PL SQL EXERCISES

Exercise 1 – Control Structures

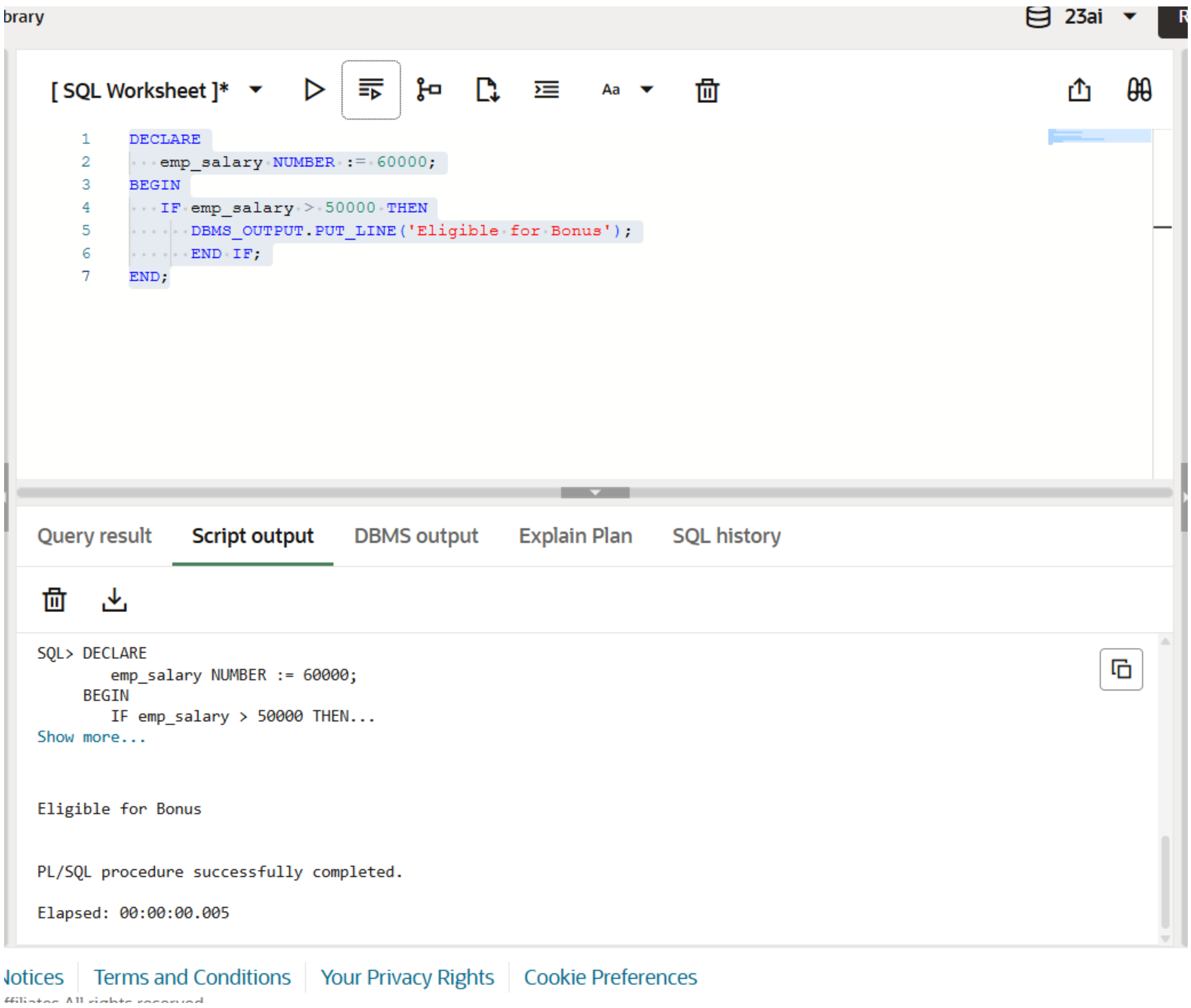
Control structures in PL/SQL are programming constructs that control the flow of execution in a PL/SQL block. They allow you to:

- Make decisions
- Perform repetitive tasks
- Control the sequence of execution

1. If Statement

```
DECLARE
  emp_salary NUMBER := 60000;
BEGIN
  IF emp_salary > 50000 THEN
    DBMS_OUTPUT.PUT_LINE('Eligible for Bonus');
  END IF;
END;
```

Output:



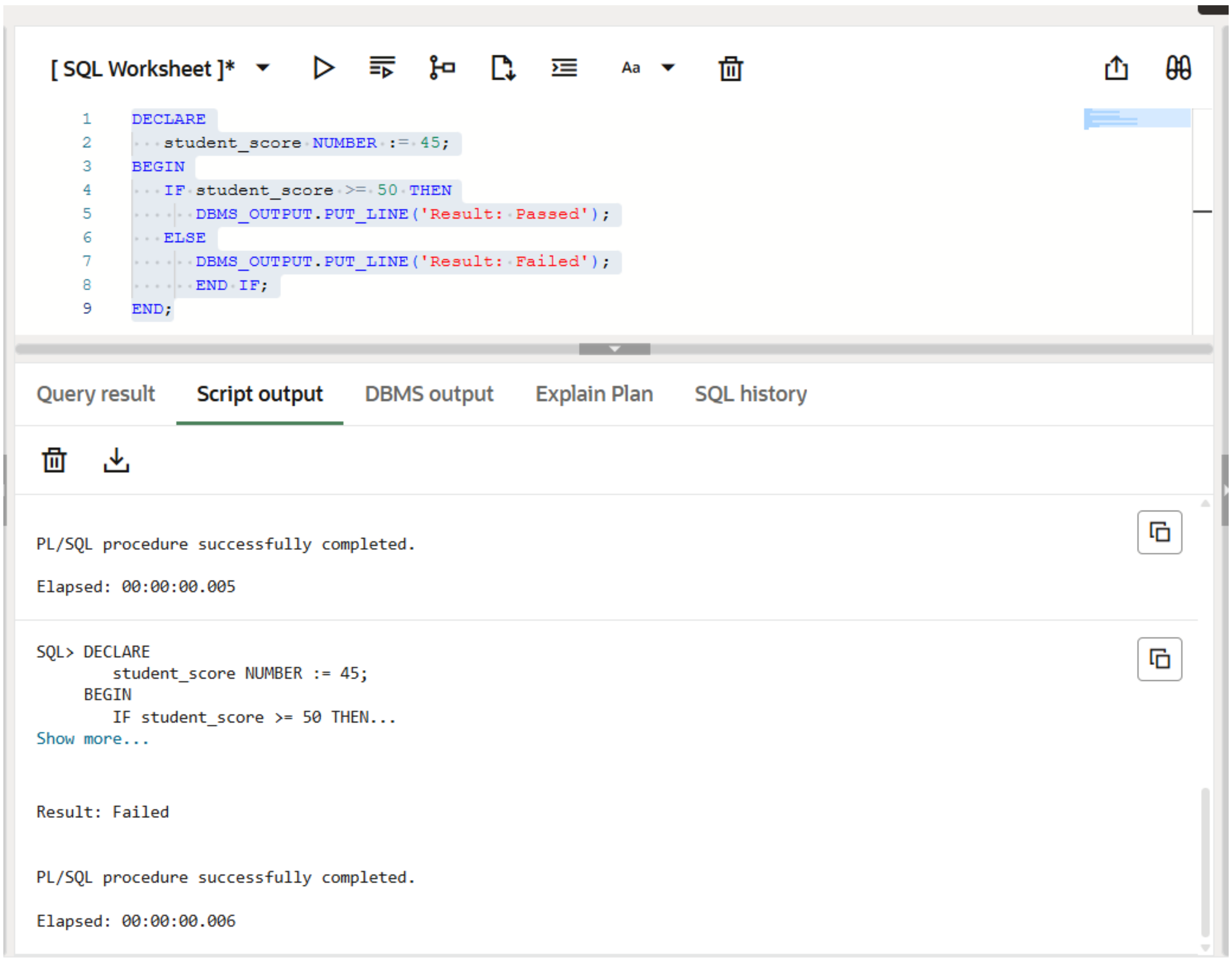
```
DECLARE
  emp_salary NUMBER := 60000;
BEGIN
  IF emp_salary > 50000 THEN
    DBMS_OUTPUT.PUT_LINE('Eligible for Bonus');
  END IF;
END;
```

```
SQL> DECLARE
        emp_salary NUMBER := 60000;
     BEGIN
        IF emp_salary > 50000 THEN...
Show more...


Eligible for Bonus


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.005
```

2. If Else

```
DECLARE
  student_score NUMBER := 45;
BEGIN
  IF student_score >= 50 THEN
    DBMS_OUTPUT.PUT_LINE('Result: Passed');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Result: Failed');
  END IF;
END;
```

Output:



```
DECLARE
   student_score NUMBER := 45;
BEGIN
   IF student_score >= 50 THEN
      DBMS_OUTPUT.PUT_LINE('Result: Passed');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Result: Failed');
      END IF;
END;
```

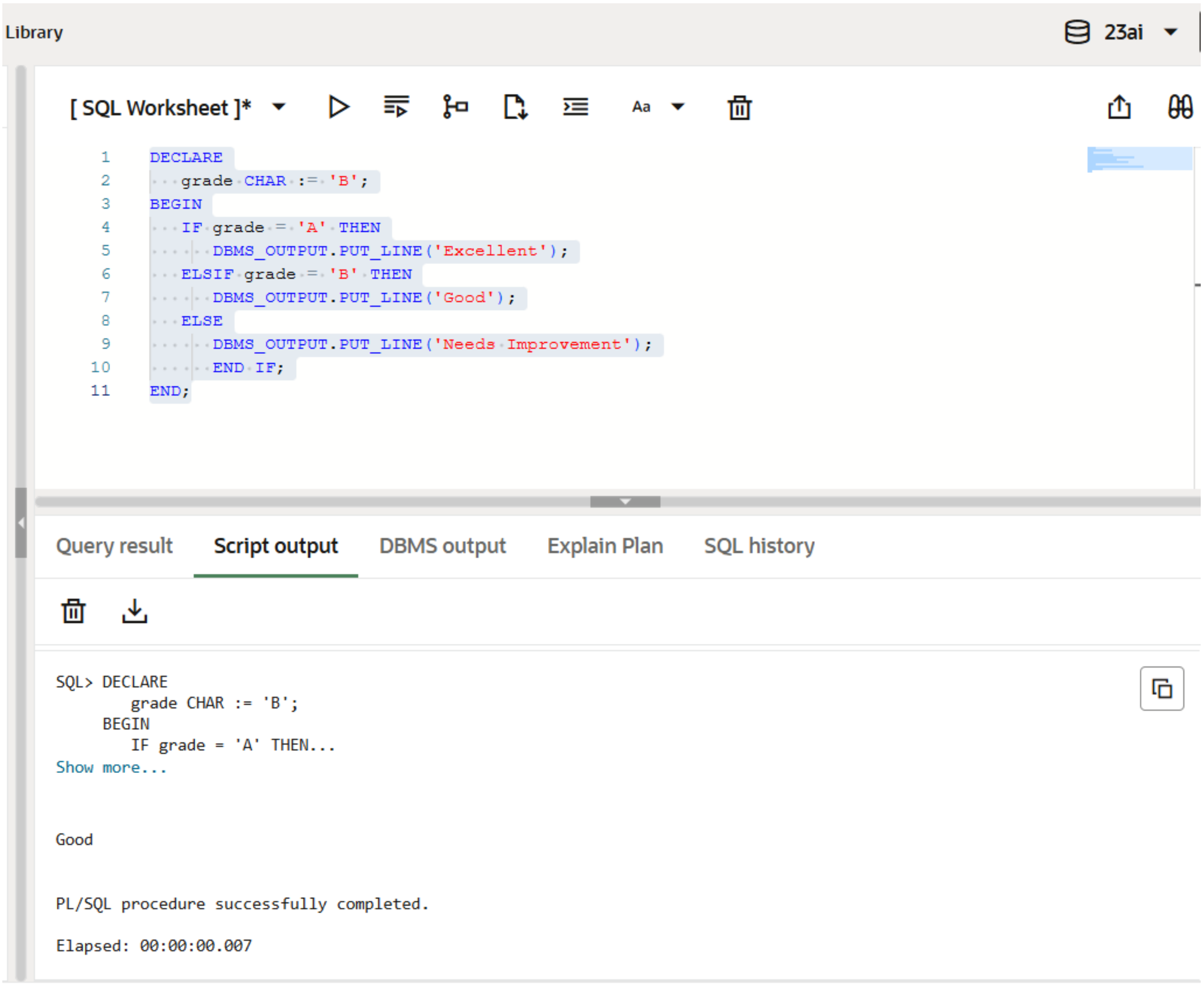
```
PL/SQL procedure successfully completed.

Elapsed: 00:00:00.005

SQL> DECLARE
        student_score NUMBER := 45;
     BEGIN
        IF student_score >= 50 THEN...
Show more...

Result: Failed

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.006
```

3. IF...ELSIF...ELSE Statement

```
DECLARE
  grade CHAR := 'B';
BEGIN
  IF grade = 'A' THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Excellent');
  ELSIF grade = 'B' THEN
    DBMS_OUTPUT.PUT_LINE('Good');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Needs Improvement');
  END IF;
END;
```

Output:



```
DECLARE
   grade CHAR := 'B';
BEGIN
   IF grade = 'A' THEN
      DBMS_OUTPUT.PUT_LINE('Excellent');
   ELSIF grade = 'B' THEN
      DBMS_OUTPUT.PUT_LINE('Good');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Needs Improvement');
      END IF;
END;
```

```
SQL> DECLARE
        grade CHAR := 'B';
     BEGIN
        IF grade = 'A' THEN...
Show more...


Good


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.007
```

4. For Loop Statement

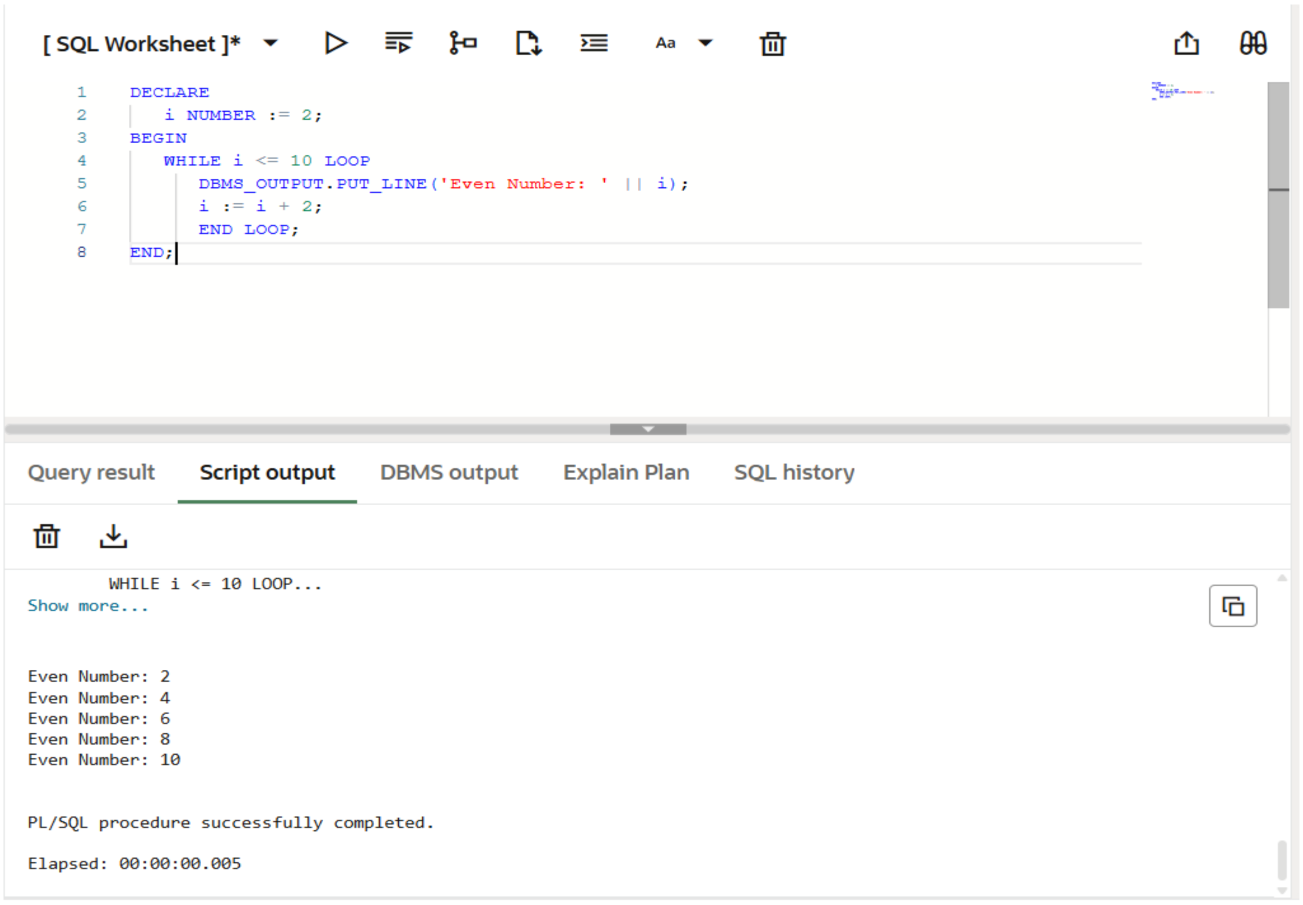
```
DECLARE
  i NUMBER := 1;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE('Count: ' || i);
    i := i + 1;
    EXIT WHEN i > 5;
  END LOOP;
END;
```

Output:

5. While Loop Statement

```
DECLARE
  i NUMBER := 2;
BEGIN
  WHILE i <= 10 LOOP
    DBMS_OUTPUT.PUT_LINE('Even Number: ' || i);
    i := i + 2;
  END LOOP;
END;
```

Output:



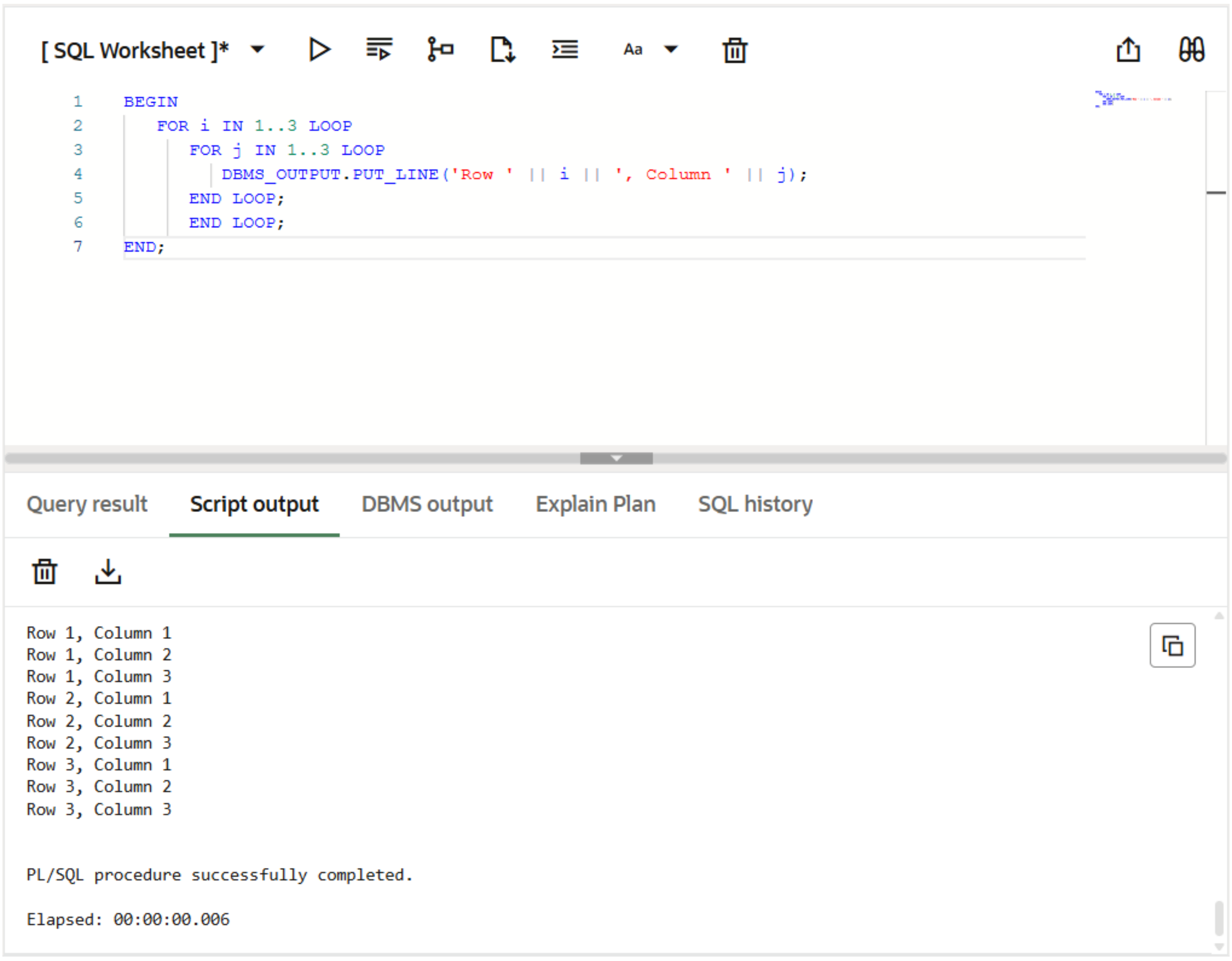
```
DECLARE
  i NUMBER := 2;
BEGIN
  WHILE i <= 10 LOOP
    DBMS_OUTPUT.PUT_LINE('Even Number: ' || i);
    i := i + 2;
    END LOOP;
END;
```

```
        WHILE i <= 10 LOOP...
Show more...


Even Number: 2
Even Number: 4
Even Number: 6
Even Number: 8
Even Number: 10


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.005
```

6. Nested Loops

```
BEGIN

  FOR i IN 1..3 LOOP

    FOR j IN 1..3 LOOP

      DBMS_OUTPUT.PUT_LINE('Row ' || i || ', Column ' || j);

    END LOOP;

  END LOOP;

END;
```

Output:



```
BEGIN
   FOR i IN 1..3 LOOP
      FOR j IN 1..3 LOOP
         DBMS_OUTPUT.PUT_LINE('Row ' || i || ', Column ' || j);
      END LOOP;
      END LOOP;
END;
```

```
Row 1, Column 1
Row 1, Column 2
Row 1, Column 3
Row 2, Column 1
Row 2, Column 2
Row 2, Column 3
Row 3, Column 1
Row 3, Column 2
Row 3, Column 3


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.006
```

STORED PROCEDURES

A stored procedure is a named block of PL/SQL code that performs a specific task and is stored in the database. It can be executed (called) multiple times with different parameters.

Key Features of Stored Procedures:

- Encapsulate business logic
- Improve code reusability
- Increase performance by reducing network traffic
- Accept input, return output, or just perform actions
- Can be called from applications, triggers, or other procedures

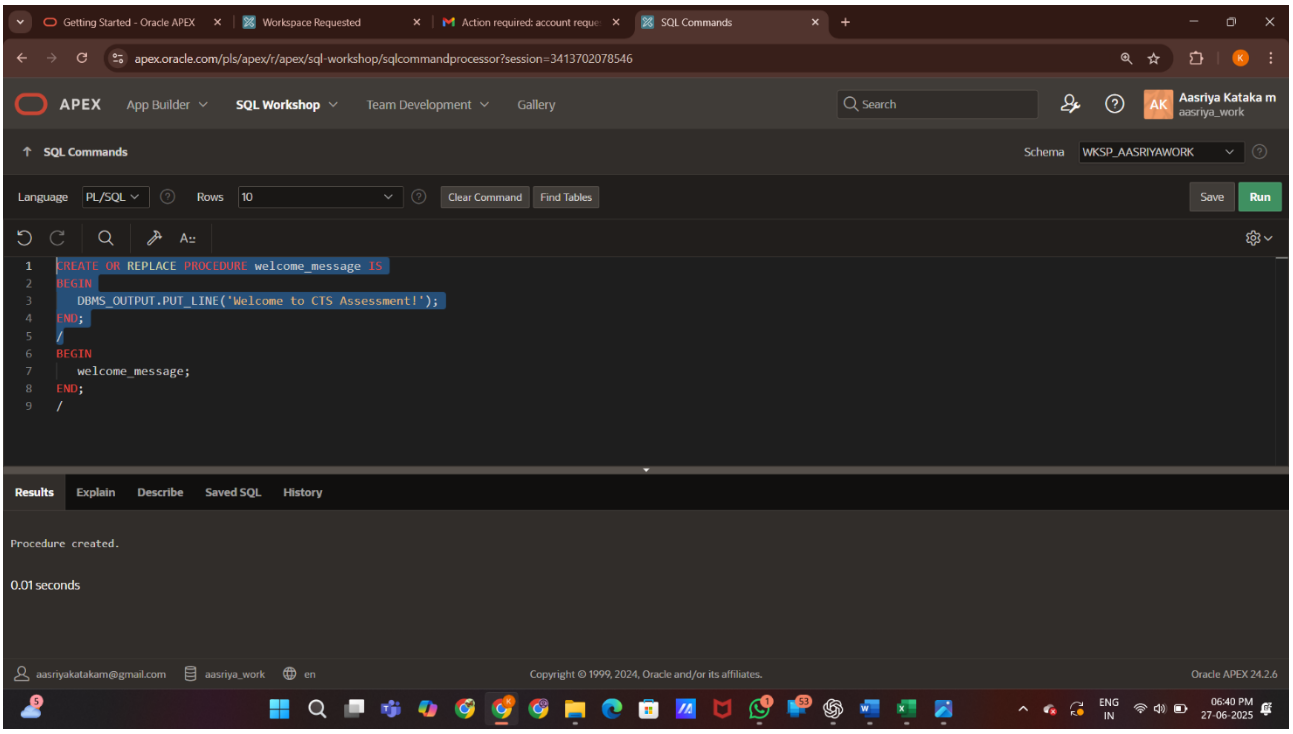
Syntax:

```
CREATE OR REPLACE PROCEDURE procedure_name (
  parameter1 IN datatype,
  parameter2 OUT datatype
) IS
BEGIN
  -- statements
END;
```

1. Stored Procedure without Parameters

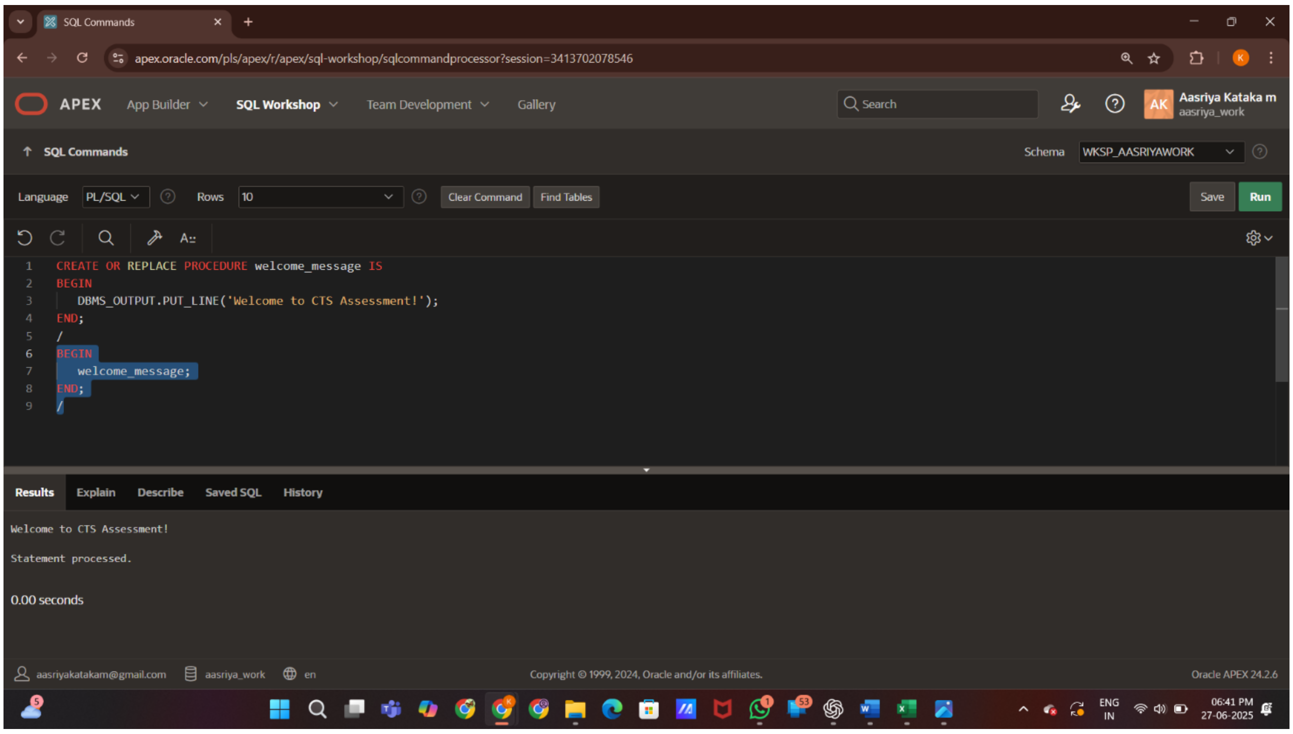
```
CREATE OR REPLACE PROCEDURE welcome_message IS
BEGIN
  DBMS_OUTPUT.PUT_LINE('Welcome to CTS Assessment!');
END;
/
BEGIN
  welcome_message;
END;
/
```

Output:



```
CREATE OR REPLACE PROCEDURE welcome_message IS
BEGIN
   DBMS_OUTPUT.PUT_LINE('Welcome to CTS Assessment!');
END;
/
BEGIN
   welcome_message;
END;
/
```

```
Procedure created.

0.01 seconds
```



```
CREATE OR REPLACE PROCEDURE welcome_message IS
BEGIN
   DBMS_OUTPUT.PUT_LINE('Welcome to CTS Assessment!');
END;
/
BEGIN
   welcome_message;
END;
/
```

```
Welcome to CTS Assessment!

Statement processed.

0.00 seconds
```

2. Stored Procedure with IN Parameter

CREATE OR REPLACE PROCEDURE greet_user(p_name IN VARCHAR2) IS

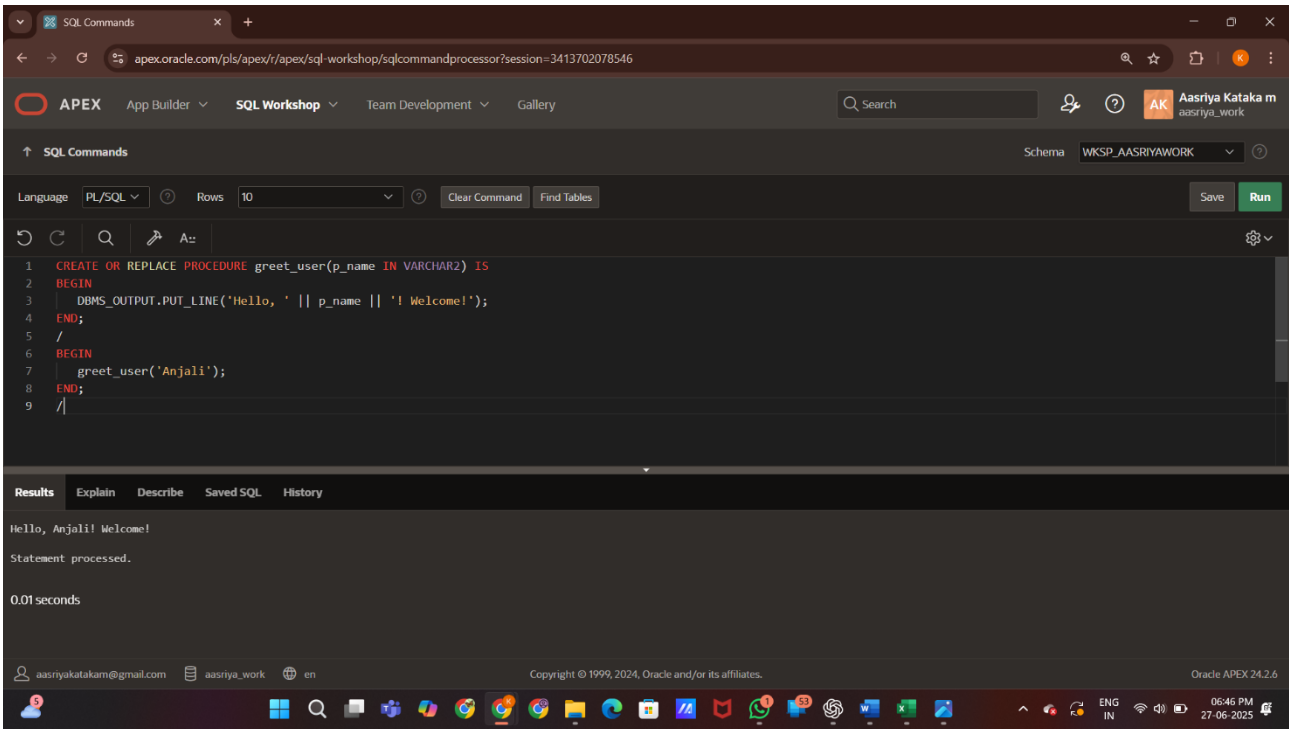
BEGIN

  DBMS_OUTPUT.PUT_LINE('Hello, ' || p_name || '! Welcome!');

END;

```
/
BEGIN
   greet_user('Anjali');
END;
/
```

Output:



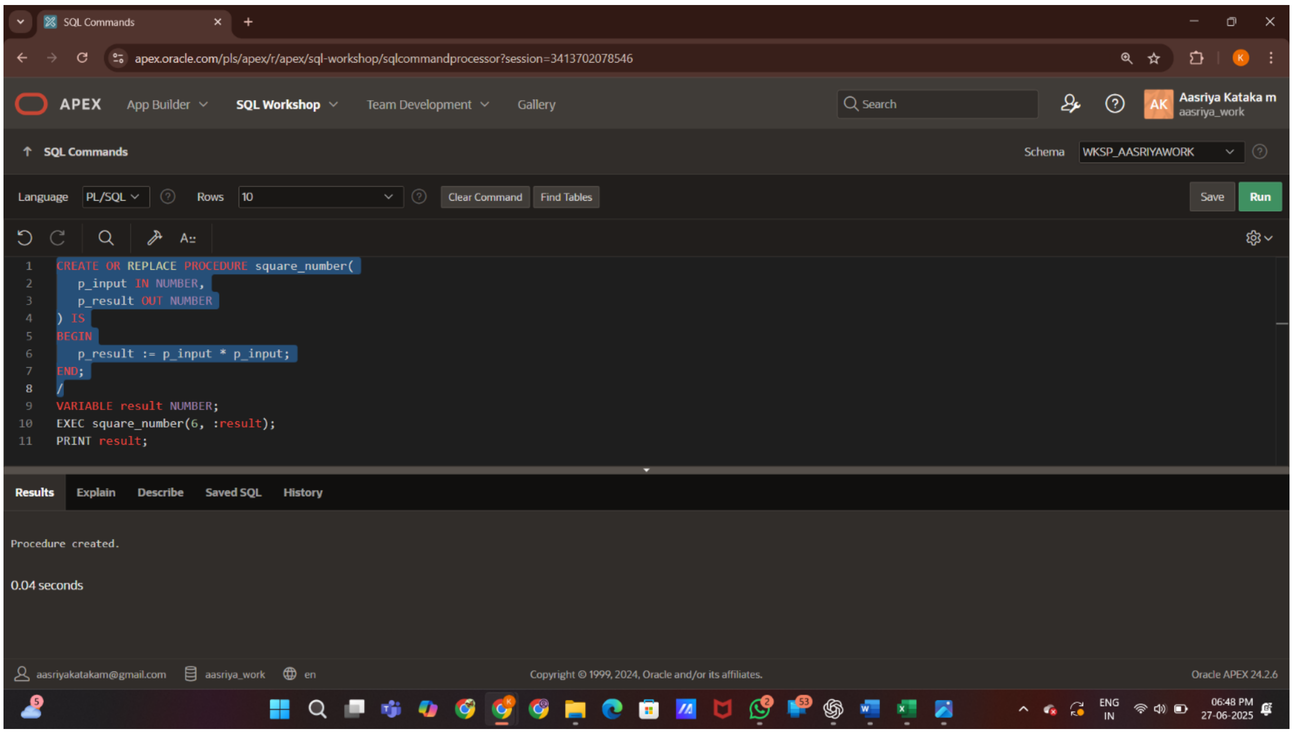
```
CREATE OR REPLACE PROCEDURE greet_user(p_name IN VARCHAR2) IS
BEGIN
   DBMS_OUTPUT.PUT_LINE('Hello, ' || p_name || '! Welcome!');
END;
/
BEGIN
   greet_user('Anjali');
END;
/
```

```
Hello, Anjali! Welcome!

Statement processed.

0.01 seconds
```

3. Stored procedure with OUT parameter

```
CREATE OR REPLACE PROCEDURE square_number(
   p_input IN NUMBER,
   p_result OUT NUMBER
) IS
BEGIN
   p_result := p_input * p_input;
END;
/
VARIABLE result NUMBER;
EXEC square_number(6, :result);
PRINT result;
```

Output:



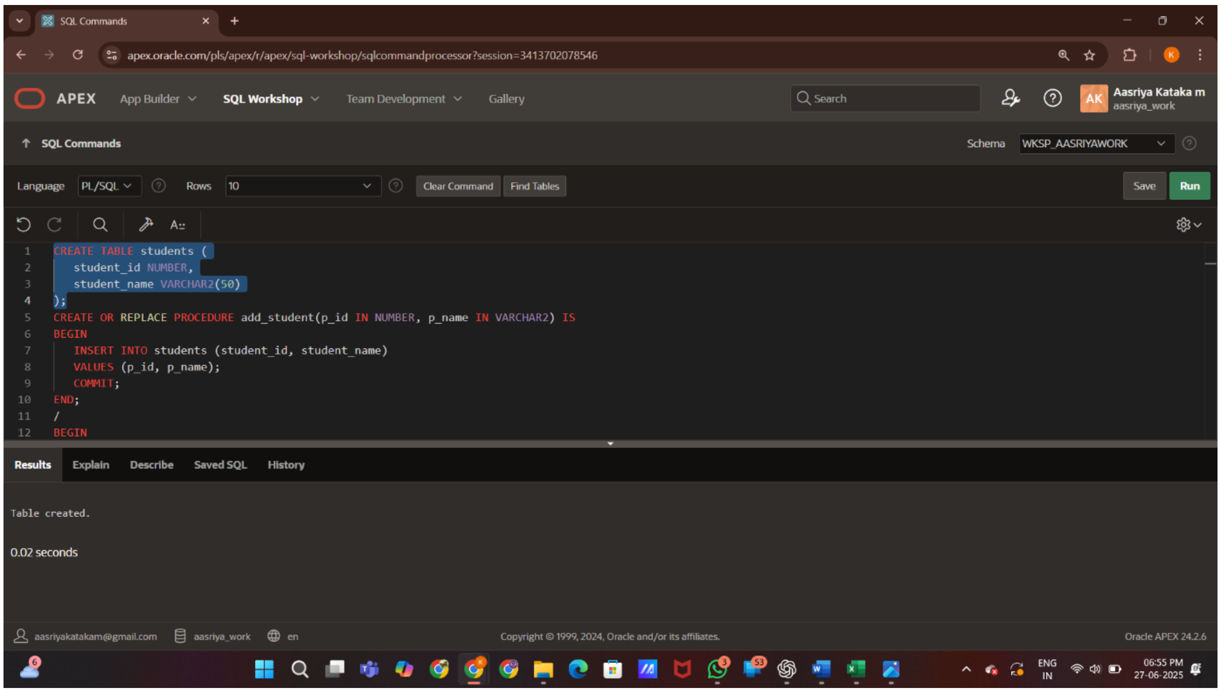
```
CREATE OR REPLACE PROCEDURE square_number(
    p_input IN NUMBER,
    p_result OUT NUMBER
) IS
BEGIN
    p_result := p_input * p_input;
END;
/
VARIABLE result NUMBER;
EXEC square_number(6, :result);
PRINT result;
```

```
Procedure created.

0.04 seconds
```

4. Procedure to Insert into a Table

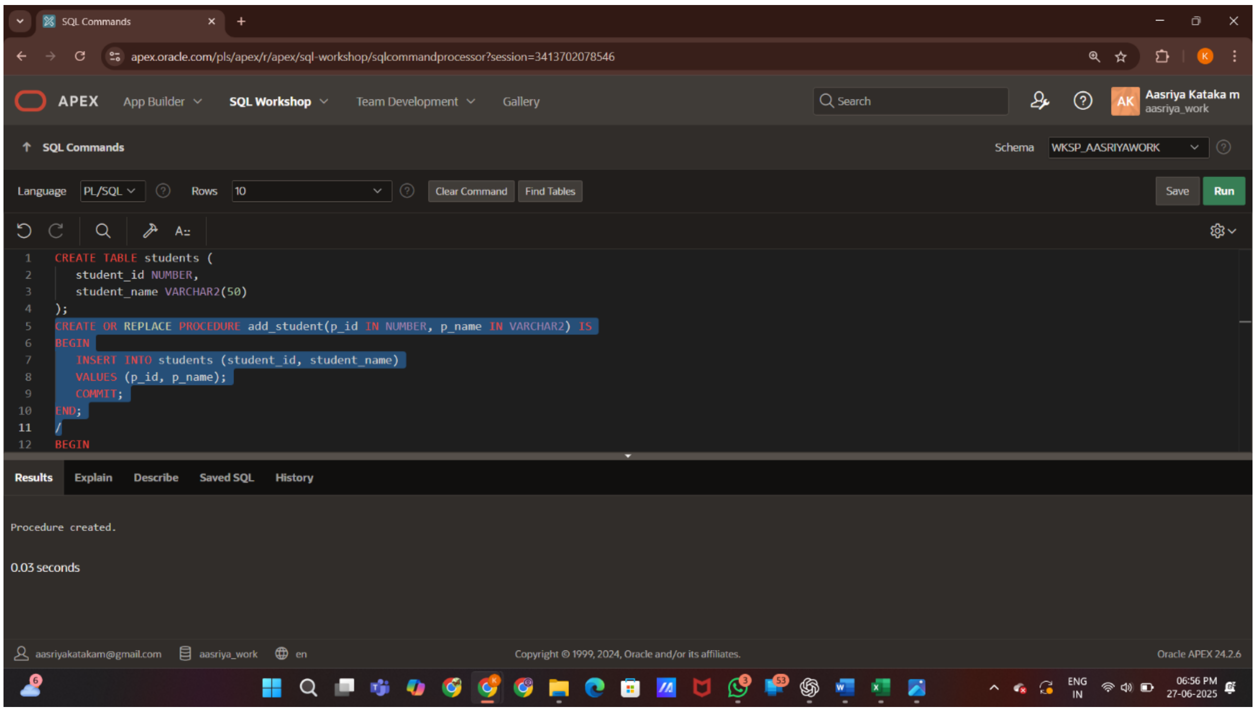
```
CREATE TABLE students (
  student_id NUMBER,
  student_name VARCHAR2(50)
);
CREATE OR REPLACE PROCEDURE add_student(p_id IN NUMBER, p_name IN
VARCHAR2) IS
BEGIN
  INSERT INTO students (student_id, student_name)
  VALUES (p_id, p_name);
  COMMIT;
END;
/
BEGIN
  add_student(1, 'Aarav');
  add_student(2,'Aarya');
  add_student(3,'Sashi');
  add_student(4,'Bunny');
END;
/
SELECT * FROM students;
```

Output:



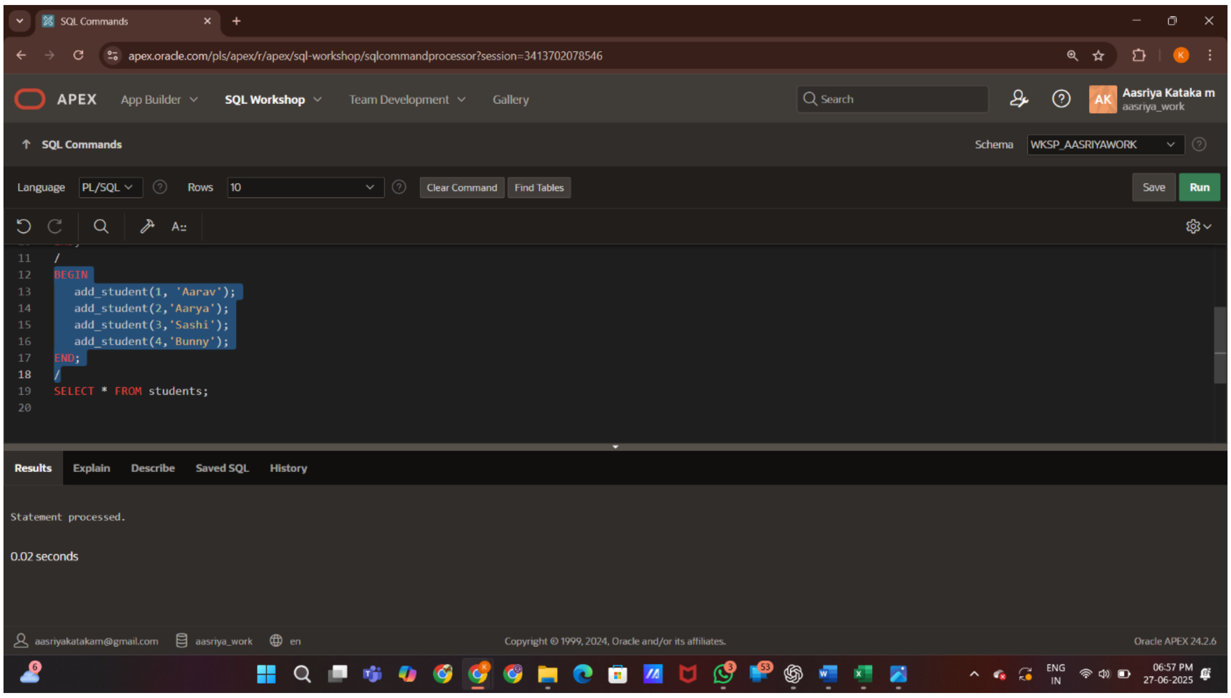
```
CREATE TABLE students (
    student_id NUMBER,
    student_name VARCHAR2(50)
);
CREATE OR REPLACE PROCEDURE add_student(p_id IN NUMBER, p_name IN VARCHAR2) IS
BEGIN
    INSERT INTO students (student_id, student_name)
    VALUES (p_id, p_name);
    COMMIT;
END;
/
BEGIN
```

```
Table created.

0.02 seconds
```



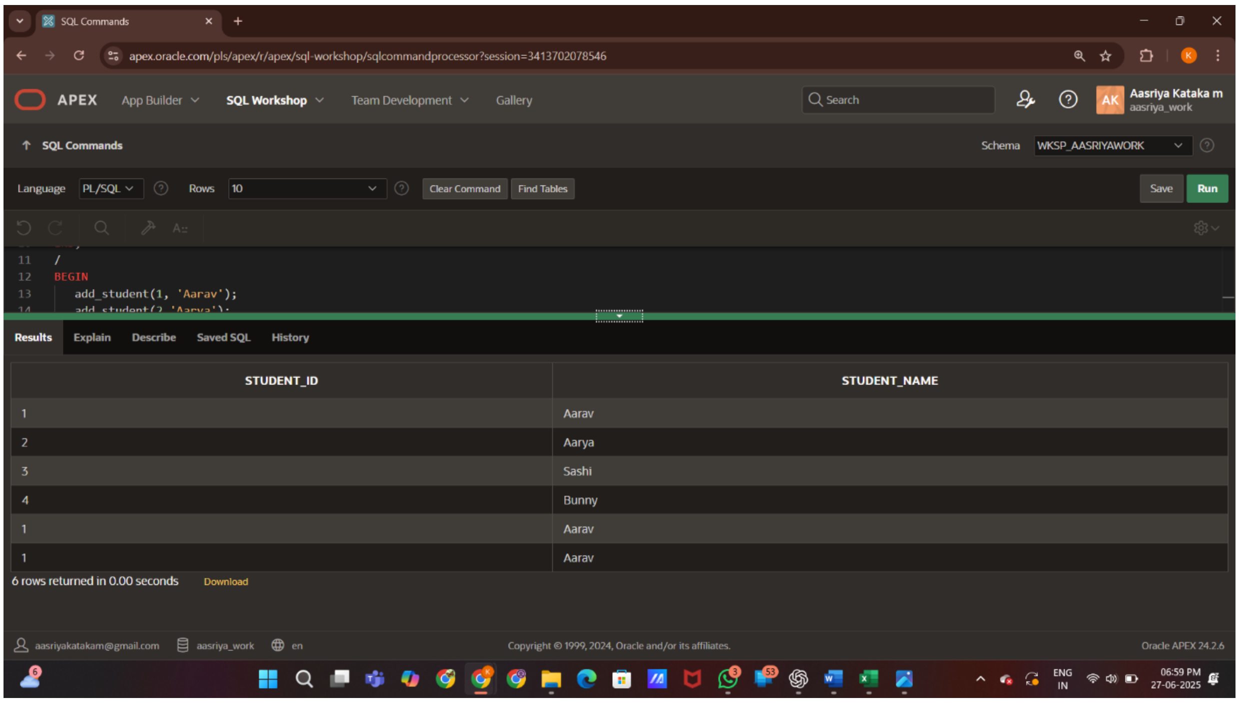
```
CREATE TABLE students (
    student_id NUMBER,
    student_name VARCHAR2(50)
);
CREATE OR REPLACE PROCEDURE add_student(p_id IN NUMBER, p_name IN VARCHAR2) IS
BEGIN
    INSERT INTO students (student_id, student_name)
    VALUES (p_id, p_name);
    COMMIT;
END;
/
BEGIN
```

```
Procedure created.

0.03 seconds
```



```
/
BEGIN
   add_student(1, 'Aarav');
   add_student(2,'Aarya');
   add_student(3,'Sashi');
   add_student(4,'Bunny');
END;
/
SELECT * FROM students;
```

```
Statement processed.

0.02 seconds
```



```
/
BEGIN
   add_student(1, 'Aarav');
```

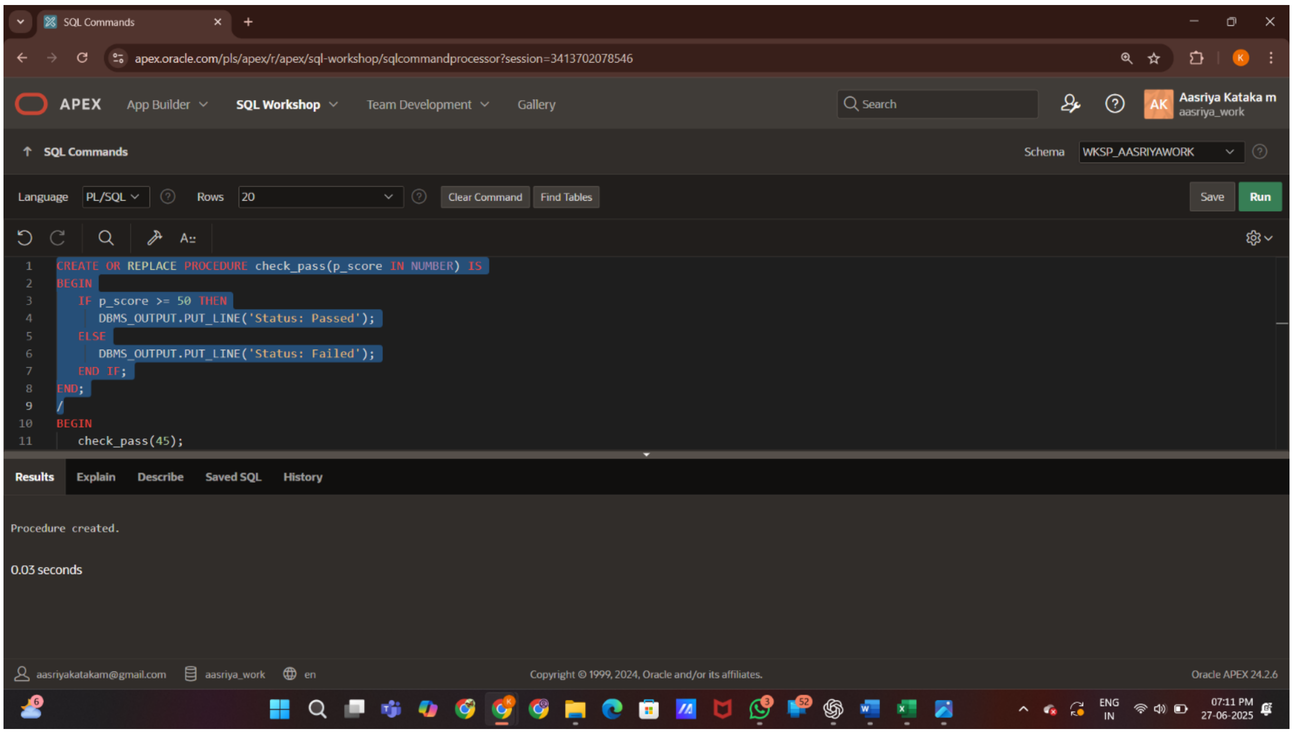
STUDENT_ID	STUDENT_NAME
1	Aarav
2	Aarya
3	Sashi
4	Bunny
1	Aarav
1	Aarav

6 rows returned in 0.00 seconds

5. Procedure with IF/ELSE Logic

```
CREATE OR REPLACE PROCEDURE check_pass(p_score IN NUMBER) IS

BEGIN

   IF p_score >= 50 THEN
      DBMS_OUTPUT.PUT_LINE('Status: Passed');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Status: Failed');
```

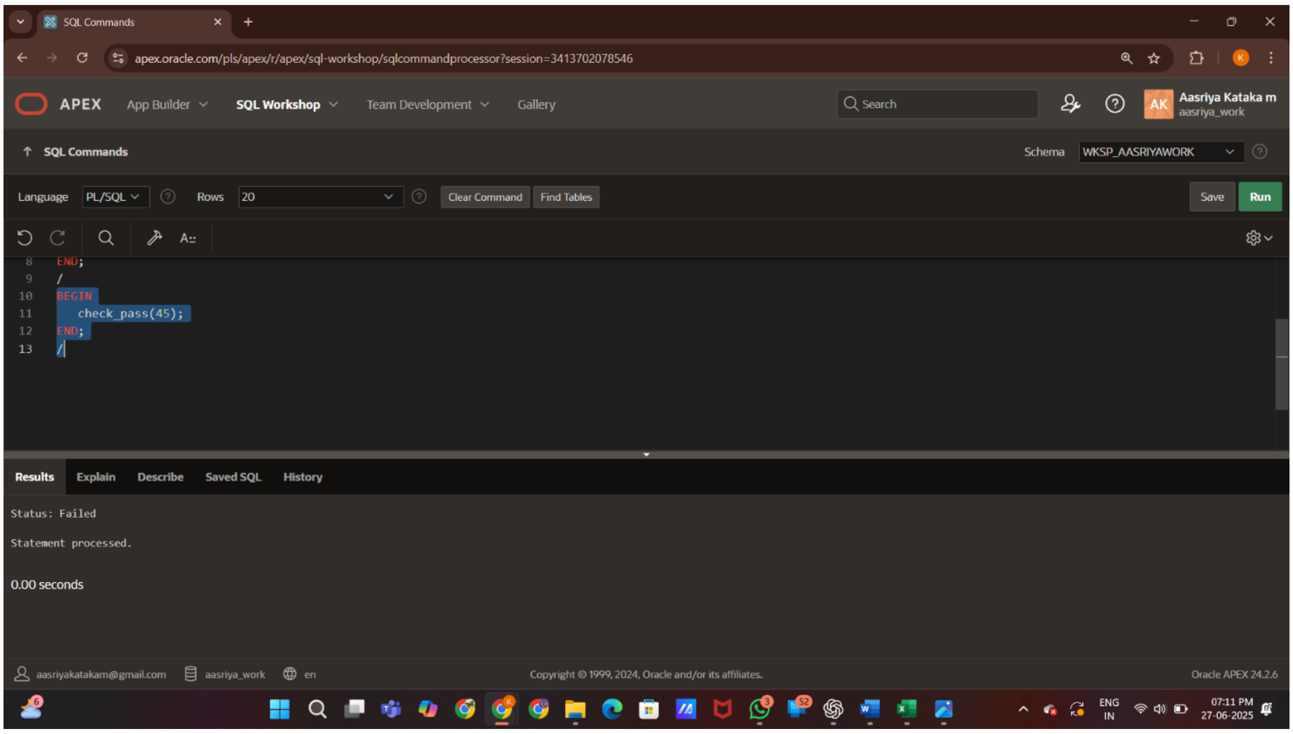
```
  END IF;
END;
/
BEGIN
  check_pass(45);
END;
/
```



```
CREATE OR REPLACE PROCEDURE check_pass(p_score IN NUMBER) IS
BEGIN
   IF p_score >= 50 THEN
      DBMS_OUTPUT.PUT_LINE('Status: Passed');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Status: Failed');
   END IF;
END;
/
BEGIN
   check_pass(45);
```

```
Procedure created.

0.03 seconds
```

Output:



```
END;
/
BEGIN
   check_pass(45);
END;
/
```

```
Status: Failed

Statement processed.

0.00 seconds
```