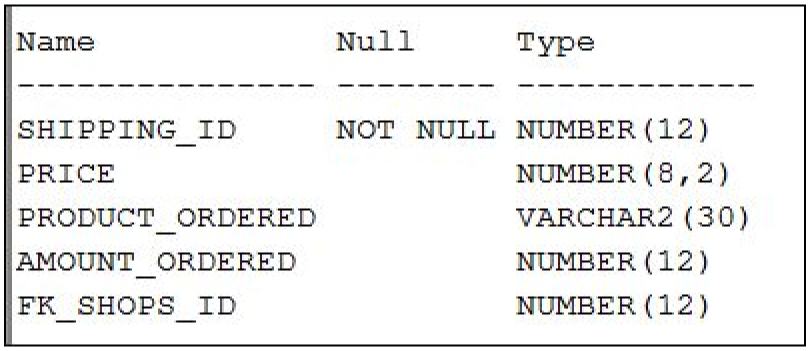
Robert Moore

Greg Swaim

CS 276 Week 5 Assignment – Records

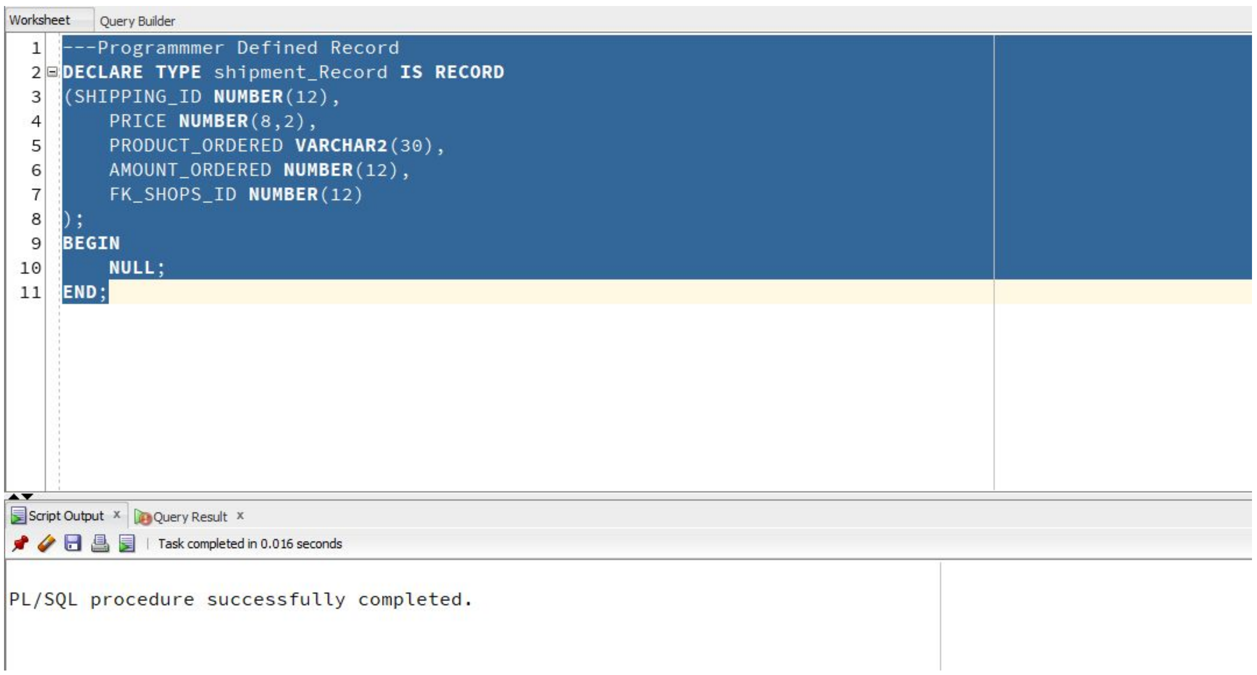
PART 1:

1) Declare a programmer defined record based on the following table:



```
Name            Null     Type
--------------- -------- ------------
SHIPPING_ID     NOT NULL NUMBER(12)
PRICE                    NUMBER(8,2)
PRODUCT_ORDERED          VARCHAR2(30)
AMOUNT_ORDERED           NUMBER(12)
FK_SHOPS_ID              NUMBER(12)
```

```
---Programmmer Defined Record
DECLARE TYPE shipment_Record IS RECORD
(SHIPPING_ID NUMBER(12),
   PRICE NUMBER(8,2),
   PRODUCT_ORDERED VARCHAR2(30),
   AMOUNT_ORDERED NUMBER(12),
   FK_SHOPS_ID NUMBER(12)
);
BEGIN
   NULL;
END;
```

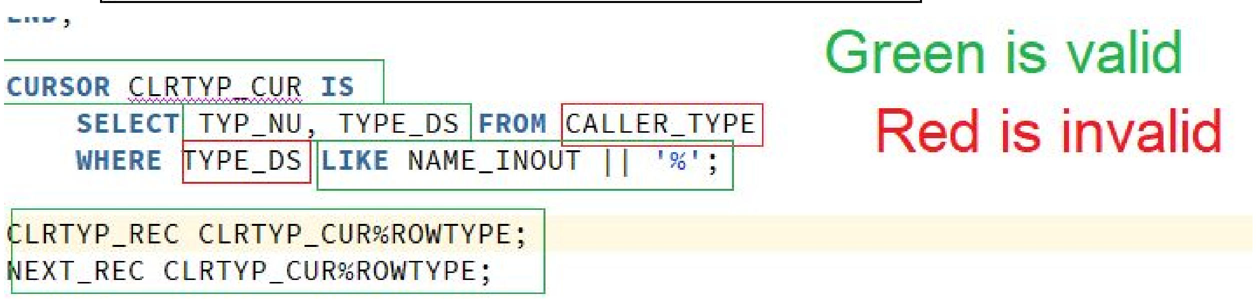


```
---Programmmer Defined Record
DECLARE TYPE shipment_Record IS RECORD
(SHIPPING_ID NUMBER(12),
    PRICE NUMBER(8,2),
    PRODUCT_ORDERED VARCHAR2(30),
    AMOUNT_ORDERED NUMBER(12),
    FK_SHOPS_ID NUMBER(12)
);
BEGIN
    NULL;
END;
```

```
PL/SQL procedure successfully completed.
```

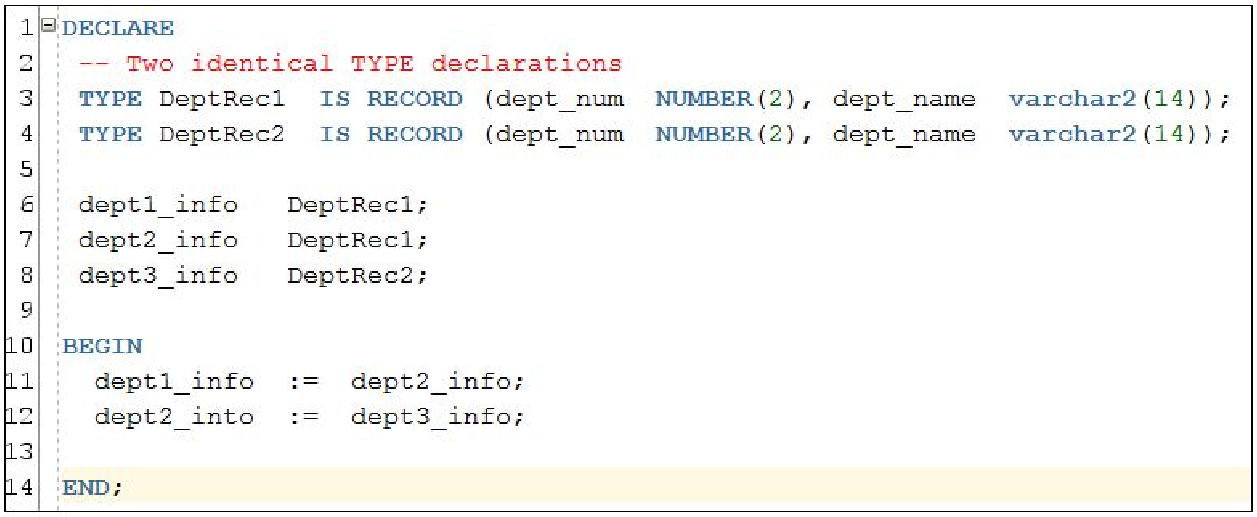
2) Is the following a legal definition of a ROWTYPE attribute for a row?

```
CURSOR CLRTYP_CUR IS
   SELECT TYP_NU, TYPE_DS
     FROM CALLER_TYPE
    WHERE TYPE_DS LIKE NAME_INOUT || '%';
CLRTYP_REC CLRTYP_CUR%ROWTYPE;
NEXT_REC CLRTYP_CUR%ROWTYPE;
```



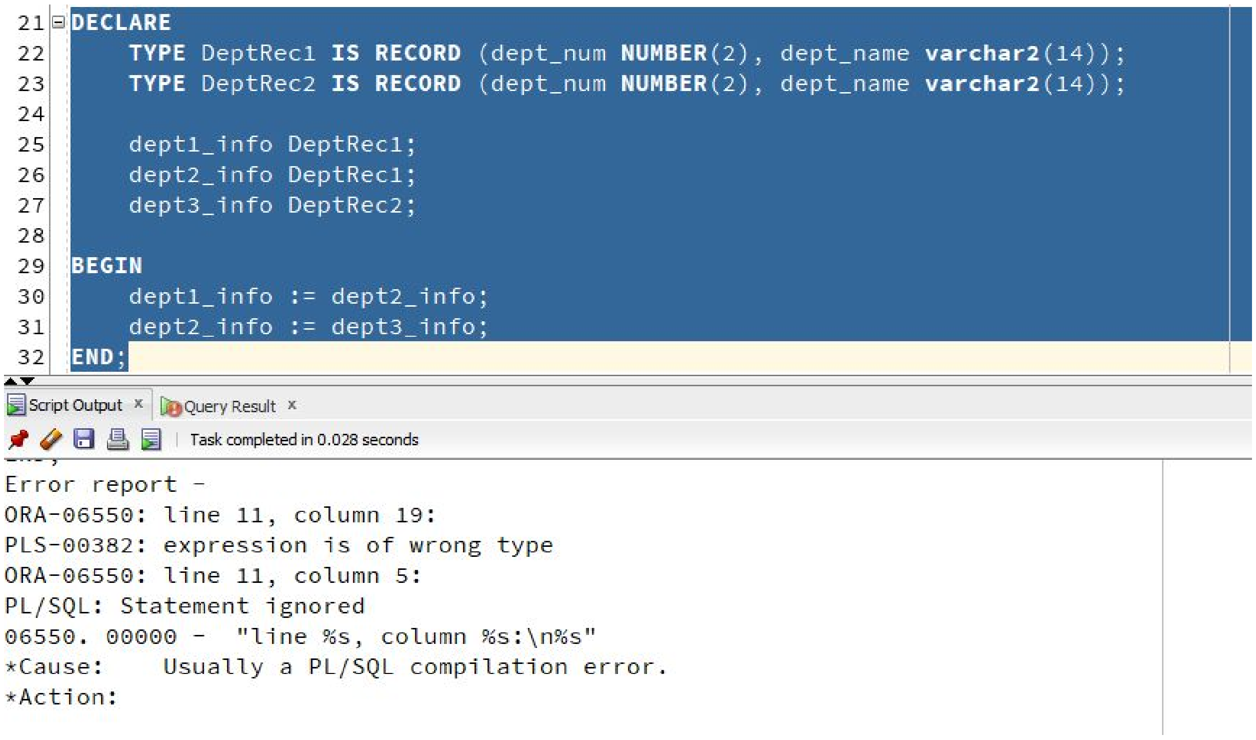
```
CURSOR CLRTYP_CUR IS
    SELECT TYP_NU, TYPE_DS FROM CALLER_TYPE
    WHERE TYPE_DS LIKE NAME_INOUT || '%';

CLRTYP_REC CLRTYP_CUR%ROWTYPE;
NEXT_REC CLRTYP_CUR%ROWTYPE;
```

3) Why won't the following code run?



```
DECLARE
 -- Two identical TYPE declarations
 TYPE DeptRec1  IS RECORD (dept_num  NUMBER(2), dept_name  varchar2(14));
 TYPE DeptRec2  IS RECORD (dept_num  NUMBER(2), dept_name  varchar2(14));

 dept1_info    DeptRec1;
 dept2_info    DeptRec1;
 dept3_info    DeptRec2;

BEGIN
  dept1_info  :=  dept2_info;
  dept2_into  :=  dept3_info;

END;
```

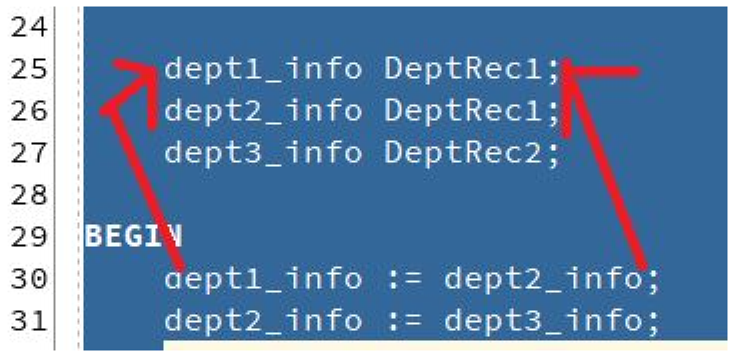
First problem



```
DECLARE
    TYPE DeptRec1 IS RECORD (dept_num NUMBER(2), dept_name varchar2(14));
    TYPE DeptRec2 IS RECORD (dept_num NUMBER(2), dept_name varchar2(14));

    dept1_info DeptRec1;
    dept2_info DeptRec1;
    dept3_info DeptRec2;

BEGIN
    dept1_info := dept2_info;
    dept2_info := dept3_info;
END;
```

```
Error report -
ORA-06550: line 11, column 19:
PLS-00382: expression is of wrong type
ORA-06550: line 11, column 5:
PL/SQL: Statement ignored
06550. 00000 -  "line %s, column %s:\n%s"
*Cause:    Usually a PL/SQL compilation error.
*Action:
```

Second Problem



```
    dept1_info DeptRec1;
    dept2_info DeptRec1;
    dept3_info DeptRec2;

BEGIN
    dept1_info := dept2_info;
    dept2_info := dept3_info;
```

Its trying to assign itself to itself, named something else. Attempt at locking? :)

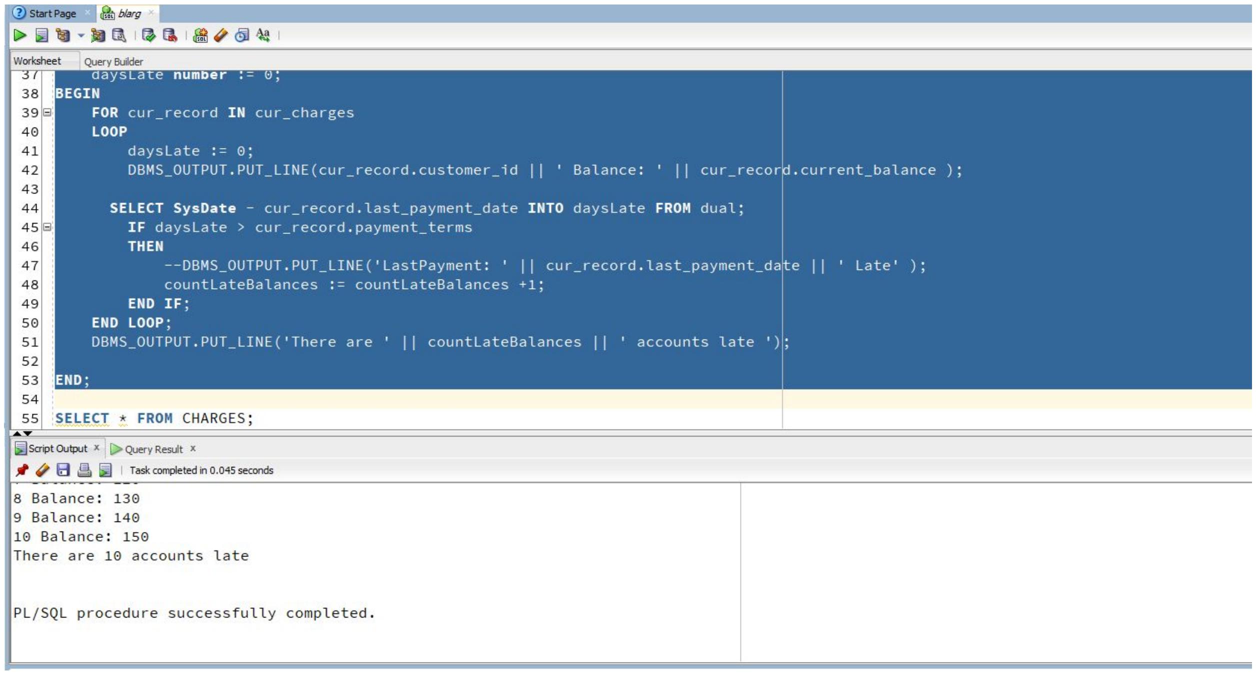
PART 2:

A record is a group of related data items stored in fields, each with its own name and datatype. You can think of a record as a variable that can hold a table row, or some columns from a table row. The fields correspond to table columns.

Records are composed of a group of fields, similar to the columns in a row. The **%ROWTYPE** attribute lets you declare a **PL/SQL record** that represents a row in a database table, without listing all the columns. Your code keeps working even after columns are added to the table. If you want to represent a subset of columns in a table, or columns from different tables, you can define a view or declare a cursor to select the right columns and do any necessary joins, and then apply **%ROWTYPE** to the view or cursor.

Run the Week 5 SQL file in the Moodle shell.

1) xxCreate an anonymous block that
   a) xxCreates a cursor of the fields in the CHARGES table.
   b) xxCreate a programmer defined record type that refers also refers to the columns in the table
   c) xxLoop through the table, fetching the record and printing out the current balance for each customer code
   d) After the loop, print out the number of late balances



```
    daysLate number := 0;
BEGIN
    FOR cur_record IN cur_charges
    LOOP
        daysLate := 0;
        DBMS_OUTPUT.PUT_LINE(cur_record.customer_id || ' Balance: ' || cur_record.current_balance );

      SELECT SysDate - cur_record.last_payment_date INTO daysLate FROM dual;
        IF daysLate > cur_record.payment_terms
        THEN
            --DBMS_OUTPUT.PUT_LINE('LastPayment: ' || cur_record.last_payment_date || ' Late' );
            countLateBalances := countLateBalances +1;
        END IF;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('There are ' || countLateBalances || ' accounts late ');

END;

SELECT * FROM CHARGES;
```

Task completed in 0.045 seconds

```
8 Balance: 130
9 Balance: 140
10 Balance: 150
There are 10 accounts late


PL/SQL procedure successfully completed.
```

SET serveroutput on;

DECLARE

  CURSOR cur_charges IS SELECT charge_id,customer_id,order_id,last_payment_date,payment_terms,current_balance FROM charges;

  TYPE charges_record IS RECORD(charge_id   CHARGES.CHARGE_ID%TYPE,

  customer_id   CHARGES.CUSTOMER_ID%TYPE,

  order_id   CHARGES.ORDER_ID%TYPE,

  last_payment_date   CHARGES.LAST_PAYMENT_DATE%TYPE,

  payment_terms   CHARGES.PAYMENT_TERMS%TYPE,

  current_balance CHARGES.CURRENT_BALANCE%TYPE);

  countLateBalances number := 0;

  daysLate number := 0;

BEGIN

  FOR cur_record IN cur_charges

```
  LOOP
    daysLate := 0;
    DBMS_OUTPUT.PUT_LINE(cur_record.customer_id || ' Balance: ' || cur_record.current_balance );


   SELECT SysDate - cur_record.last_payment_date INTO daysLate FROM dual;
    IF daysLate > cur_record.payment_terms
    THEN
      --DBMS_OUTPUT.PUT_LINE('LastPayment: ' || cur_record.last_payment_date || ' Late' );
      countLateBalances := countLateBalances +1;
    END IF;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('There are ' || countLateBalances || ' accounts late ');


END;
```

PART 3:

EXTRA CREDIT:

Create a table SHOPPER which has the following fields:

a) SHOPPER_ID
b) SHOPPER_FIRST_NAME
c) SHOPPER_LAST_NAME

Create a table SHOPPING LIST which has the following fields:

For product_size you would have a number, for product_unit you would have “gallon”,”ounces”, etc. so for a 5 gallon container of water PRODUCT_SIZE would be 5 and PRODUCT_UNIT would be “gallon”).

a) PRODUCT_ID

b) FK_SHOPPER_ID
c) PRODUCT_NAME
d) PRODUCT_PRICE
e) PRODUCT_SIZE
f) PRODUCT_UNIT

x

xDefine a record type SHOPPER_TYPE that has the same fields as the SHOPPER table.

xDefine a second record type SHOPPING_LIST_TYPE that has the same fields as the SHOPPING_LIST table.

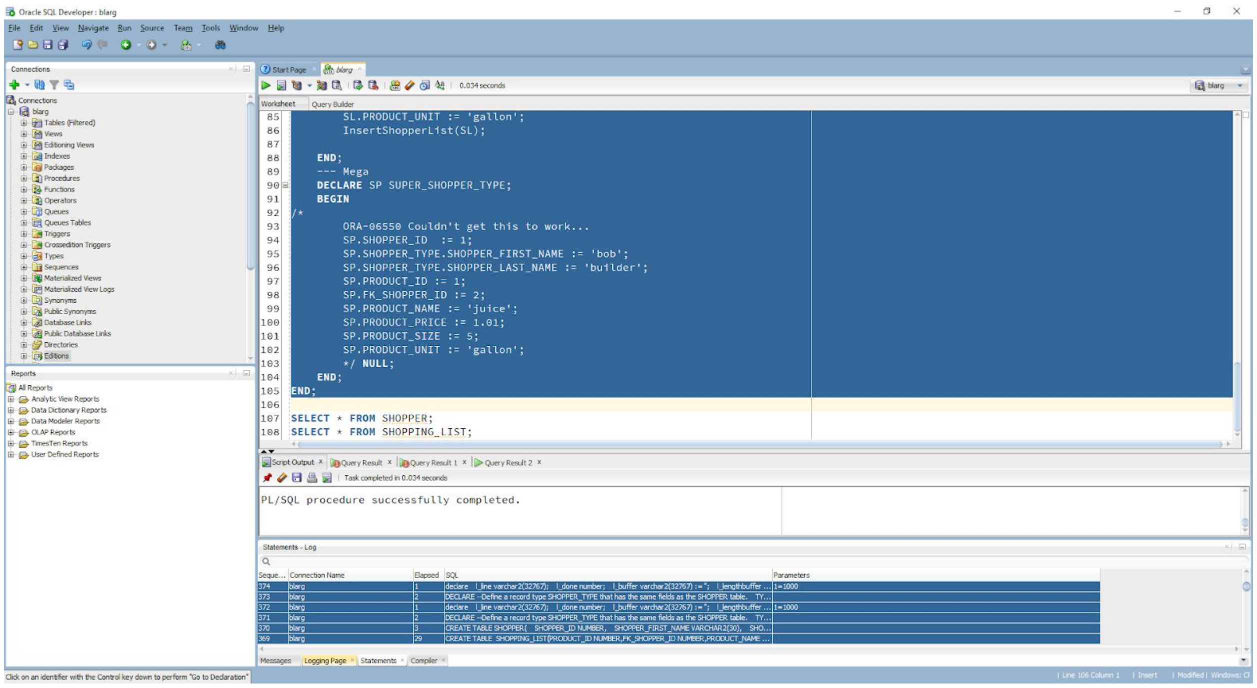
xDefine a third record type of the two above defined record types – i.e. one of SHOPPER and one of SHOPPING_LIST named SHOPPER_SHOPPING_LIST.

Define a fourth record type which is the compound record type of SHOPPER_SHOPPING_LIST.

xCreate two procedures, which should be declared in your declaration section, to insert to the two tables of SHOPPER and SHOPPING LIST.

xIn your execution section, assign values for all fields in the defined record types.

x Call your two procedures, inserting at least two rows into each.



```
        SL.PRODUCT_UNIT := 'gallon';
        InsertShopperList(SL);

    END;
    --- Mega
    DECLARE SP SUPER_SHOPPER_TYPE;
    BEGIN
/*
        ORA-06550 Couldn't get this to work...
        SP.SHOPPER_ID  := 1;
        SP.SHOPPER_TYPE.SHOPPER_FIRST_NAME := 'bob';
        SP.SHOPPER_TYPE.SHOPPER_LAST_NAME := 'builder';
        SP.PRODUCT_ID := 1;
        SP.FK_SHOPPER_ID := 2;
        SP.PRODUCT_NAME := 'juice';
        SP.PRODUCT_PRICE := 1.01;
        SP.PRODUCT_SIZE := 5;
        SP.PRODUCT_UNIT := 'gallon';
        */ NULL;
    END;
END;

SELECT * FROM SHOPPER;
SELECT * FROM SHOPPING_LIST;
```

Task completed in 0.034 seconds

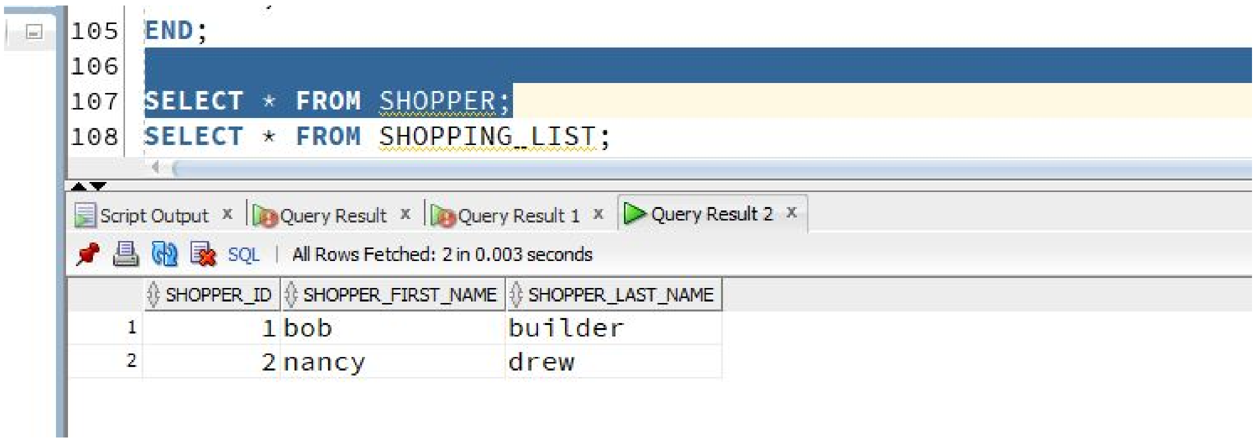
```
PL/SQL procedure successfully completed.
```

Seque...	Connection Name	Elapsed	SQL	Parameters
374	blarg	1	declare l_line varchar2(32767); l_done number; l_buffer varchar2(32767) :=''; l_lengthbuffer ...	1=1000
373	blarg	2	DECLARE --Define a record type SHOPPER_TYPE that has the same fields as the SHOPPER table. TY...	
372	blarg	1	declare l_line varchar2(32767); l_done number; l_buffer varchar2(32767) :=''; l_lengthbuffer ...	1=1000
371	blarg	2	DECLARE --Define a record type SHOPPER_TYPE that has the same fields as the SHOPPER table. TY...	
370	blarg	3	CREATE TABLE SHOPPER(SHOPPER_ID NUMBER, SHOPPER_FIRST_NAME VARCHAR2(30), SHO...	
369	blarg	29	CREATE TABLE SHOPPING_LIST(PRODUCT_ID NUMBER,FK_SHOPPER_ID NUMBER,PRODUCT_NAME ...	

```
CREATE TABLE SHOPPER(

   SHOPPER_ID NUMBER,

   SHOPPER_FIRST_NAME VARCHAR2(30),
   SHOPPER_LAST_NAME VARCHAR2(30)
);
--For product_size you would have a number, for product_unit you would have “gallon”,”ounces”, etc.
--so for a 5 gallon container of water PRODUCT_SIZE would be 5 and PRODUCT_UNIT would be “gallon”).
CREATE TABLE  SHOPPING_LIST(
PRODUCT_ID NUMBER,
FK_SHOPPER_ID NUMBER,
PRODUCT_NAME VARCHAR2(30),
PRODUCT_PRICE NUMBER(10,2),
PRODUCT_SIZE  NUMBER,
PRODUCT_UNIT VARCHAR2(10)
);

DECLARE
--Define a record type SHOPPER_TYPE that has the same fields as the SHOPPER table.
   TYPE SHOPPER_TYPE IS RECORD (
   SHOPPER_ID NUMBER,
   SHOPPER_FIRST_NAME VARCHAR2(30),
   SHOPPER_LAST_NAME VARCHAR2(30));
--Define a second record type SHOPPING_LIST_TYPE that has the same fields as the SHOPPING_LIST table.
   TYPE  SHOPPER_LIST_TYPE IS RECORD (PRODUCT_ID NUMBER,
   FK_SHOPPER_ID NUMBER,
   PRODUCT_NAME VARCHAR2(30),
   PRODUCT_PRICE NUMBER(10,2),
   PRODUCT_SIZE  NUMBER,
   PRODUCT_UNIT VARCHAR2(10));
--Define a third record type of the two above defined record types – i.e. one of SHOPPER and      one of SHOPPING_LIST named SHOPPER_SHOPPING_LIST.
```

```
  TYPE SUPER_SHOPPER_TYPE IS RECORD (
  SHOPPER_ID SHOPPER_TYPE,
  SHOPPER_FIRST_NAME# SHOPPER_TYPE,
  SHOPPER_LAST_NAME# SHOPPER_TYPE,
  PRODUCT_ID# SHOPPER_LIST_TYPE,
  FK_SHOPPER_ID# SHOPPER_LIST_TYPE,
  PRODUCT_NAME# SHOPPER_LIST_TYPE,
  PRODUCT_PRICE# SHOPPER_LIST_TYPE,
  PRODUCT_SIZE# SHOPPER_LIST_TYPE,
  PRODUCT_UNIT# SHOPPER_LIST_TYPE);

  PROCEDURE InsertShopper(ST IN SHOPPER_TYPE)
  IS
  BEGIN
    INSERT INTO SHOPPER VALUES ST;
  END;

  PROCEDURE InsertShopperList(SL IN SHOPPER_LIST_TYPE)
  IS
  BEGIN
    INSERT INTO SHOPPING_LIST VALUES SL;
  END;



BEGIN
  -- Assign values to each type
  DECLARE ST SHOPPER_TYPE;
  BEGIN
```

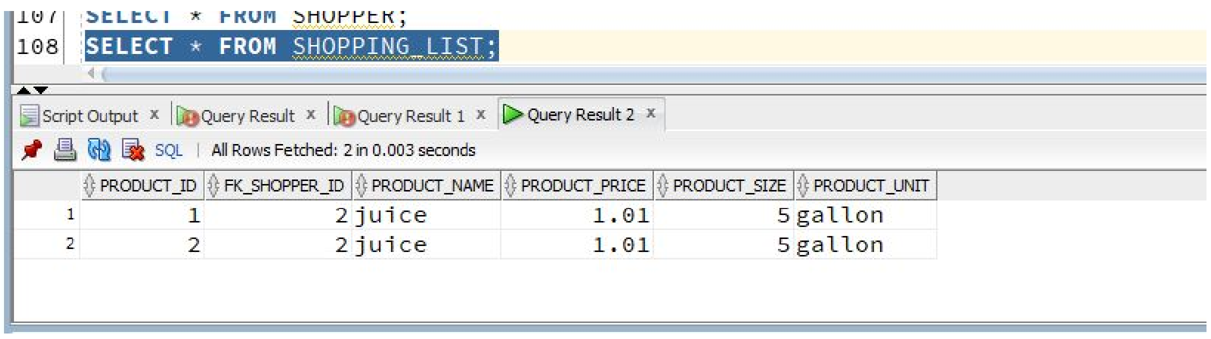
```
    ST.SHOPPER_ID  := 1;
    ST.SHOPPER_FIRST_NAME := 'bob';
    ST.SHOPPER_LAST_NAME := 'builder';
    InsertShopper(ST);

    ST.SHOPPER_ID  := 2;
    ST.SHOPPER_FIRST_NAME := 'nancy';
    ST.SHOPPER_LAST_NAME := 'drew';
    InsertShopper(ST);
END;
-- SHopping List
DECLARE SL SHOPPER_LIST_TYPE;
BEGIN
    SL.PRODUCT_ID := 1;
    SL.FK_SHOPPER_ID := 2;
    SL.PRODUCT_NAME := 'juice';
    SL.PRODUCT_PRICE := 1.01;
    SL.PRODUCT_SIZE := 5;
    SL.PRODUCT_UNIT := 'gallon';
    InsertShopperList(SL);

    SL.PRODUCT_ID := 2;
    SL.FK_SHOPPER_ID := 2;
    SL.PRODUCT_NAME := 'juice';
    SL.PRODUCT_PRICE := 1.01;
    SL.PRODUCT_SIZE := 5;
    SL.PRODUCT_UNIT := 'gallon';
    InsertShopperList(SL);
    END;

    --- Mega

    DECLARE SP SUPER_SHOPPER_TYPE;

    BEGIN

/*

      ORA-06550 Couldn't get this to work...

      SP.SHOPPER_ID  := 1;

      SP.SHOPPER_TYPE.SHOPPER_FIRST_NAME := 'bob';

      SP.SHOPPER_TYPE.SHOPPER_LAST_NAME := 'builder';

      SP.PRODUCT_ID := 1;

      SP.FK_SHOPPER_ID := 2;

      SP.PRODUCT_NAME := 'juice';

      SP.PRODUCT_PRICE := 1.01;

      SP.PRODUCT_SIZE := 5;

      SP.PRODUCT_UNIT := 'gallon';

      */ NULL;

    END;

END;
```



```
END;

SELECT * FROM SHOPPER;
SELECT * FROM SHOPPING_LIST;
```

All Rows Fetched: 2 in 0.003 seconds

	SHOPPER_ID	SHOPPER_FIRST_NAME	SHOPPER_LAST_NAME
1	1	bob	builder
2	2	nancy	drew



```
SELECT * FROM SHOPPER;
SELECT * FROM SHOPPING_LIST;
```

All Rows Fetched: 2 in 0.003 seconds

	PRODUCT_ID	FK_SHOPPER_ID	PRODUCT_NAME	PRODUCT_PRICE	PRODUCT_SIZE	PRODUCT_UNIT
1	1	2	juice	1.01	5	gallon
2	2	2	juice	1.01	5	gallon

PART 2:

1) How would I alter the above so that I am able to PASS values for my fields in the defined record types, such that I don’t have to hard code them in the code itself? You will need to create a package that contains your procedures and variables.