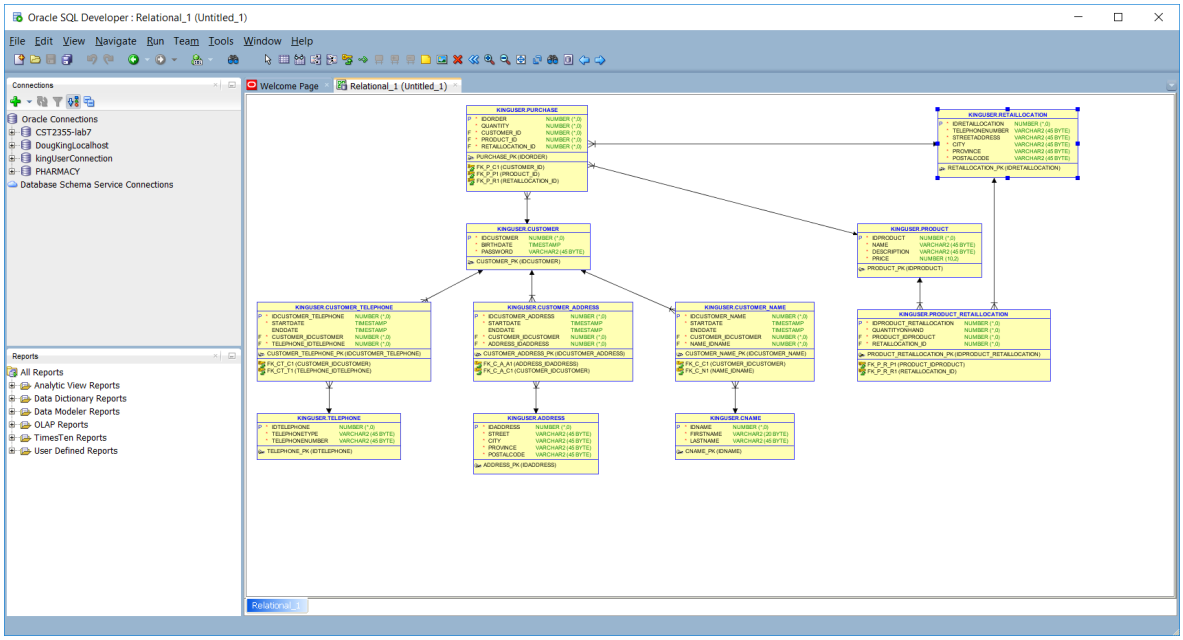
CST2355 – Database Systems Lab Assignment 9

Hand-in:

1. The lab assignment will be graded out of a maximum 4 points.
2. This template should be used to submit your lab assignment.
3. Make sure you have enough screenshots to completely document that you have completed all the steps.

Activities (Steps):

1. We are going to create a package that shares items across a set of stored procedures and functions, based on the model that was used in lab 8: (see below)



1.1. First look at the tutorial at: https://www.tutorialspoint.com/plsql/plsql_packages.htm It contains an implementation of a package to manage customer records.

1.2. Prepare an sql script called "lab9-sequences.sql" that contains the CREATE SEQUENCE statements to create sequences **that will be used in this lab when inserting new entries in each of the tables in your schema**. Choose appropriate starting values so that the existing data is not in conflict with the new numbers. (e.g., start them all at 100?) Run the script to create the sequences.

1.2.1. Provide a screenshot showing the contents of your script.

```
-- SEQUENCE FOR ADDRESS TABLE
CREATE SEQUENCE ADDRESS_LAB9_SEQ
INCREMENT BY 1
START WITH 100
NOCYCLE
NOCACHE;
-- SEQUENCE FOR CNAME TABLE
CREATE SEQUENCE CNAME_LAB9_SEQ
INCREMENT BY 1
START WITH 100
NOCYCLE
NOCACHE;
-- SEQUENCE FOR CUSTOMER TABLE
CREATE SEQUENCE CUSTOMER_LAB9_SEQ
INCREMENT BY 1
START WITH 100
NOCYCLE
NOCACHE;
-- SEQUENCE FOR CUSTOMER_ADDRESS TABLE
CREATE SEQUENCE CUSTOMER_ADDRESS_LAB9_SEQ
```

1.2.2. Provide a screenshot showing the successful running of the sequences script.

```
Sequence ADDRESS_LAB9_SEQ created.

Sequence CNAME_LAB9_SEQ created.

Sequence CUSTOMER_LAB9_SEQ created.

Sequence CUSTOMER_ADDRESS_LAB9_SEQ created.

Sequence CUSTOMER_NAME_LAB9_SEQ created.

Sequence CUSTOMER_TELEPHONE_LAB9_SEQ created.

Sequence PRODUCT_LAB9_SEQ created.
```

2. Create a package in the *yourName*User schema called customer_pkg with the following specification:

2.1. The package should contain three stored procedures that each use an IDCUSTOMER IN parameter along with the appropriate IN parameters to update the underlying tables.

2.1.1. change_telephone()

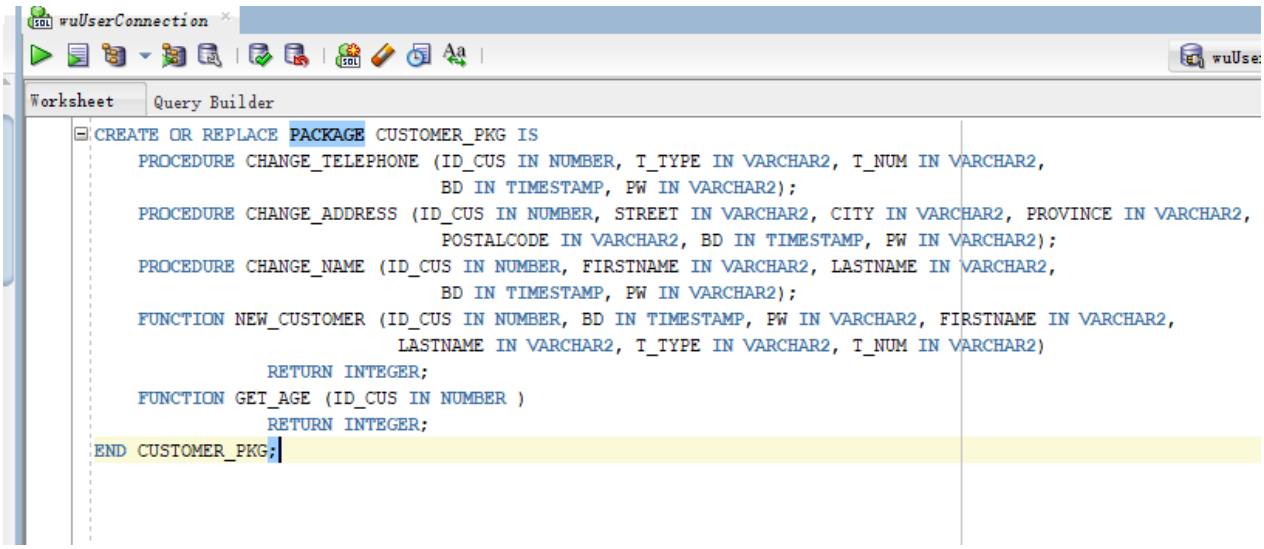
2.1.2. change_address()

2.1.3. change_name()

2.2. The package should also contain a function called new_customer() that returns an INTEGER containing the IDCUSTOMER for a new customer record. The new_customer() function should have mandatory fields for birthdate and password, and optional fields for each of the data fields in the telephone, address, and name tables.

2.3. The package should also contain a function called get_age() that returns an INTEGER containing the age in years (rounded down to the nearest integer value), for a given IDCUSTOMER.

2.4. Provide a screenshot or screenshots showing the package **specification** (just the package specification – not the entire package body) below:



3. Provide the package body for customer_pkg using the following criteria.

3.1.1. Each procedure should <u>insert</u> a new telephone/address/name as appropriate

3.1.2. Each procedure should <u>update</u> the entry in the relationship association table to have the sysdate timestamp as the enddate for the current entry (that is the one with NULL enddate gets updated to have enddate as sysdate). If there is no previous related item (i.e., no current entry with a NULL enddate) then this step should get skipped.

3.1.3. Each procedure should <u>insert</u> a new record in the relationship association table that has the startdate as sysdate and a NULL enddate.

3.1.4. The new_customer() function should create the customer record along with the telephone, address, and name records as required. If all the non-key fields in a telephone, address or name record would be null, then the associated record should not be created. [**<u>NOTE: The new_customer function should use the three stored procedures in the package to create the related records.</u>**]

3.1.5. The get_age() function should returns an INTEGER containing the age in years (rounded down to the nearest integer value), for a given IDCUSTOMER.

3.1.6. Provide screenshots for the three stored procedures below:

```
Worksheet   Query Builder
PROCEDURE CHANGE_TELEPHONE (ID_CUS IN NUMBER, T_TYPE IN VARCHAR2, T_NUM IN VARCHAR2, BD IN TIMESTAMP, PW IN VARCHAR2) IS
CUS_T_ID_QTY NUMBER;
TEL_LAB9_SEQ INTEGER;
BEGIN
        SELECT COUNT(*) INTO CUST_T_ID_QTY FROM CUSTOMER_TELEPHONE WHERE CUSTOMER_IDCUSTOMER = ID_CUS;
        IF (CUS_T_ID_QTY = 1) THEN
            UPDATE CUSTOMER_TELEPHONE SET ENDDATE = SYSDATE WHERE ENDDATE IS NULL AND CUSTOMER_IDCUSTOMER = ID_CUS;
        END IF;
-- INSERTING NEW INFO INTO RELATED TABLE
TEL_LAB9_SEQ := TELEPHONE_LAB9.SEQ.NEXTVAL;
INSERT INTO CUSTOMER VALUES (ID_CUS, BD, PW);
INSERT INTO TELEPHONE VALUES (TEL_LAB9_SEQ, T_TYPE, T_NUM);
INSERT INTO CUSTOMER_TELEPHONE VALUES (CUSTOMER_TELEPHONE_LAB9_SEQ.NEXTVAL, SYSDATE, NULL, ID_CUS, TEL_LAB9_SEQ);
END CHANGE_TELEPHONE;
PROCEDURE CHANGE_ADDRESS (ID_CUS IN NUMBER, STREET IN VARCHAR2, CITY IN VARCHAR2, PROVINCE IN VARCHAR2, POSTALCODE IN VARCHAR2, BD IN TIMESTAMP, PW IN VARCHAR2) IS
CUST_A_ID_QTY NUMBER;
ADS_LAB9_SEQ INTEGER;
BEGIN
        SELECT COUNT(*) INTO CUST_A_ID_QTY FROM CUSTOMER_ADDRESS WHERE CUSTOMER_IDCUSTOMER = ID_CUS;
        IF (CUS_A_ID_QTY = 1) THEN
            UPDATE CUSTOMER_ADDRESS SET ENDDATE = SYSDATE WHERE ENDDATE IS NULL AND CUSTOMER_IDCUSTOMER = ID_CUS;
        END IF;
-- INSERTING NEW INFO INTO RELATED TABLE
ADS_LAB9_SEQ := ADDRESS_LAB9.SEQ.NEXTVAL;
INSERT INTO CUSTOMER VALUES (ID_CUS, BD, PW);
INSERT INTO ADDRESS VALUES (ADS_LAB9_SEQ, STREET, CITY, PROVINCE, POSTALCODE);
INSERT INTO CUSTOMER_ADDRESS VALUES (CUSTOMER_ADDRESS_LAB9_SEQ.NEXTVAL, SYSDATE, NULL, ID_CUS, ADS_LAB9_SEQ);
END CHANGE_ADDRESS;
PROCEDURE CHANGE_NAME (ID_CUS IN NUMBER, FIRSTNAME IN VARCHAR2, LASTNAME IN VARCHAR2, BD IN TIMESTAMP, PW IN VARCHAR2) IS
CUST_N_ID_QTY NUMBER;
N_LAB9_SEQ INTEGER;
BEGIN
        SELECT COUNT(*) INTO CUST_N_ID_QTY FROM CUSTOMER_NAME WHERE CUSTOMER_IDCUSTOMER = ID_CUS;
        IF (CUS_N_ID_QTY = 1) THEN
            UPDATE CUSTOMER_NAME SET ENDDATE = SYSDATE WHERE ENDDATE IS NULL AND CUSTOMER_IDCUSTOMER = ID_CUS;
        END IF;
-- INSERTING NEW INFO INTO RELATED TABLE
N_LAB9_SEQ := CNAME_LAB9.SEQ.NEXTVAL;
INSERT INTO CUSTOMER VALUES (ID_CUS, BD, PW);
INSERT INTO CNAME VALUES (N_LAB9_SEQ, FIRSTNAME, LASTNAME);
INSERT INTO CUSTOMER_NAME VALUES (CUSTOMER_NAME_LAB9_SEQ.NEXTVAL, SYSDATE, NULL, ID_CUS, N_LAB9_SEQ);
END CHANGE_NAME;
```

3.1.7. Provide a screenshot showing the new_customer function below:

```
FUNCTION NEW_CUSTOMER (ID_CUS IN NUMBER, BD IN TIMESTAMP, PW IN VARCHAR2, ST IN VARCHAR2, CT IN VARCHAR2, PROV IN VARCHAR2,
                       PCODE IN VARCHAR2, FN IN VARCHAR2, LN IN VARCHAR2, T_TYPE IN VARCHAR2, T_NUM IN VARCHAR2) RETURN INTEGER
IS
T_ID NUMBER (7,0);
A_ID NUMBER (7,0);
N_ID NUMBER (7,0);
BEGIN
T_ID := TELEPHONE_LAB9_SEQ.NEXTVAL;
A_ID := ADDRESS_LAB9_SEQ.NEXTVAL;
N_ID := CNAME_LAB9_SEQ.NEXTVAL;
IF (T_TYPE IS NOT NULL) OR (T_NUM IS NOT NULL) THEN
    CHANGE_TELEPHONE (ID_CUS, T_TYPE, T_NUM, BD, PW);
END IF;
IF (ST IS NOT NULL) OR (CT IS NOT NULL) OR (PROV IS NOT NULL) OR (PCODE IS NOT NULL) THEN
    CHANGE_ADDRESS (ID_CUS, ST, CT, PROV, PCODE, BD, PW);
END IF;
IF (FN IS NOT NULL) OR (LN IS NOT NULL) THEN
    CHANGE_NAME (ID_CUS, FN, LN, BD, PW);
END IF;
RETURN ID_CUS;
END NEW_CUSTOMER;
```

3.1.8. Provide a screenshot showing the get_age function below:

```
FUNCTION GET_AGE (ID_CUS IN NUMBER) RETURN INTEGER
IS
AGE INTEGER;
BEGIN
SELECT TRUNC(MONTHS_BETWEEN(SYSDATE,BIRTHDATE)/12) INTO AGE FROM CUSTOMER WHERE IDCUSTOMER = ID_CUS;
RETURN AGE;
END GET_AGE;
```

4. You will be creating a package based on distance calculations, and to facilitate that package, you need to alter the table definitions as follows:

   4.1. Add fields named “longitude” and “latitude” of type NUMBER(12,6) to house the longitude and latitude associated with each address and retail_location.

   4.2. **Manually** use Google Maps to lookup the longitude and latitude for each location you currently have populated in your address and retail_location tables.

   4.3. Provide a screenshot of your updated address table showing the latitudes and longitudes.

wuUserConnection_LAB9.sql | ADDRESS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	IDADDRESS	STREET	CITY	PROVINCE	POSTALCODE	LONGITUDE	LATITUDE
1	2	234 Derek Drive	Toronto	Ontario	M5W1E5	43.79	-79.16
2	3	1 Lewis Street	Ottawa	Ontario	K2S1E6	45.41	-75.68
3	4	78 Patterson Street	Ottawa	Ontario	K1G7E3	45.4	-75.68
4	5	2-67 Rideau Court	Ottawa	Ontario	K1K1H0	45.42	-75.69
5	6	1 North Pole Circle	North Pole	Nunavut	H0H0H0	83.01	-74.8
6	7	2 North Pole Circle	North Pole	Nunavut	H0H0H0	76.75	-100.31
7	8	11 Lewis Street	Ottawa	Ontario	K2S1E6	45.41	-75.68

   4.4. Provide a screenshot of your updated retail_location table showing the latitudes and longitudes.

wuUserConnection_LAB9.sql | RETAILLOCATION

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	IDRETAILLOCATION	TELEPHONENUMBER	STREETADDRESS	CITY	PROVINCE	POSTALCODE	LONGITUDE	LATITUDE
1	3	(613) 678-9012	34 Sussex Drive	Ottawa	Ontario	K2G2S7	45.44	-75.69
2	4	(519) 789-0123	100 College Avenue	Guelph	Ontario	N1H2G3	43.53	-80.22

5. Create a package in the *yourName*User schema called locations_pkg with the following specification:

5.1.1. The package should contain a function calc_kms (longitude1, latitude1, longitude2, latitude2) that takes 4 numbers and returns the distance between the two points. You should base your function on the file provided in Brightspace (which uses the sdo_geom package from Oracle).

5.1.1.1. Provide a screenshot showing your calc_kms () implementation after being compiled in your database.

wuUserConnection_LAB9.sql

Worksheet | Query Builder

```
create or replace FUNCTION CALC_KMS (ID_CUS IN NUMBER, ID_RET IN NUMBER) RETURN NUMBER
IS
CUS_LONGT NUMBER;
CUS_LATI NUMBER;
RET_LONGT NUMBER;
RET_LATI NUMBER;
DIST NUMBER;
BEGIN
    SELECT LONGITUDE INTO CUS_LONGT FROM ADDRESS
    WHERE IDADDRESS = (SELECT ADDRESS_IDADDRESS FROM CUSTOMER_ADDRESS WHERE CUSTOMER_IDCUSTOMER = ID_CUS AND ENDDATE IS NULL);
    SELECT LATITUDE INTO CUS_LATI FROM ADDRESS
    WHERE IDADDRESS = (SELECT ADDRESS_IDADDRESS FROM CUSTOMER_ADDRESS WHERE CUSTOMER_IDCUSTOMER = ID_CUS AND ENDDATE IS NULL);
    SELECT LONGITUDE INTO RET_LONGT FROM CUSTOMER C, RETAILLOCATION R, PURCHASE P
    WHERE C.IDCUSTOMER = P.CUSTOMER_ID AND R.IDRETAILLOCATION = P.RETAILLOCATION_ID AND R.IDRETAILLOCATION = ID_RET;
    SELECT LATITUDE INTO RET_LATI FROM CUSTOMER C, RETAILLOCATION R, PURCHASE P
    WHERE C.IDCUSTOMER = P.CUSTOMER_ID AND R.IDRETAILLOCATION = P.RETAILLOCATION_ID AND R.IDRETAILLOCATION = ID_RET;
--USING SDO_GEOM PACKAGE
    SELECT SDO_GEOM.SDO_DISTANCE(
        SDO_GEOMETRY(2001,4326,SDO_POINT_TYPE(CUS_LONGT, CUS_LATI, NULL), NULL, NULL),
        SDO_GEOMETRY(2001,4326,SDO_POINT_TYPE(RET_LONGT, RET_LATI, NULL), NULL, NULL),
        0.01,
        'UNIT=KM')
    INTO DIST
    FROM DUAL;
    RETURN DIST;
END CALC_KMS;
```

Script Output | Task completed in 0.141 seconds

```
Function CALC_KMS compiled
```

5.2. The package should contain a function get_nearest(IDCUSTOMER, numberInList) that takes an Integer parameter as the IDCUSTOMER and returns a sys_refcursor containing a list of up to numberInList entries from the retail_location table that are nearest to that customer's current address (e.g., if numberInList = 1, then it gives the closest one only, if 2, then the closest 2, etc.) The returned records with the cursor should be sorted by increasing distance from the customer.

5.2.1. You can use the following code as a model for how to return a 'sys_refcursor'. Please note that this code returns an open cursor – it does not need to be opened.:

```
create or replace function get_employee
  (p_loc in number)
  return sys_refcursor
as
  ret_val sys_refcursor;
begin
  open ret_val
   for select a.first_name, a.last_name, b.department_name
         from employees a,
              departments b
        where a.department_id=b.department_id
          and location_id=p_loc;


  return ret_val;
end;
```

5.2.2. Provide a screenshot showing your get_nearest () implementation.

```
CREATE OR REPLACE FUNCTION GET_NEAREST (ID_CUS IN NUMBER, NUMBERINLIST IN NUMBER) RETURN SYS_REFCURSOR
AS
RET_VAL SYS_REFCURSOR;
BEGIN
OPEN RET_VAL FOR
SELECT IDCUSTOMER, IDRETAILLOCATION, CALC_KMS(IDCUSTOMER, IDRETAILLOCATION) AS NEAR_DISTANCE
FROM WUUSER.CUSTOMER, RETAILLOCATION WHERE IDCUSTOMER = ID_CUS
ORDER BY NEAR_DISTANCE ASC
FETCH FIRST NUMBERINLIST ROWS ONLY;
RETURN RET_VAL;
END GET_NEAREST;
```

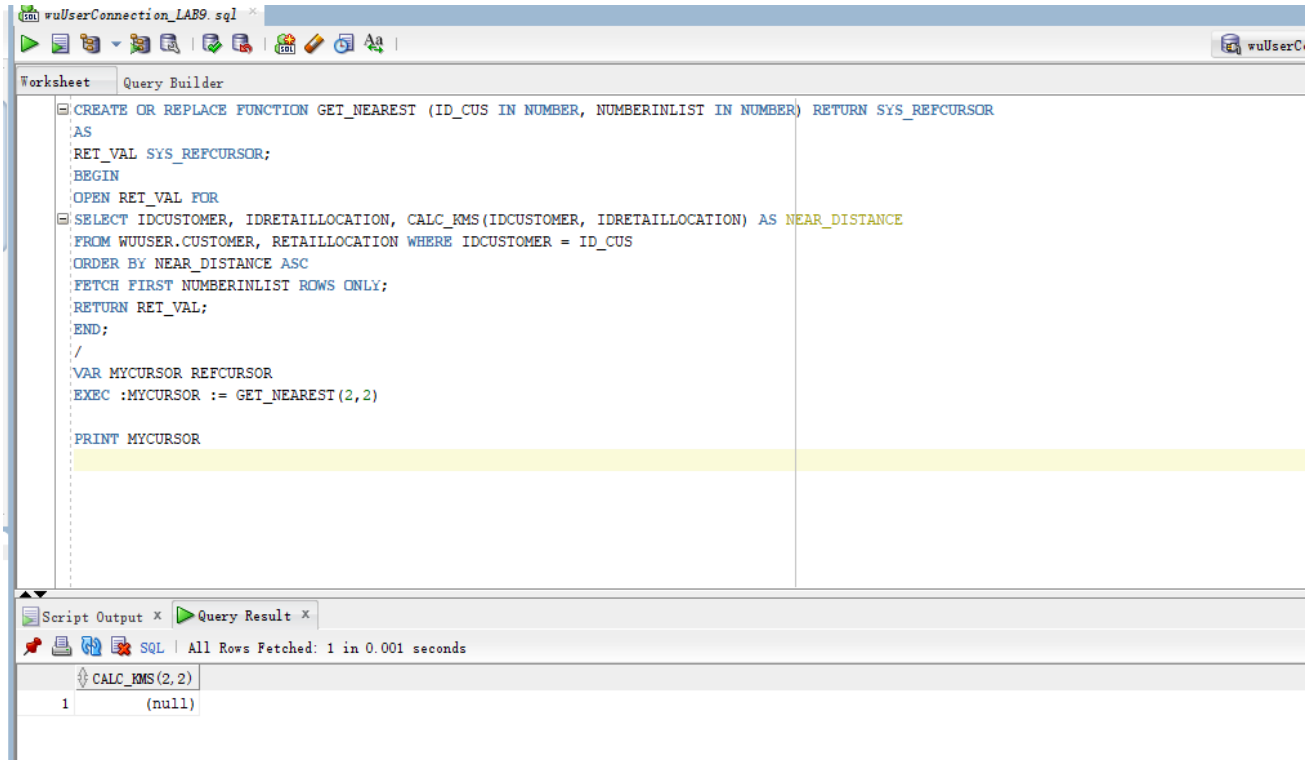
```
Script Output
Task completed in 0.029 seconds

Function GET_NEAREST compiled
```

5.3. Create an anonymous block (query script) using the "Query Builder" window in SQL Developer as follows:

5.3.1. The script should call your getNearest function for a particular IDCUSTOMER.and store the sys_refcursor in a variable.

5.3.2. After calling getNearest, the script should use the 'PRINT *mycursor'* command to print out the contents of the cursor in the SQL developer output pane.

5.3.3. Provide one or more screenshots showing an example of running your script in sql developer Make sure you clearly show how the IDCUSTOMER and numberInList parameters worked for your example.



6. Once you have embedded all of your screenshots, submit the file in Brightspace and you're done!