DBS311A Lab 5

1.

create or replace PROCEDURE Even\_Odd(v\_value1 NUMBER)

IS

division NUMBER := 0;

BEGIN

division := MOD(v\_value1, 2);

IF division = 0

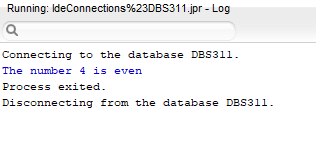
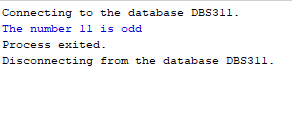
THEN DBMS\_OUTPUT.PUT\_LINE ('The number '|| v\_value1 ||' is even');

ELSE

DBMS\_OUTPUT.PUT\_LINE ('The number '|| v\_value1 ||' is odd');

END IF;

END Even\_Odd;

2.

create or replace PROCEDURE FIND\_EMPLOYEE(p\_empid NUMBER)

IS

firstName employees.first\_name%TYPE;

lastName employees.last\_name%TYPE;

email employees.email%TYPE;

phone employees.phone%TYPE;

hireDate VARCHAR2(255 BYTE);

jobTitle employees.job\_title%TYPE;

BEGIN

SELECT first\_Name, last\_Name, email, phone, TO\_CHAR(hire\_date, 'DD-MON-YY'), job\_title

INTO firstName, lastName, email, phone, hireDate, jobTitle

FROM employees WHERE employee\_id = p\_empid;

DBMS\_OUTPUT.PUT\_LINE('First Name: '||firstname);

DBMS\_OUTPUT.PUT\_LINE('Last Name: ' || lastName);

DBMS\_OUTPUT.PUT\_LINE('Email: '||email);

DBMS\_OUTPUT.PUT\_LINE('Phone: '||phone);

DBMS\_OUTPUT.PUT\_LINE('Hire date: '||hireDate);

DBMS\_OUTPUT.PUT\_LINE('Job Title: '||jobTitle);

EXCEPTION

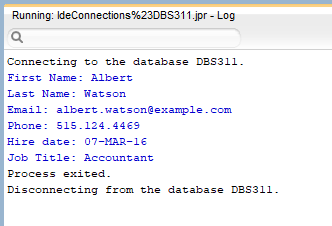
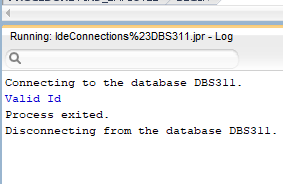
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE ('Valid Id');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('SOME ERROR HAPPENED');

END FIND\_EMPLOYEE;

3.

create or replace PROCEDURE UPDATE\_PRICE\_BY\_CAT (p\_catid IN NUMBER, p\_amount IN NUMBER)

IS

Rows\_updated NUMBER;

BEGIN

UPDATE PRODUCTS

SET list\_price = list\_price + p\_amount

WHERE category\_id=p\_catid AND list\_price > 0;

Rows\_updated := sql%rowcount;

IF rows\_updated=0

THEN DBMS\_OUTPUT.PUT\_LINE ('Invalid CatId');

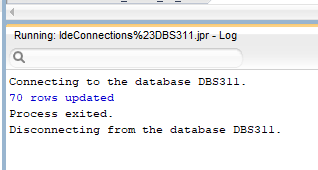
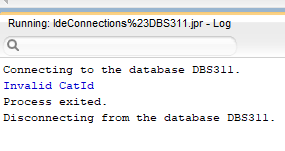
ELSE

DBMS\_OUTPUT.PUT\_LINE (rows\_updated ||' rows updated');

END IF;

ROLLBACK;

END UPDATE\_PRICE\_BY\_CAT;

4.

create or replace PROCEDURE UPDATE\_PRICE\_UNDER\_AVG

IS

avgPrice products.list\_price%TYPE;

Rows\_updated NUMBER;

BEGIN

SELECT AVG(list\_price) INTO avgPrice FROM products;

IF avgPrice <=1000

THEN

UPDATE Products SET list\_price = list\_price \* 1.02 WHERE list\_price < avgPrice;

ELSE

UPDATE Products SET list\_price = list\_price \* 1.01 WHERE list\_price < avgPrice;

END IF;

Rows\_updated := sql%rowcount;

DBMS\_OUTPUT.PUT\_LINE (rows\_updated ||' rows updated');

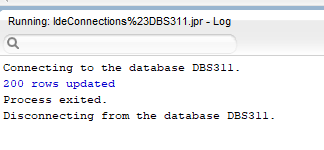
ROLLBACK;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('SOME ERROR HAPPENED');

END UPDATE\_PRICE\_UNDER\_AVG;



5.

create or replace PROCEDURE PRODUCT\_PRICE\_REPORT

IS

cheap\_count NUMBER;

fair\_count NUMBER;

exp\_count NUMBER;

avg\_price products.list\_price%TYPE;

min\_price products.list\_price%TYPE;

max\_price products.list\_price%TYPE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE ('Category 1');

SELECT AVG(list\_price), MIN(list\_price), MAX(list\_price) INTO avg\_price, min\_price, max\_price FROM products WHERE category\_id = 1;

SELECT COUNT(\*) INTO cheap\_count FROM products WHERE list\_price < ((avg\_price - min\_price) / 2) AND category\_id=1;

SELECT COUNT(\*) INTO exp\_count FROM products WHERE list\_price > ((max\_price - avg\_price) / 2) AND category\_id=1;

SELECT COUNT(\*) INTO fair\_count FROM products WHERE (list\_price > (avg\_price - min\_price) / 2) AND list\_price < ((max\_price - avg\_price) / 2) AND category\_id=1;

DBMS\_OUTPUT.PUT\_LINE ('Cheap: '|| cheap\_count);

DBMS\_OUTPUT.PUT\_LINE ('Fair: '|| fair\_count);

DBMS\_OUTPUT.PUT\_LINE ('Expensive: '|| exp\_count);

DBMS\_OUTPUT.PUT\_LINE ('Category 4');

SELECT AVG(list\_price), MIN(list\_price), MAX(list\_price) INTO avg\_price, min\_price, max\_price FROM products WHERE category\_id = 4;

SELECT COUNT(\*) INTO cheap\_count FROM products WHERE list\_price < ((avg\_price - min\_price) / 2) AND category\_id=4;

SELECT COUNT(\*) INTO exp\_count FROM products WHERE list\_price > ((max\_price - avg\_price) / 2) AND category\_id=4;

SELECT COUNT(\*) INTO fair\_count FROM products WHERE (list\_price > (avg\_price - min\_price) / 2) AND list\_price < ((max\_price - avg\_price) / 2) AND category\_id=4;

DBMS\_OUTPUT.PUT\_LINE ('Cheap: '|| cheap\_count);

DBMS\_OUTPUT.PUT\_LINE ('Fair: '|| fair\_count);

DBMS\_OUTPUT.PUT\_LINE ('Expensive: '|| exp\_count);

DBMS\_OUTPUT.PUT\_LINE ('Category 5');

SELECT AVG(list\_price), MIN(list\_price), MAX(list\_price) INTO avg\_price, min\_price, max\_price FROM products WHERE category\_id = 5;

SELECT COUNT(\*) INTO cheap\_count FROM products WHERE list\_price < ((avg\_price - min\_price) / 2) AND category\_id=5;

SELECT COUNT(\*) INTO exp\_count FROM products WHERE list\_price > ((max\_price - avg\_price) / 2) AND category\_id=5;

SELECT COUNT(\*) INTO fair\_count FROM products WHERE (list\_price > (avg\_price - min\_price) / 2) AND list\_price < ((max\_price - avg\_price) / 2) AND category\_id=5;

DBMS\_OUTPUT.PUT\_LINE ('Cheap: '|| cheap\_count);

DBMS\_OUTPUT.PUT\_LINE ('Fair: '|| fair\_count);

DBMS\_OUTPUT.PUT\_LINE ('Expensive: '|| exp\_count);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('SOME ERROR HAPPENED');

END PRODUCT\_PRICE\_REPORT;

