PL/SQL Examples
Eric J. Schwabe
CSC 355 Winter 2020

(This document is for study and review purposes only. Copying any part of the examples in this document into a submitted assignment constitutes plagiarism.)

=====

```
-- Hello.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

declare
      -- no variables declared...
begin
      dbms_output.put_line( 'Hello Everyone!' );
      dbms_output.put_line( 'This is very simple PL/SQL output.' );
end;
/
```

=====

```
-- OutputScript.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

declare
      X NUMBER(2,0) := 10;
      Y X%type := 4;
      Z X%type;
      A X%type;
begin
      DBMS_OUTPUT.PUT_LINE( 'X is ' || X );
      DBMS_OUTPUT.PUT_LINE( 'Y is ' || Y );
      Z := X * Y;
      DBMS_OUTPUT.PUT_LINE( 'X*Y is ' || Z );
      A := X + Y;
      DBMS_OUTPUT.PUT_LINE( 'X+Y is ' || A );
end;
/
```

=====

```
-- InputScript.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

-- Using substitution variables for user input...

declare
      X NUMBER(2,0) := &X;
      Y X%type := &Y;
      Z X%type;
      A X%type;
begin
      DBMS_OUTPUT.PUT_LINE( 'X is ' || X );
      DBMS_OUTPUT.PUT_LINE( 'Y is ' || Y );
      Z := X * Y;
```

```
      DBMS_OUTPUT.PUT_LINE( 'X*Y is ' || Z );
      A := X + Y;
      DBMS_OUTPUT.PUT_LINE( 'X+Y is ' || A );
end;
/

=====

-- Branching.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

DECLARE

      votingage NUMBER(2,0) := 18;
      age votingage%type := &age;

BEGIN

      DBMS_OUTPUT.PUT( 'Age ' || age || ' is ' );

      IF (age >= votingage) THEN
            DBMS_OUTPUT.PUT_LINE( 'old enough to vote, and please do...');
      ELSE
            DBMS_OUTPUT.PUT_LINE( 'not old enough to vote yet...');
      END IF;

END;
/

=====

-- Case.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

DECLARE

      grade CHAR(1) := &grade;

BEGIN

-- one way...

      DBMS_OUTPUT.PUT('Grade ' || grade || ': ');
      IF (grade = 'A') THEN
            DBMS_OUTPUT.PUT_LINE('Excellent!');
      ELSIF (grade = 'B') THEN
            DBMS_OUTPUT.PUT_LINE('Very good.');
      ELSIF (grade = 'C') THEN
            DBMS_OUTPUT.PUT_LINE('Satisfactory.');
      ELSIF (grade = 'D') THEN
            DBMS_OUTPUT.PUT_LINE('Borderline.');
      ELSIF (grade = 'F') THEN
            DBMS_OUTPUT.PUT_LINE('Failed.');
      ELSE
            DBMS_OUTPUT.PUT_LINE('Invalid grade...?');
      END IF;

-- or the other way...
```

```
/*
      CASE (grade)

      WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent!');
      WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE('Very good.');
      WHEN 'C' THEN DBMS_OUTPUT.PUT_LINE('Satisfactory.');
      WHEN 'D' THEN DBMS_OUTPUT.PUT_LINE('Borderline.');
      WHEN 'F' THEN DBMS_OUTPUT.PUT_LINE('Failed.');
      ELSE DBMS_OUTPUT.PUT_LINE('Invalid grade...?');

      END CASE;
*/

END;
/

=====

-- BasicLoop.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

declare
      counter INTEGER := 5;
begin
      loop
            dbms_output.put_line(counter);

            if (counter = 0) then
                  dbms_output.put_line('Blast off!');
                  exit;
            else
                  counter := counter - 1;
            end if;
      end loop;
end;
/

=====

-- WhileLoop.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

declare
      counter INTEGER := 5;
begin
      while (counter >= 0) loop
            dbms_output.put_line(counter);
            counter := counter - 1;
      end loop;

      dbms_output.put_line('Blast off!');
end;
/

=====

-- ForLoop.sql
```

```
-- Eric J. Schwabe
-- CSC 355 Winter 2020

declare
      -- counter is declared implicitly by for loop...
begin
      for counter in reverse 0..5 loop
            dbms_output.put_line(counter);
      end loop;

      dbms_output.put_line('Blast off!');
end;
/

=====

-- SimpleQuery.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

-- First, build and display the GRADING table...

DROP TABLE GRADING;
CREATE TABLE GRADING
(
      CNumber     CHAR(7),
      CTitle      VARCHAR(20),
      SID         CHAR(3),
      SName       VARCHAR(20),
      Grade       VARCHAR(2),
      CONSTRAINT PK_GRADING
            PRIMARY KEY (CNumber, SID)
);
INSERT INTO GRADING
      VALUES ('CSC 355', 'Database Systems', '111', 'Tanner', 'A-');
INSERT INTO GRADING
      VALUES ('CSC 352', 'Database Programming', '111', 'Tanner', 'A-');
INSERT INTO GRADING
      VALUES ('CSC 370', 'Intro to Robotics', '222', 'Tanner', 'B+');
INSERT INTO GRADING
      VALUES ('CSC 452', 'Database Programming', '333', 'Gibler', 'A');
INSERT INTO GRADING
      VALUES ('CSC 355', 'Database Systems', '444', 'Katsopolis', 'A-');
SELECT * FROM GRADING;

-- This anonymous PL/SQL block defines three variables to hold
-- the values of a single record returned by a query, executes the
-- query and stores the results in the variables, and then
-- displays them...

declare
      c GRADING.CNumber%type;
      s GRADING.SName%type;
      g GRADING.Grade%type;
      targetID GRADING.SID%type := &targetID;
begin
      DBMS_OUTPUT.PUT( 'Querying GRADING table for SID ' || targetID || ' ... ');

      -- This statement stores the result of the query in the three variables
```

```
      SELECT CNumber, SName, Grade
      INTO c, s, g
      FROM GRADING
      WHERE SID = targetID;

      DBMS_OUTPUT.PUT_LINE( 'Query done!' );
      DBMS_OUTPUT.PUT_LINE('');

      DBMS_OUTPUT.PUT_LINE( 'For SID ' || targetID || ':');
      DBMS_OUTPUT.PUT_LINE( 'Name is ' || s ||
                        ', Course is ' || c ||
                        ', Grade is ' || g );
end;
/

=====

-- FancyQuery.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

-- First, build and display the GRADING table...

DROP TABLE GRADING;
CREATE TABLE GRADING
(
      CNumber     CHAR(7),
      CTitle      VARCHAR(20),
      SID         CHAR(3),
      SName       VARCHAR(20),
      Grade       VARCHAR(2),
      CONSTRAINT PK_GRADING
            PRIMARY KEY (CNumber, SID)
);
INSERT INTO GRADING
      VALUES ('CSC 355', 'Database Systems', '111', 'Tanner', 'A-');
INSERT INTO GRADING
      VALUES ('CSC 352', 'Database Programming', '111', 'Tanner', 'A-');
INSERT INTO GRADING
      VALUES ('CSC 370', 'Intro to Robotics', '222', 'Tanner', 'B+');
INSERT INTO GRADING
      VALUES ('CSC 452', 'Database Programming', '333', 'Gibler', 'A');
INSERT INTO GRADING
      VALUES ('CSC 355', 'Database Systems', '444', 'Katsopolis', 'A-');
SELECT * FROM GRADING;

-- This anonymous PL/SQL block defines three variables to hold
-- the values of a record returned by a query and a cursor to
-- traverse a set of records, executes a query, and then uses
-- the cursor to retrieve the records one at a time and display them...

declare
      c GRADING.CNumber%type;
      s GRADING.SName%type;
      g GRADING.Grade%type;

      cursor gradePtr is SELECT CNumber, SName, Grade FROM GRADING;
begin
      dbms_output.put_line( 'Traversing GRADING table with a cursor:' );
      dbms_output.put_line( '' );
```

```
      -- Opens the cursor

      open gradePtr;

      -- Loop fetches one record at a time until no more records are found

      loop
            fetch gradePtr into c, s, g;
            if gradePtr%found then
                  dbms_output.put_line( 'Row ' || gradePtr%rowcount || ' ...' );
                  dbms_output.put_line( ' Name: ' || s );
                  dbms_output.put_line( ' Course: ' || c );
                  dbms_output.put_line( ' Grade: '|| g);
                  dbms_output.put_line( '' );
            else
                  exit;
            end if;
      end loop;

      -- Closes the cursor

      close gradePtr;

      dbms_output.put_line( 'Done!' );
end;
/

=====

-- Records.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

-- First, build and display the GRADING table...

DROP TABLE GRADING;
CREATE TABLE GRADING
(
      CNumber     CHAR(7),
      CTitle      VARCHAR(20),
      SID         CHAR(3),
      SName       VARCHAR(20),
      Grade       VARCHAR(2),
      CONSTRAINT PK_GRADING
            PRIMARY KEY (CNumber, SID)
);
INSERT INTO GRADING
      VALUES ('CSC 355', 'Database Systems', '111', 'Tanner', 'A-');
INSERT INTO GRADING
      VALUES ('CSC 352', 'Database Programming', '111', 'Tanner', 'A-');
INSERT INTO GRADING
      VALUES ('CSC 370', 'Intro to Robotics', '222', 'Tanner', 'B+');
INSERT INTO GRADING
      VALUES ('CSC 452', 'Database Programming', '333', 'Gibler', 'A');
INSERT INTO GRADING
      VALUES ('CSC 355', 'Database Systems', '444', 'Katsopolis', 'A-');
SELECT * FROM GRADING;

-- This anonymous PL/SQL block defines a record to hold
```

```
-- the single complete talbe row returned by a query, executes
-- the query and stores the results in the record, and then
-- displays its fields...

declare
      gradeInfo GRADING%rowtype;
      targetID GRADING.SID%type := &targetID;

begin
      DBMS_OUTPUT.PUT_LINE( 'Querying GRADING table for ID ' || targetID || ':' );

      -- This statement stores the result of the query in the record

      SELECT *
      INTO gradeInfo
      FROM GRADING
      WHERE SID = targetID;

      DBMS_OUTPUT.PUT_LINE( 'Query done!' );
      DBMS_OUTPUT.PUT_LINE('');

      DBMS_OUTPUT.PUT_LINE( 'For SID ' || targetID || ':');
      DBMS_OUTPUT.PUT_LINE( 'Name is ' || gradeInfo.SName ||
                        ', Course is ' || gradeInfo.CNumber ||
                        ', Grade is ' || gradeinfo.Grade );
end;
/

=====

-- RecordsLoop.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

-- This anonymous PL/SQL block defines a record to hold
-- the set of complete rows returned by a query, executes the query
-- and uses a cursor to traverse the records in the result...
-- Note the use of a cursor for loop, where open/fetch/close are implicit...

declare
      gradeInfo GRADING%rowtype;

      cursor gradePtr is SELECT * FROM GRADING;
begin
      DBMS_OUTPUT.PUT_LINE( 'Querying GRADING table...' );
      DBMS_OUTPUT.PUT_LINE( '' );

      for gradeInfo in gradePtr
      loop
            dbms_output.put_line( ' SID: ' || gradeInfo.SID);
            dbms_output.put_line( ' Name: ' || gradeInfo.SName );
            dbms_output.put_line( ' Course: ' || gradeInfo.CNumber );
            dbms_output.put_line( ' Grade: '|| gradeInfo.Grade);
            dbms_output.put_line( '' );
      end loop;

      DBMS_OUTPUT.PUT_LINE( 'Done!' );

end;
/
```

```
=====

-- Procedure.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020
-- (not covered in class, just included as an additional example...)

-- Creating a named procedure

CREATE OR REPLACE PROCEDURE
      raisetopower (base IN INTEGER, exponent IN INTEGER, result IN OUT INTEGER)
AS
      x INTEGER:= 1;
BEGIN
      DBMS_OUTPUT.PUT_LINE(' raisetopower called with base = ' || base ||
                  ' and exponent = ' || exponent || ' ...');
      FOR i IN 1..exponent LOOP
            x := x * base;
      END LOOP;

      result := x;

      DBMS_OUTPUT.PUT_LINE(' ... now result = ' || x);
END;
/

-- Calling the procedure from an anonymous PL/SQL block

DECLARE
      a INTEGER := &a;
      b INTEGER := &b;
      c INTEGER;
BEGIN
      DBMS_OUTPUT.PUT_LINE( 'a = ' || a );
      DBMS_OUTPUT.PUT_LINE( 'b = ' || b );
      DBMS_OUTPUT.PUT_LINE( 'Calling procedure raisetopower...' );
      raisetopower(a, b, c);
      DBMS_OUTPUT.PUT_LINE( 'c = ' || c );
END;
/

=====

--- Function.sql
--- Eric J. Schwabe
--- CSC 355 Winter 2020
-- (not covered in class, just included as an additional example...)

-- Creating a named function

CREATE OR REPLACE FUNCTION
      raisetopoweragain (base IN INTEGER, exponent IN INTEGER) RETURN INTEGER
AS
      x INTEGER:= 1;
BEGIN
      FOR i IN 1..exponent LOOP
            x := x * base;
      END LOOP;
```

```
      return x;
END;
/

-- Calling the function from an anonymous block

DECLARE
      a INTEGER := &a;
      b INTEGER := &b;
BEGIN

      DBMS_OUTPUT.PUT_LINE( 'a is ' || a );
      DBMS_OUTPUT.PUT_LINE( 'b is ' || b );
      DBMS_OUTPUT.PUT_LINE( 'Calling function raisetopoweragain...' );
      DBMS_OUTPUT.PUT_LINE( 'result is ' || raisetopoweragain(a, b) );

END;
/

=====
```