL/SQL, and the Rows limit is 30. There are buttons for Clear Command and Find Tables. The main area contains a PL/SQL program that uses a loop to iterate over all employees and their job history, displaying their IDs, names, and job end dates in a formatted manner. The program is as follows:

```
1 BEGIN
2   DBMS_OUTPUT.PUT_LINE(
3     RPAD('Employee ID', 12) ||
4     RPAD('Name', 25) ||
5     RPAD('Job End Date', 20)
6   );
7   DBMS_OUTPUT.PUT_LINE(RPAD('-', 60, '-'));
8   FOR emp_rec IN (
9     SELECT e.employee_id,
10            e.first_name || ' ' || e.last_name AS full_name,
11            jh.end_date
12     FROM employees e
13     JOIN job_history jh ON e.employee_id = jh.employee_id
14     ORDER BY e.employee_id
15   ) LOOP
16     DBMS_OUTPUT.PUT_LINE(
17       RPAD(emp_rec.employee_id, 12) ||
18       RPAD(emp_rec.full_name, 25) ||
19       RPAD(TO_CHAR(emp_rec.end_date, 'DD-MON-YYYY'), 20)
20     );
21   END LOOP;
22 END;
23 /
```

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is active, showing the output of the program. The output is a table with three columns: Employee ID, Name, and Job End Date. The data is as follows:

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

Below the table, it says "Statement processed." and "0.01 seconds".