PL/SQL

- **P**rocedural **L**anguage/**S**tructured **Q**uery **L**anguage
- Procedural commands (IF statements, loops, assignments)
- Organized within blocks
- Complement and extends SQL

Blocks

- Block-structured language
- Defined by DECLARE, BEGIN, EXCEPTION, END

- Can be divided into 3 sections
- 1. Declarative statements: Declare variables, constants, other code elements to be used in block. OPTIONAL
- 2. Executable statements: Executes statements in a block. MANDATORY
- 3. Exception handling: Catching exceptions from executable statements. OPTIONAL

Subprograms

- PL/SQL block that can be invoked repeatedly
- Subprogram can be a procedure or a function
- Procedure performs an action
- Function computes and returns a value

Procedure

- One possible way a procedure can be created is within the declarative statements and before the executable statements of a block
- DECLARE

```
        variable1 data_type
        PROCEDURE procedure_name(
                proc_var1 data_type
        )
        IS
        -- declarative statements
        BEGIN
        -- executable statements
        EXCEPTION
        -- exception handling
        END procedure_name;
BEGIN
        procedure_name(variable1);
        DBMS_OUTPUT.PUT_LINE('procedure_name invoked');
END;
```

Procedure

- Another way is just to define a procedure and compile and run it

```
set serveroutput on;
CREATE OR REPLACE  PROCEDURE TESTNO IS
proc_var number:=21;
BEGIN
   IF proc_var >= 20 THEN
      DBMS_OUTPUT.PUT_LINE('more than 20');
   ELSE
      DBMS_OUTPUT.PUT_LINE('less than 20');
   END IF;
END;
```

- Then after compile call that with EXECUTE statement separately as:

```
EXECUTE TESTNO ;
```

Triggers

- Similar to the procedure a trigger can be compiles and use . The difference is triggers are done automatic when they are defined to respond to INSERT, UPDATE and DELETE.
- For example to create a log file that keeps track of changes on order table one way is:

```
CREATE OR REPLACE TRIGGERS log AFTER INSERT ON order
FOR EACH ROW
BEGIN
INSERT INTO log (change_date, change_type, order_no)
VALUES(SYSDATE,'append',:NEW.order_no)
END;
```

- Then after compile on each insert to order table a new record contain order's order_no together with the date and type of action is added to the log table

Transaction Support in ORACLE (using Sqldeveloper)

- Try to write the procedure of Assignment 3 as a Transaction by using the following example of writing transaction with Sqldeveloper (5 marks):

drop table TESTJ;

create table TESTJ (Name varchar(30));

……..

SET TRANSACTION NAME 'Add name';

--------- use savepoint/commit/roleback wherever is required

……

SELECT * from TESTJ ;