**TASK 1:**

Write PL/SQL blocks to show the declaration of variables. Follow the following

instructions:

• Declare variables for all attributes existing in employee table.

• Variables datatypes should be defined appropriately and use rowtype for any two

variables.

• Initialize any two variables at time of declaration.

• Assign values to remaining variables by showing the complete employee table’s

attribute data.

**SOLUTION:**

DECLARE

i integer :=0;

empno NUMBER;

ename VARCHAR2(50);

job VARCHAR2(50);

mgr NUMBER;

hiredate DATE;

sal NUMBER;

comm NUMBER;

deptno NUMBER;

emp\_row emp%ROWTYPE;

emp\_row\_2 emp%ROWTYPE;

BEGIN

FOR i in (select \* from emp)

LOOP

empno := i.empno;

ename := i.ename;

job := i.job;

mgr := i.mgr;

hiredate := i.hiredate;

sal := i.sal;

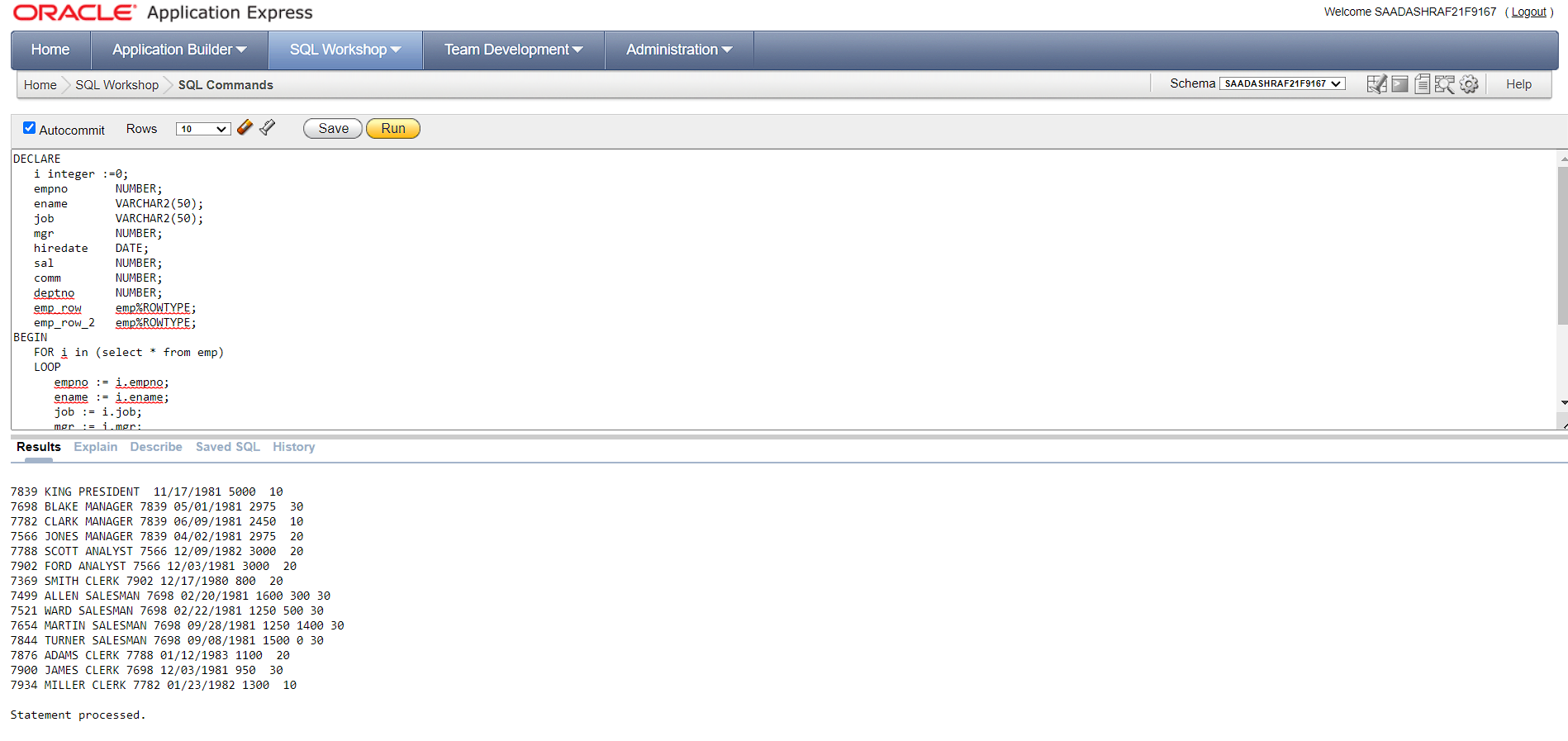
comm := i.comm;

deptno := i.deptno;

dbms\_output.put\_line(empno || ' ' || ename || ' ' || job || ' ' || mgr || ' ' || hiredate || ' ' || sal || ' ' || comm || ' ' || deptno);

END LOOP;

END;



**TASK 2:**

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

**SOLUTION:**

DECLARE

v\_salary emp.sal%TYPE;

v\_incentive NUMBER(8,2);

BEGIN

SELECT sal INTO v\_salary FROM emp WHERE empno = 7839;

IF v\_salary > 5000 THEN

v\_incentive := v\_salary \* 0.1;

ELSE

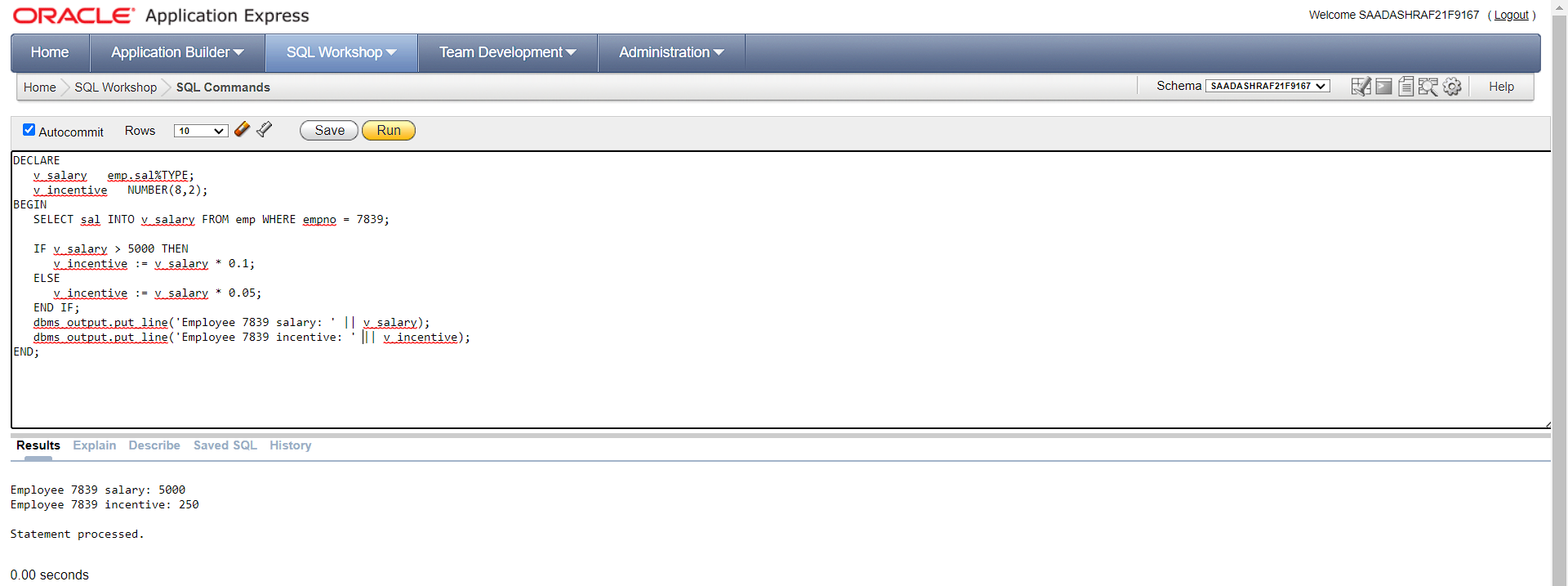
v\_incentive := v\_salary \* 0.05;

END IF;

dbms\_output.put\_line('Employee 7839 salary: ' || v\_salary);

dbms\_output.put\_line('Employee 7839 incentive: ' || v\_incentive);

END;



**TASK 3:**

Write a PL/SQL program to check whether a number is even or odd by using IF END.

**SOLUTION:**

DECLARE

num NUMBER := 10;

BEGIN

IF MOD(num, 2) = 0 THEN

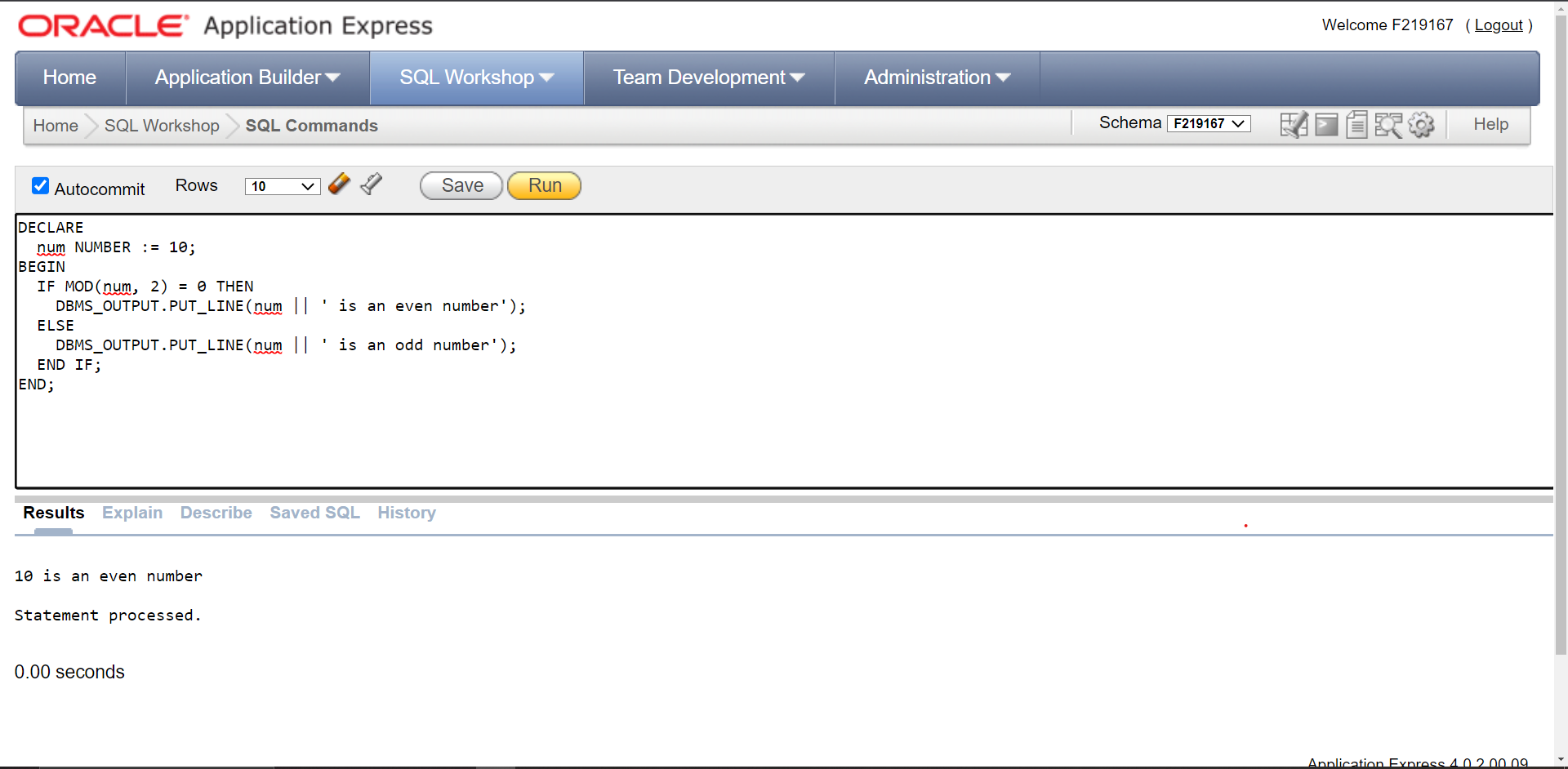
DBMS\_OUTPUT.PUT\_LINE(num || ' is an even number');

ELSE

DBMS\_OUTPUT.PUT\_LINE(num || ' is an odd number');

END IF;

END;



**TASK 4:**

Write a PL/SQL program to check whether a given number is positive, negative or zero IF

ELSE END IF.

**SOLUTION:**

DECLARE

num NUMBER := 10;

BEGIN

IF num > 0 THEN

DBMS\_OUTPUT.PUT\_LINE(num || ' is a positive number.');

ELSIF num < 0 THEN

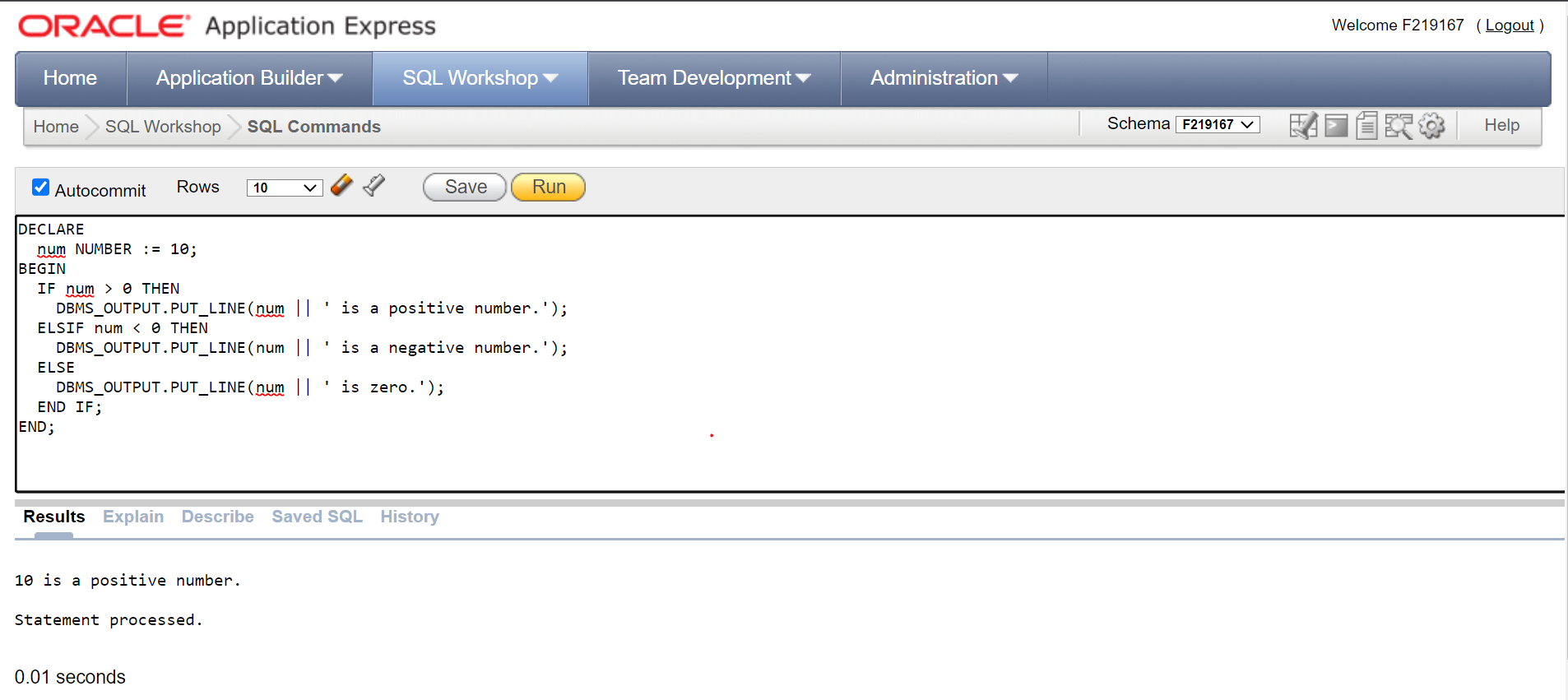
DBMS\_OUTPUT.PUT\_LINE(num || ' is a negative number.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(num || ' is zero.');

END IF;

END;



**TASK 5:**

Write a PL/SQL program to count the number of employees in department 50 and check

whether this department has any vacancies or not. There are 45 vacancies in this

department.

**SOLUTION:**

DECLARE

v\_deptno emp.deptno%TYPE := 50;

v\_num\_employees NUMBER;

v\_vacancies NUMBER := 45;

BEGIN

SELECT COUNT(\*) INTO v\_num\_employees FROM emp WHERE deptno = v\_deptno;

DBMS\_OUTPUT.PUT\_LINE('There are ' || v\_num\_employees || ' employees in department ' || v\_deptno || '.');

IF v\_num\_employees >= v\_vacancies THEN

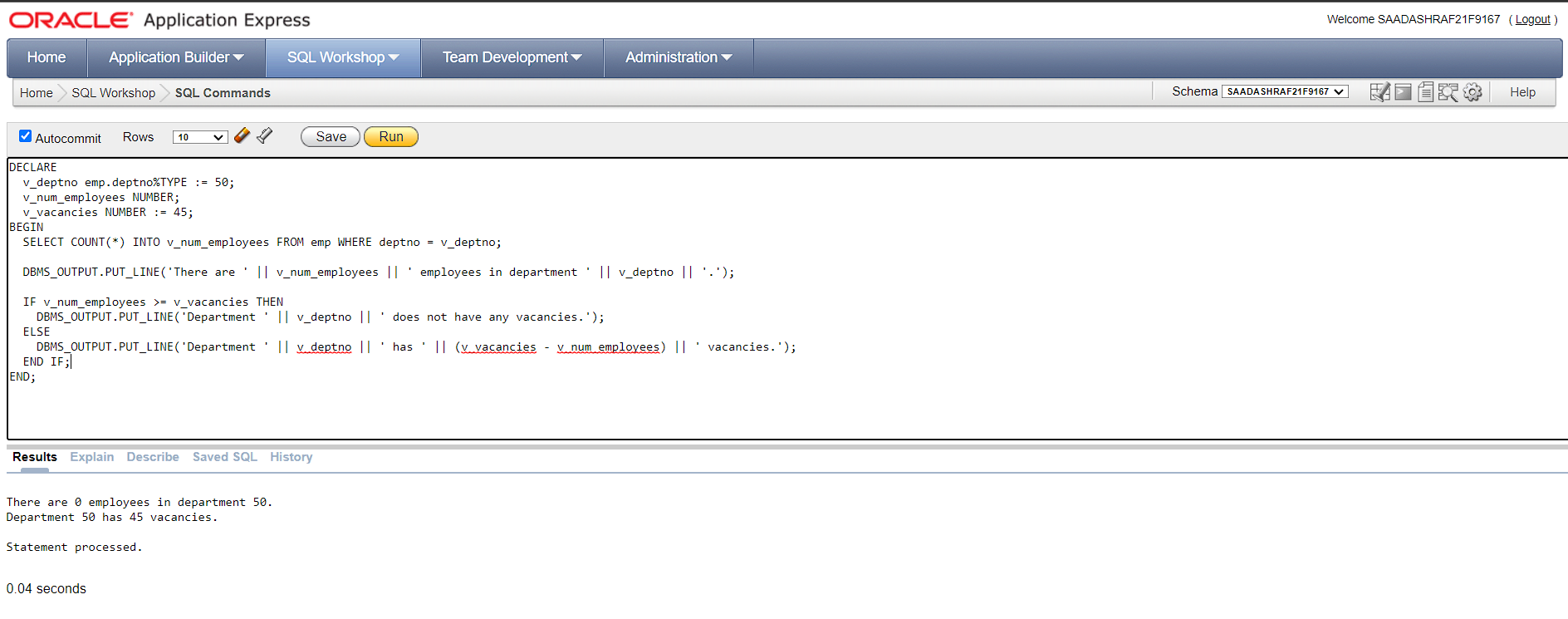
DBMS\_OUTPUT.PUT\_LINE('Department ' || v\_deptno || ' does not have any vacancies.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Department ' || v\_deptno || ' has ' || (v\_vacancies - v\_num\_employees) || ' vacancies.');

END IF;

END;

****

**TASK 6:**

Write a PL/SQL program to make a calculator (DMAS rules) using CASE statement.

**SOLUTION:**

DECLARE

v\_num1 NUMBER;

v\_num2 NUMBER;

v\_operator CHAR(1);

v\_result NUMBER;

BEGIN

v\_num1 := 5;

v\_num2 := 4;

v\_operator := '+';

CASE v\_operator

WHEN '+' THEN

v\_result := v\_num1 + v\_num2;

WHEN '-' THEN

v\_result := v\_num1 - v\_num2;

WHEN '\*' THEN

v\_result := v\_num1 \* v\_num2;

WHEN '/' THEN

v\_result := v\_num1 / v\_num2;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Invalid operator entered.');

RETURN;

END CASE;

DBMS\_OUTPUT.PUT\_LINE(v\_num1 || ' ' || v\_operator || ' ' || v\_num2 || ' = ' || v\_result);

END;



**TASK 7:**

Write a program in PL/SQL to print factorial of given number. (using for loop)

**SOLUTION:**

DECLARE

num NUMBER := 5;

factorial NUMBER := 1;

BEGIN

for i in 1..num

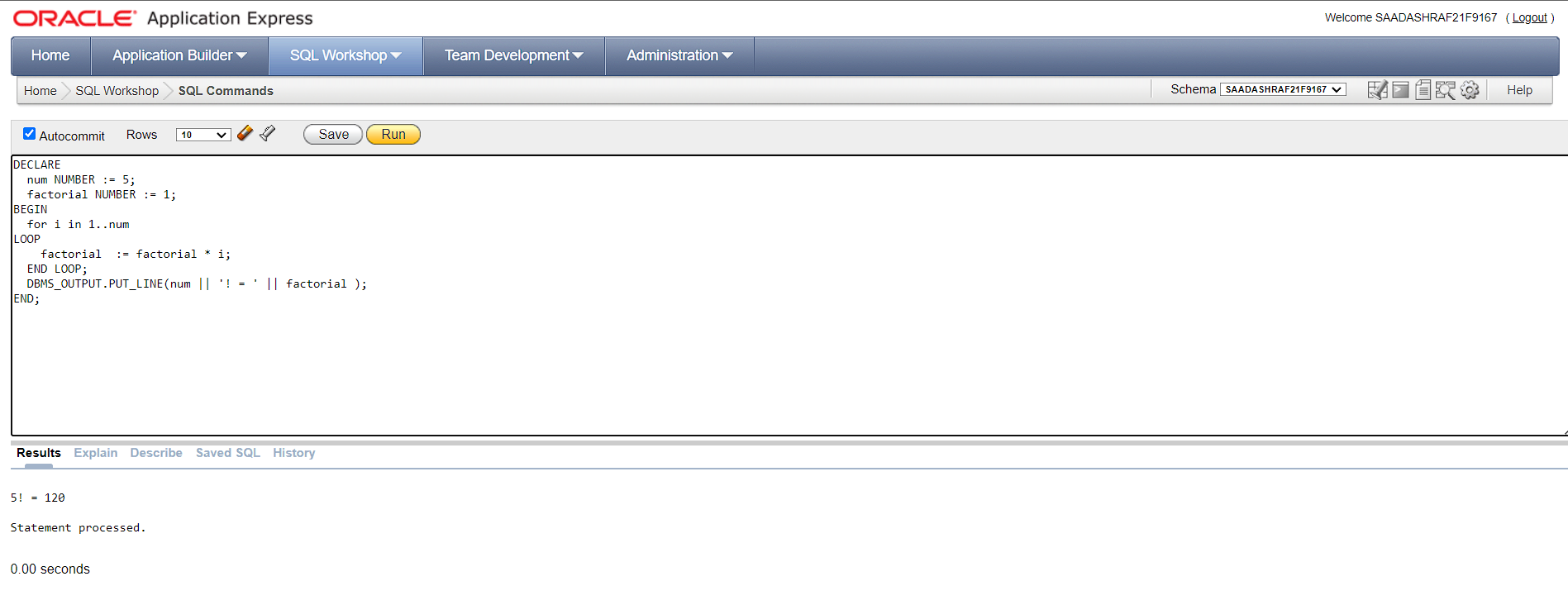
LOOP

factorial := factorial \* i;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(num || '! = ' || factorial );

END;



**TASK 8:**

Write a program in PL/SQL to update the salary of a specific employee by 8% if the salary

exceeds the mid-range of the salary against this job and update up to mid-range if the salary

is less than the mid-range of the salary, and display a suitable message.

**SOLUTION:**

DECLARE

v\_empno emp.empno%TYPE := 7698;

v\_salary emp.sal%TYPE;

v\_job emp.job%TYPE;

v\_min\_salary NUMBER;

v\_max\_salary NUMBER;

v\_mid\_salary NUMBER;

BEGIN

SELECT sal, job INTO v\_salary, v\_job FROM emp WHERE empno = v\_empno;

SELECT MIN(sal), MAX(sal) INTO v\_min\_salary, v\_max\_salary FROM emp WHERE job = v\_job;

v\_mid\_salary := (v\_min\_salary + v\_max\_salary) / 2;

IF v\_salary > v\_mid\_salary THEN

v\_salary := v\_salary \* 1.08;

IF v\_salary > v\_max\_salary THEN

v\_salary := v\_max\_salary;

END IF;

UPDATE emp SET sal = v\_salary WHERE empno = v\_empno;

DBMS\_OUTPUT.PUT\_LINE('Employee ' || v\_empno || ' salary updated to ' || v\_salary || ' (8% increase)');

ELSIF v\_salary < v\_mid\_salary THEN

v\_salary := v\_mid\_salary;

UPDATE emp SET sal = v\_salary WHERE empno = v\_empno;

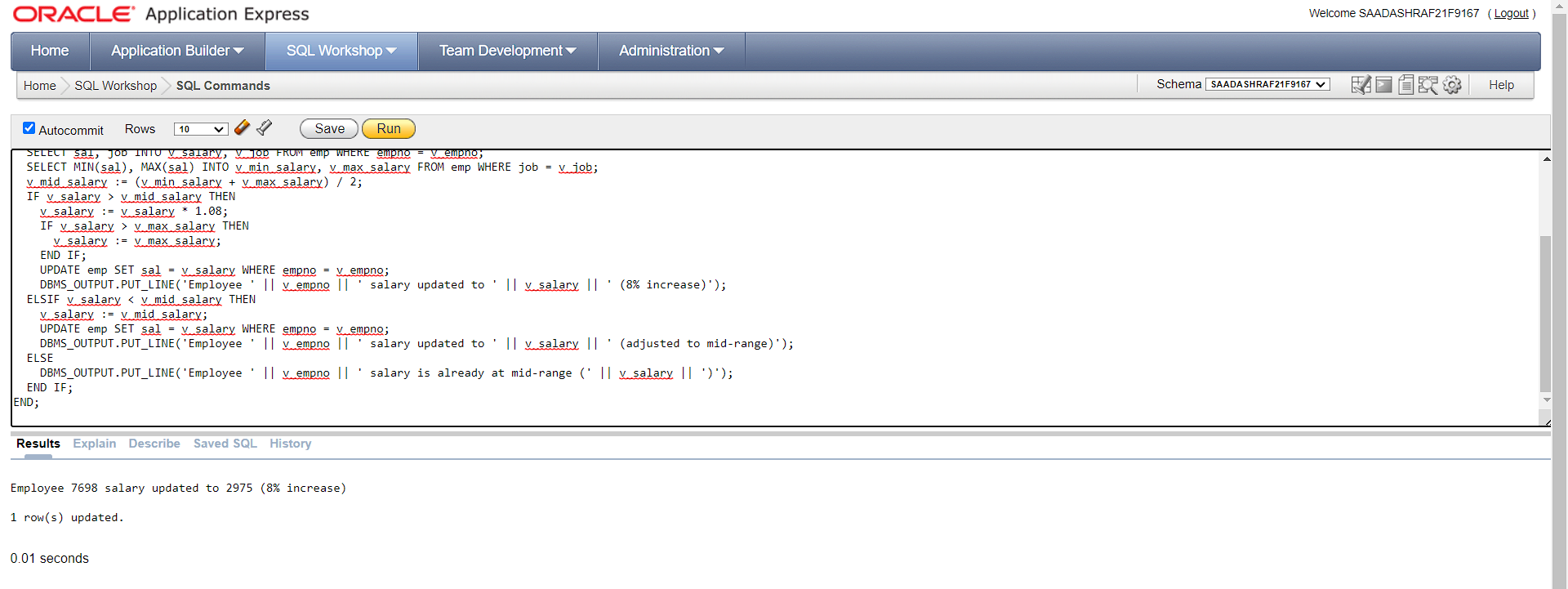
DBMS\_OUTPUT.PUT\_LINE('Employee ' || v\_empno || ' salary updated to ' || v\_salary || ' (adjusted to mid-range)');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Employee ' || v\_empno || ' salary is already at mid-range (' || v\_salary || ')');

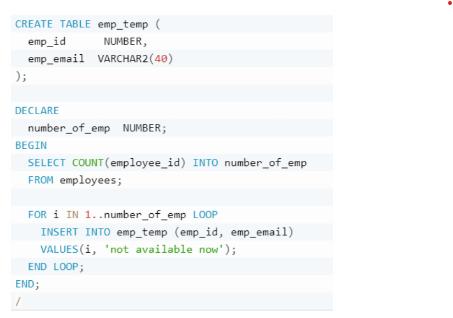
END IF;

END;



**TASK 9:**

Correct the program and find the output:



**SOLUTION:**

DECLARE

number\_of\_emp NUMBER;

BEGIN

SELECT COUNT(**emp\_id**) INTO number\_of\_emp

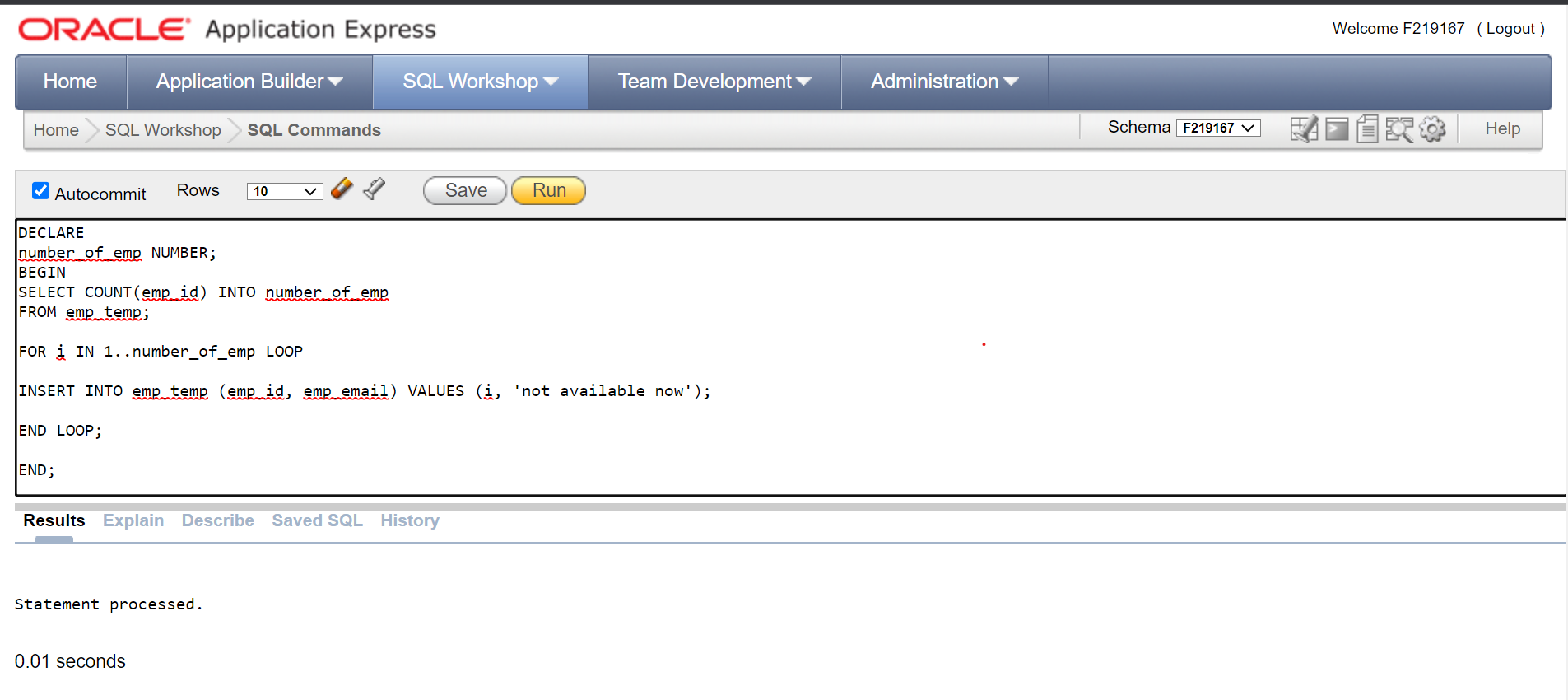
FROM **emp\_temp**;

FOR i IN 1..number\_of\_emp LOOP

INSERT INTO emp\_temp (emp\_id, emp\_email) VALUES (i, 'not available now');

END LOOP;

END;



The output of the program will depend on the number of employees in the employees table at the time it is run. For each employee in the table, a row will be inserted into the emp\_temp table with emp\_id set to a value starting at 1 and incrementing by 1 for each row, and emp\_email set to 'not available now'.