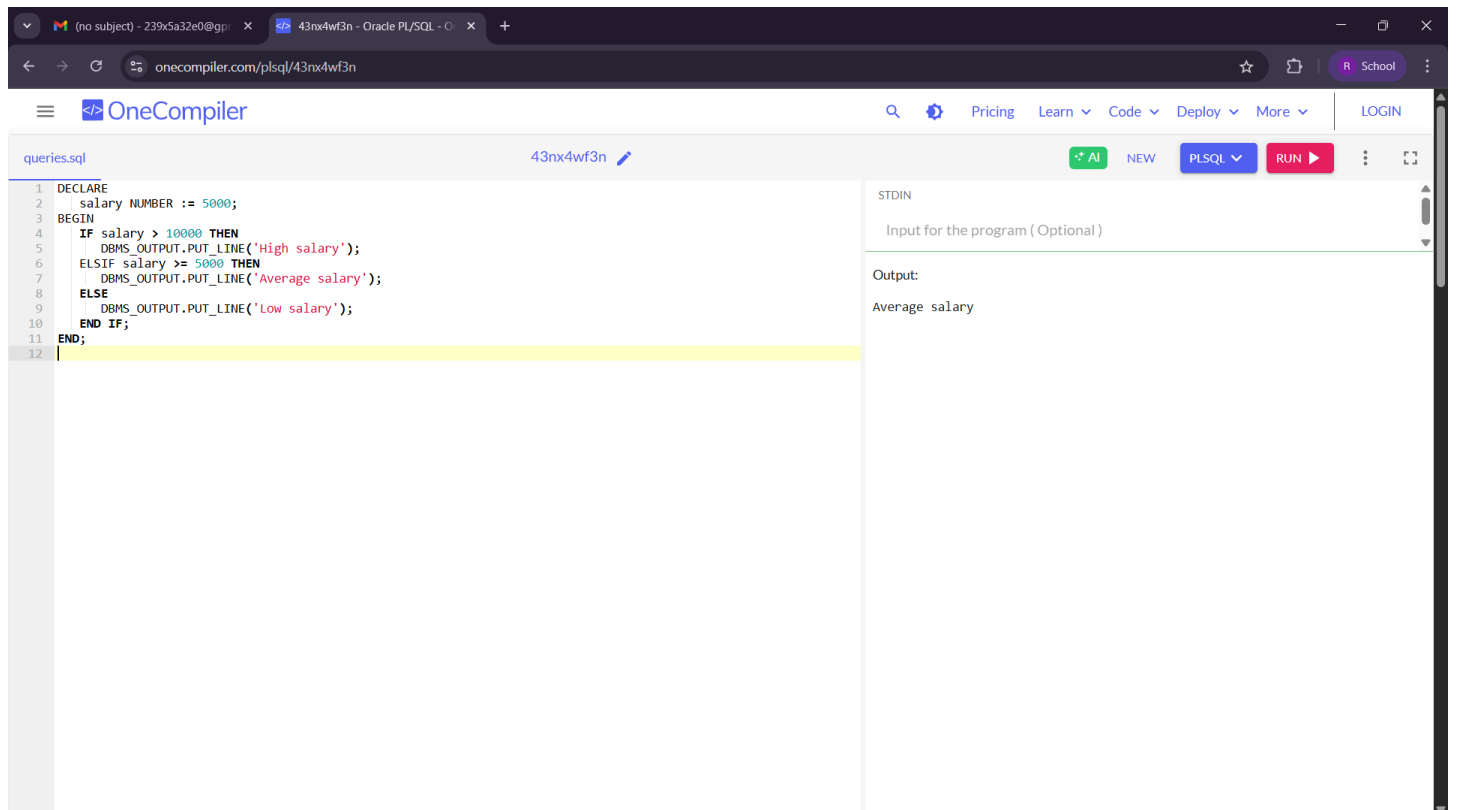


Week 2 hands on exercise solutions

PLSQL_Exercises:

Exercise 1: Control Structures



The screenshot displays the OneCompiler web interface for PL/SQL. The code editor on the left contains the following PL/SQL program:

```
1 DECLARE
2   salary NUMBER := 5000;
3 BEGIN
4   IF salary > 10000 THEN
5     DBMS_OUTPUT.PUT_LINE('High salary');
6   ELSIF salary >= 5000 THEN
7     DBMS_OUTPUT.PUT_LINE('Average salary');
8   ELSE
9     DBMS_OUTPUT.PUT_LINE('Low salary');
10  END IF;
11 END;
```

The right-hand side of the interface shows the execution results. Under the 'STDIN' section, there is a field for 'Input for the program (Optional)'. Under the 'Output' section, the text 'Average salary' is displayed, indicating the program's output for the given salary value.

Browser tabs: (no subject) - 239x5a32e0@gp... x 43nx4wf3n - Oracle PL/SQL - O x +

Address bar: onecompiler.com/plsql/43nx4wf3n

OneCompiler interface:

- Navigation: Pricing, Learn, Code, Deploy, More, LOGIN
- File: queries.sql, 43nx4wf3n
- Buttons: AI, NEW, PLSQL, RUN

```
1 DECLARE
2   i NUMBER := 1;
3 BEGIN
4   LOOP
5     DBMS_OUTPUT.PUT_LINE('Number: ' || i);
6     i := i + 1;
7     EXIT WHEN i > 5;
8   END LOOP;
9 END;
```

STDIN

Input for the program (Optional)

Output:

```
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
```

Browser tabs: (no subject) - 239x5a32e0@gp... x 43nx4wf3n - Oracle PL/SQL - O x +

Address bar: onecompiler.com/plsql/43nx4wf3n

OneCompiler interface:

- Navigation: Pricing, Learn, Code, Deploy, More, LOGIN
- File: queries.sql, 43nx4wf3n
- Buttons: AI, NEW, PLSQL, RUN

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

STDIN

Input for the program (Optional)

Output:

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

The screenshot shows the OneCompiler web interface. The code editor on the left contains a PL/SQL program named `queries.sql` with the following logic: it declares a `grade` character, then uses an `IF-ELSE` block to check for 'A' (Excellent!), 'B' (Very Good!), or 'C' (Invalid grade!). A `GOTO` statement is used to skip a message if the grade is 'C'. The output pane on the right shows "Processing completed."

```
1 DECLARE
2   grade CHAR := 'C';
3 BEGIN
4   IF grade = 'A' THEN
5     DBMS_OUTPUT.PUT_LINE('Excellent!');
6   ELSIF grade = 'B' THEN
7     DBMS_OUTPUT.PUT_LINE('Very Good!');
8   ELSIF grade = 'C' THEN
9     GOTO skip_message;
10  ELSE
11    DBMS_OUTPUT.PUT_LINE('Invalid grade');
12  END IF;
13  <<skip_message>>
14  DBMS_OUTPUT.PUT_LINE('Processing completed.');
```

The screenshot shows the OneCompiler web interface with a different PL/SQL program. The code editor on the left contains a program named `queries.sql` that uses a `FOR` loop to iterate from 1 to 5, printing the value of `i` in each iteration. The output pane on the right shows the results: "Value: 1", "Value: 2", "Value: 3", "Value: 4", and "Value: 5".

```
1 BEGIN
2   FOR i IN 1..5 LOOP
3     DBMS_OUTPUT.PUT_LINE('Value: ' || i);
4   END LOOP;
5 END;
```

Stored Procedures

queries.sql43nx4wf3n

AI

NEW

PLSQL

RUN

1

-- Single-page PL/SQL: Stored Procedure with IN, OUT, IN OUT and DBMS_OUTPUT

2

-- Create a procedure to calculate square and double of a number

4

CREATE OR REPLACE PROCEDURE process_number (

5

input_num IN NUMBER,

6

square_out OUT NUMBER,

7

double_inout IN OUT NUMBER

8

)

9

IS

10

BEGIN

11

square_out := input_num * input_num;

12

double_inout := double_inout * 2;

13

DBMS_OUTPUT.PUT_LINE('Procedure executed successfully.');

14

END;

15

/

16

-- Call the procedure in an anonymous block

18

DECLARE

19

n NUMBER := 4;

20

sqr NUMBER;

21

dbl NUMBER := 5;

22

BEGIN

23

process_number(n, sqr, dbl);

24

DBMS_OUTPUT.PUT_LINE('Input Number : ' || n);

25

DBMS_OUTPUT.PUT_LINE('Square (OUT) : ' || sqr);

26

DBMS_OUTPUT.PUT_LINE('Double (IN OUT) : ' || dbl);

27

END;

28

/

29

STDIN

Input for the program (Optional)

Output:

Procedure executed successfully.

Input Number : 4

Square (OUT) : 16

Double (IN OUT) : 10