

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:

The screenshot displays an SQL IDE interface. The top pane, titled '[SQL Worksheet]*', contains the following PL/SQL code:

```
1 DECLARE
2   emp_salary NUMBER := 60000;
3 BEGIN
4   IF emp_salary > 50000 THEN
5     DBMS_OUTPUT.PUT_LINE('Eligible for Bonus');
6   END IF;
7 END;
```

The bottom pane shows the execution results. It includes a 'Query result' tab, a 'Script output' tab (which is active), a 'DBMS output' tab, an 'Explain Plan' tab, and an 'SQL history' tab. The 'Script output' tab displays the following text:

```
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');

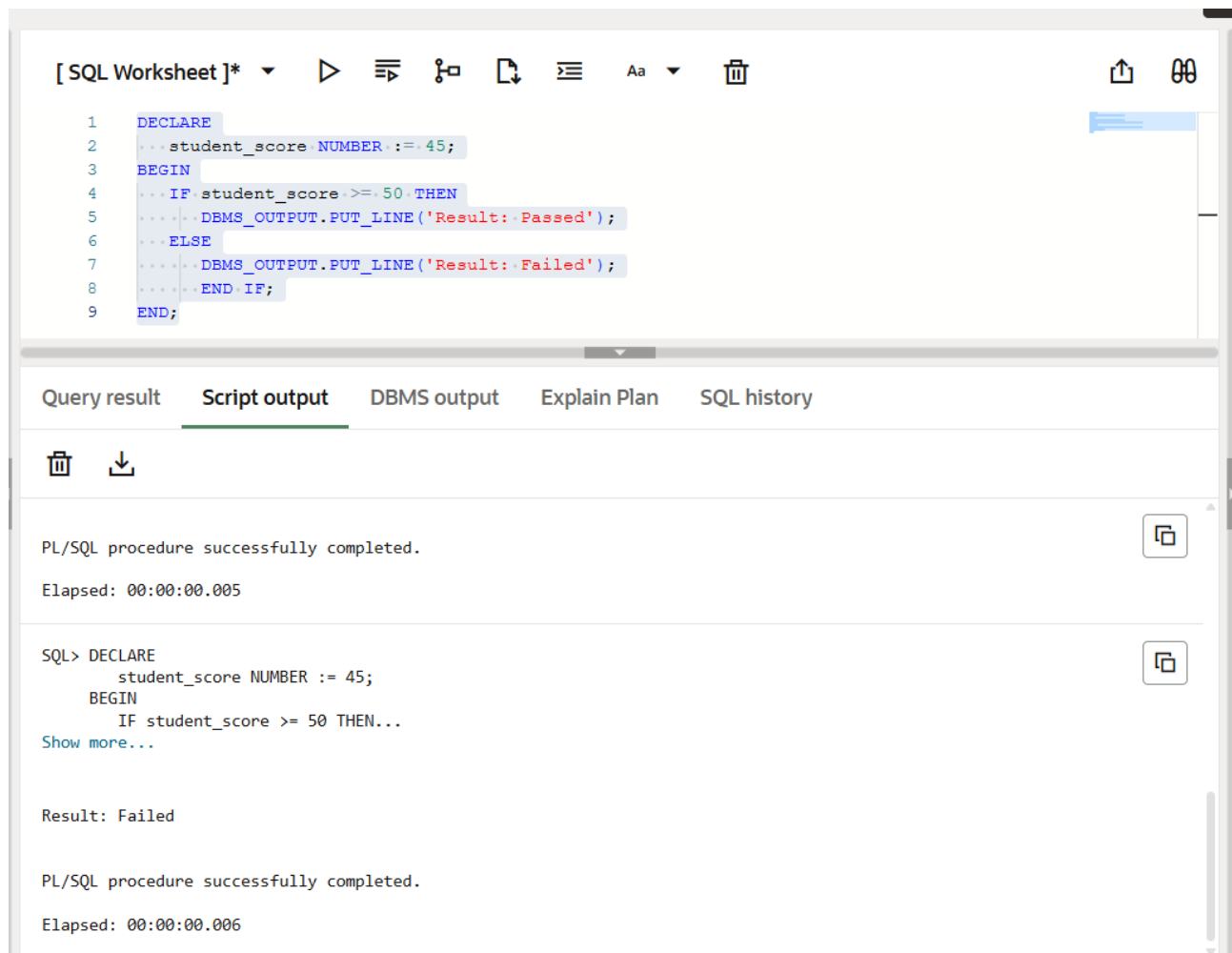
ELSE

DBMS_OUTPUT.PUT_LINE('Result: Failed');

END IF;

END;

Output:



The screenshot displays a SQL Worksheet interface. The top section shows the PL/SQL code being executed, which declares a variable 'student_score' with a value of 45 and uses an IF-ELSE statement to check if the score is greater than or equal to 50. Since the score is 45, the output is 'Result: Failed'. The bottom section shows the execution results, including the SQL prompt, the code snippet, and the output 'Result: Failed'. The interface also shows the execution time as 00:00:00.005 and 00:00:00.006.

```
[ SQL Worksheet ]*  ▶  ⌵  🔗  📄  ⌵  Aa  🗑️  ⬆️  ⌵
```

```
1 DECLARE
2   student_score NUMBER := 45;
3 BEGIN
4   IF student_score >= 50 THEN
5     DBMS_OUTPUT.PUT_LINE('Result: Passed');
6   ELSE
7     DBMS_OUTPUT.PUT_LINE('Result: Failed');
8   END IF;
9 END;
```

Query result Script output DBMS output Explain Plan SQL history

🗑️ ⬆️

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

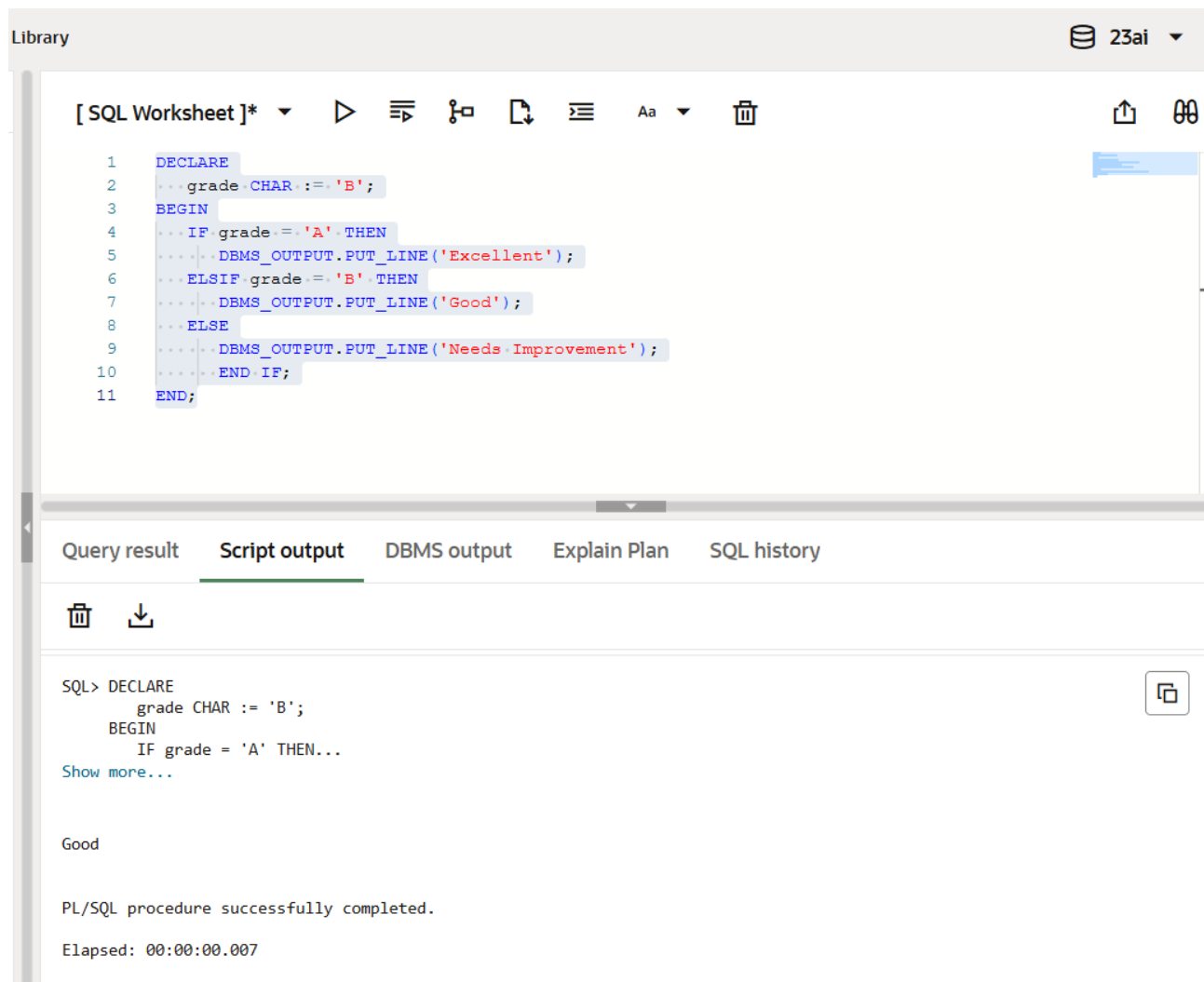
IF grade = 'A' THEN

```

    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:



The screenshot shows an SQL IDE interface. At the top, there's a 'Library' tab and a user '23ai'. Below that is a toolbar with various icons. The main editor area contains a PL/SQL script with line numbers 1 through 11. The script declares a variable 'grade' of type 'CHAR' and sets it to 'B'. It then uses an IF-ELSIF-ELSE structure to output 'Excellent', 'Good', or 'Needs Improvement' based on the value of 'grade'. The output pane at the bottom shows the execution results, including the SQL script, the output 'Good', and a message indicating the procedure completed successfully.

```

1  DECLARE
2  .. grade CHAR := 'B';
3  BEGIN
4  .. IF grade = 'A' THEN
5  .... DBMS_OUTPUT.PUT_LINE('Excellent');
6  .. ELSIF grade = 'B' THEN
7  .... DBMS_OUTPUT.PUT_LINE('Good');
8  .. ELSE
9  .... DBMS_OUTPUT.PUT_LINE('Needs Improvement');
10 .. END IF;
11 END;

```

Query result **Script output** DBMS output Explain Plan SQL history

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:

The screenshot shows an SQL IDE interface. The top toolbar includes icons for running, saving, and other functions. The main editor displays the following PL/SQL code:

```
1 DECLARE  
2     i NUMBER := 2;  
3 BEGIN  
4     WHILE i <= 10 LOOP  
5         DBMS_OUTPUT.PUT_LINE('Even Number: ' || i);  
6         i := i + 2;  
7     END LOOP;  
8 END;
```

Below the editor, the 'Script output' tab is active, showing the execution results:

```
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:

The screenshot displays an SQL IDE interface. The top toolbar includes icons for running, saving, and other standard IDE functions. The main editor window contains the following SQL script:

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

Below the editor, the 'Script output' tab is active, showing the results of the script execution:

```
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
```

At the bottom of the output window, a status message reads: "PL/SQL procedure successfully completed." followed by the execution time: "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

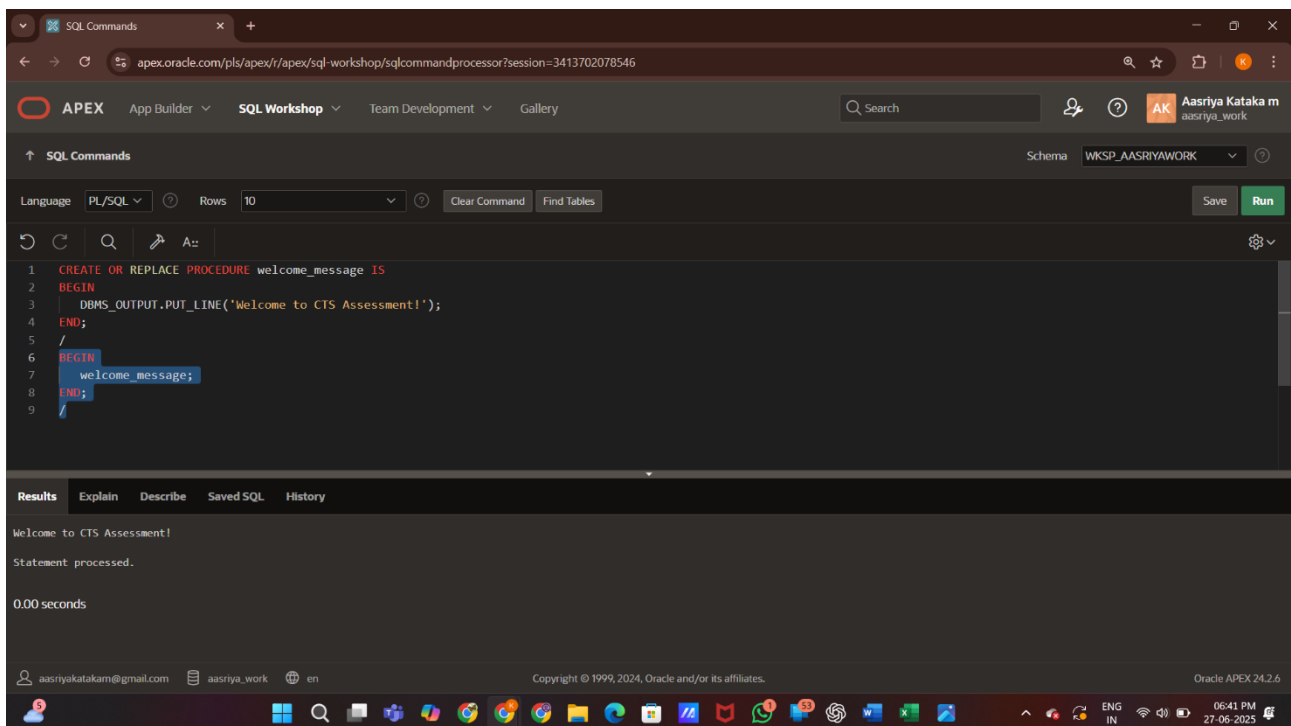
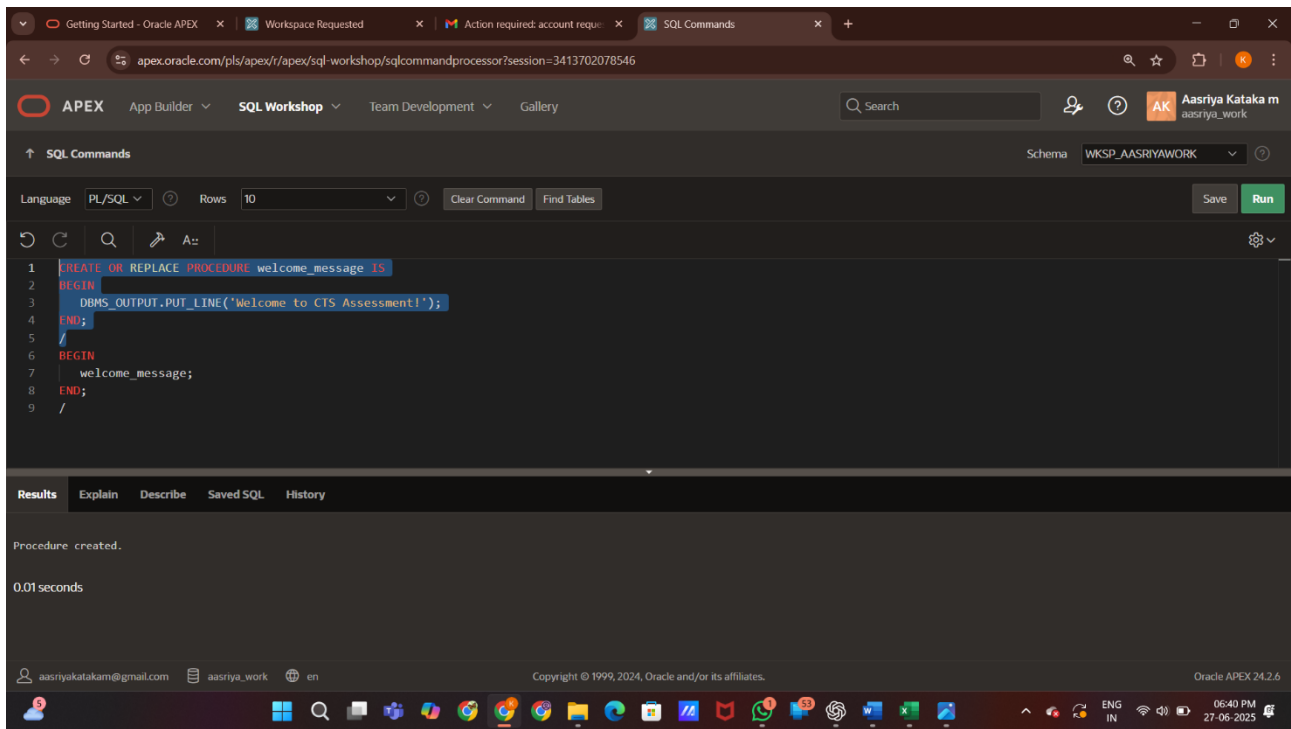
Syntax:

```
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:



2. Stored Procedure with IN Parameter

CREATE OR REPLACE PROCEDURE greet_user(p_name IN VARCHAR2) IS

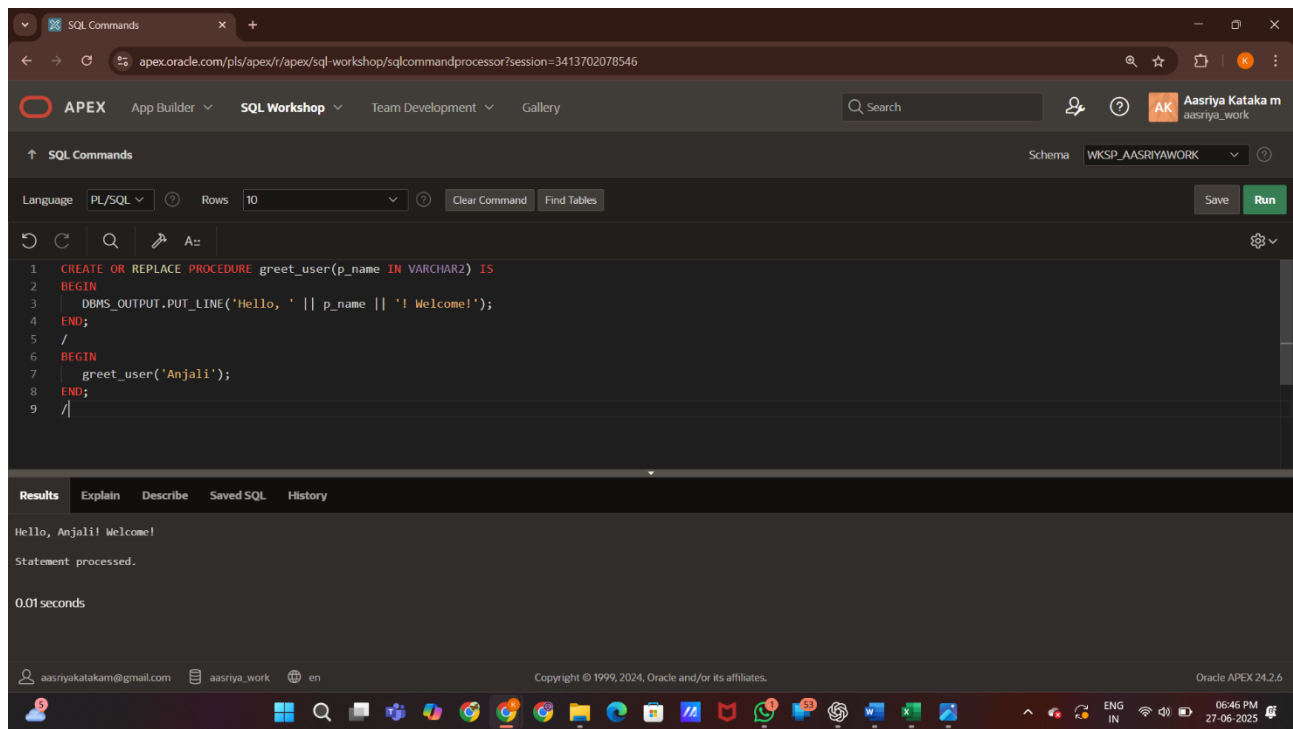
BEGIN

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

END;

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

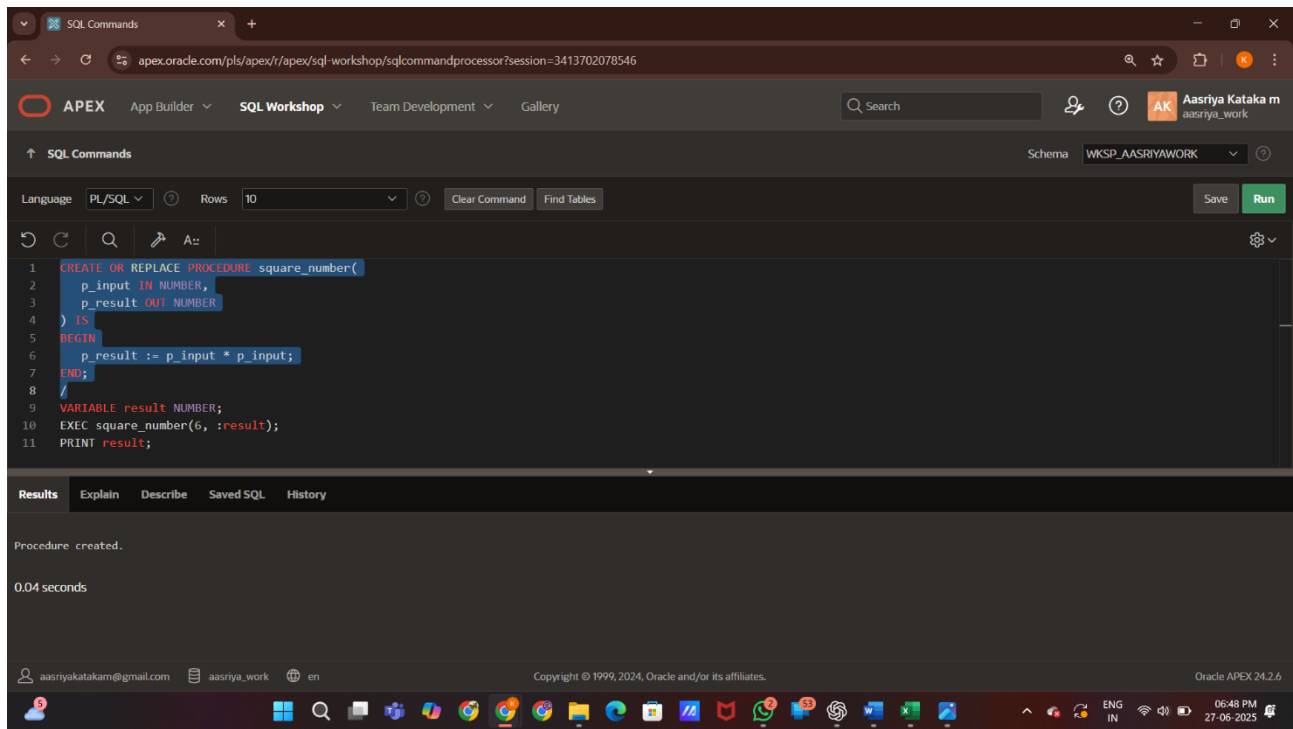
Output:



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:



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:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL editor contains the following code:

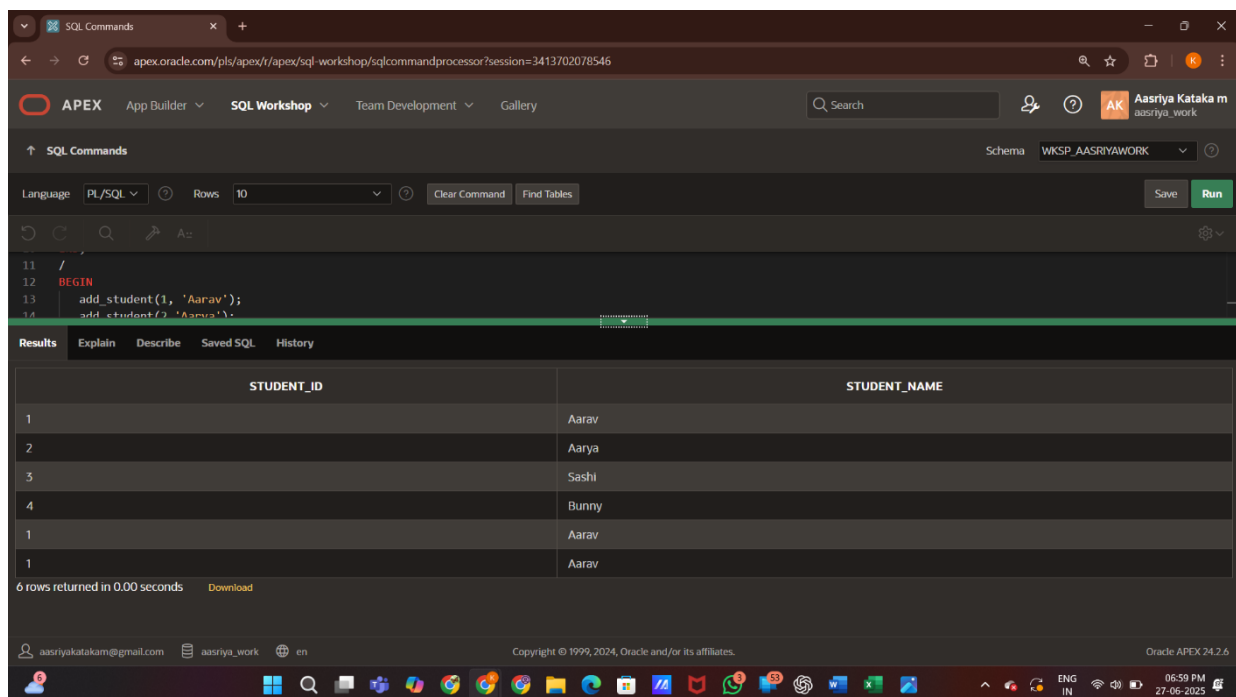
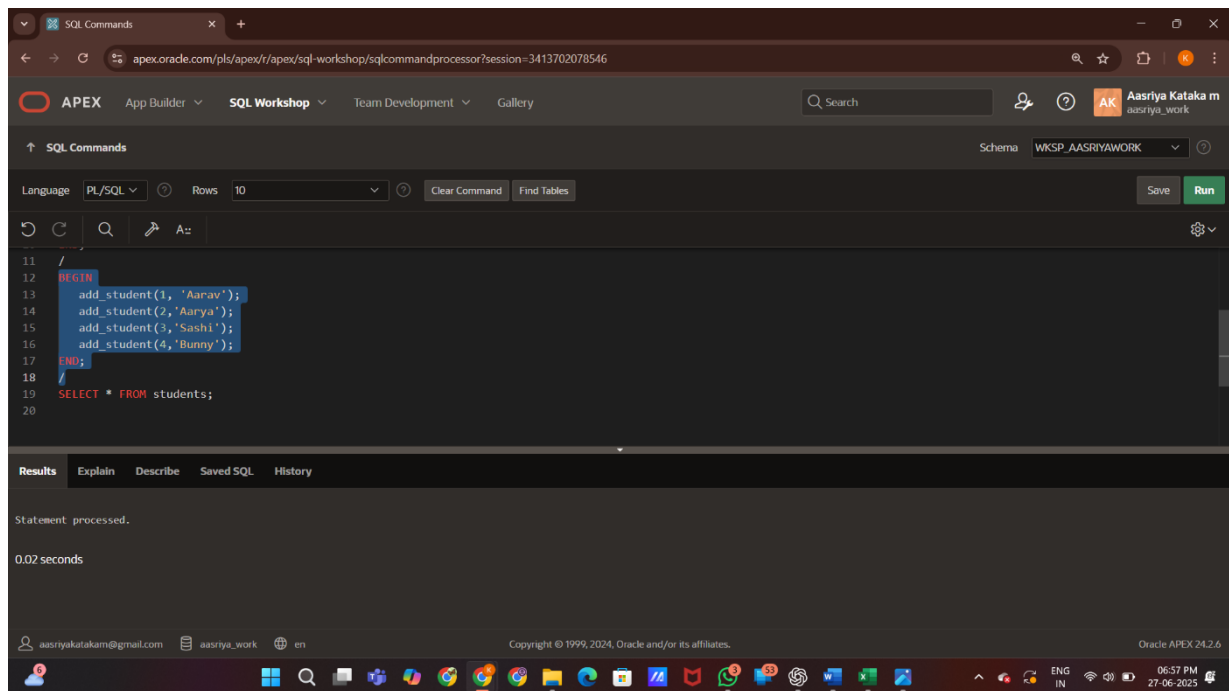
```
1 CREATE TABLE students (  
2     student_id NUMBER,  
3     student_name VARCHAR2(50)  
4 );  
5 CREATE OR REPLACE PROCEDURE add_student(p_id IN NUMBER, p_name IN VARCHAR2) IS  
6 BEGIN  
7     INSERT INTO students (student_id, student_name)  
8     VALUES (p_id, p_name);  
9     COMMIT;  
10 END;  
11 /  
12 BEGIN
```

The 'Results' tab is selected, displaying the message 'Table created.' and the execution time '0.02 seconds'.

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL editor contains the following code:

```
1 CREATE TABLE students (  
2     student_id NUMBER,  
3     student_name VARCHAR2(50)  
4 );  
5 CREATE OR REPLACE PROCEDURE add_student(p_id IN NUMBER, p_name IN VARCHAR2) IS  
6 BEGIN  
7     INSERT INTO students (student_id, student_name)  
8     VALUES (p_id, p_name);  
9     COMMIT;  
10 END;  
11 /  
12 BEGIN
```

The 'Results' tab is selected, displaying the message 'Procedure created.' and the execution time '0.03 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;
/

```

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The user is logged in as 'Aasriya Katakam'. The 'SQL Commands' tab is active, showing a PL/SQL script. The script defines a procedure 'check_pass' that takes a score and outputs 'Status: Passed' or 'Status: Failed'. Below the script, the 'Results' tab shows the message 'Procedure created.' and a execution time of '0.03 seconds'.

```

1 CREATE OR REPLACE PROCEDURE check_pass(p_score IN NUMBER) IS
2 BEGIN
3     IF p_score >= 50 THEN
4         DBMS_OUTPUT.PUT_LINE('Status: Passed');
5     ELSE
6         DBMS_OUTPUT.PUT_LINE('Status: Failed');
7     END IF;
8 END;
9 /
10 BEGIN
11     check_pass(45);

```

Results

Procedure created.

0.03 seconds

Output:

This screenshot shows the same Oracle APEX SQL Workshop interface, but with the 'check_pass' procedure being executed. The 'Results' tab now displays 'Status: Failed' and 'Statement processed.' with a execution time of '0.00 seconds'.

```

8 END;
9 /
10 BEGIN
11     check_pass(45);
12 END;
13 /

```

Results

Status: Failed

Statement processed.

0.00 seconds