-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
-- Name: Chi Ming Lai  
-- Student ID: 158400200  
-- Date: 05 July 2022  
-- Purpose: Lab 5 DBS311  
-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 1----Write a stored procedure to define a num is even or odd**

CREATE OR REPLACE PROCEDURE oddEven(num IN number)

as

BEGIN

if mod(num, 2) = 0

then dbms\_output.put\_line('The number is even');

else

dbms\_output.put\_line('The number is odd');

end if;

END oddEven;

BEGIN

oddEven('124');

END;

**Question 2----display the employee that matches the passed employee’s ID**

CREATE OR REPLACE PROCEDURE find\_me (empID IN number)

AS

fName VARCHAR2(20 BYTE);

lName VARCHAR2(25 BYTE);

emails VARCHAR2(25 BYTE);

phone VARCHAR2(20 BYTE);

hireDate VARCHAR2(30 BYTE);

jobID VARCHAR2(10 BYTE);

BEGIN

SELECT first\_name, last\_name, email, phone\_number, TO\_CHAR(hire\_date, 'DD-MON-YY'), job\_id

INTO fName, lName, emails, phone, hireDate, jobID

FROM newEmployees

WHERE employee\_id = empID;

dbms\_output.put\_line('First name: ' || fName);

dbms\_output.put\_line('Last name: ' || lName);

dbms\_output.put\_line('Email: ' || emails);

dbms\_output.put\_line('phone: ' || phone);

dbms\_output.put\_line('Hire date: ' || hireDate);

dbms\_output.put\_line('Job title: ' || jobID);

EXCEPTION

WHEN NO\_DATA\_FOUND

THEN

dbms\_output.put\_line('No employees found');

END;

BEGIN

find\_me(&input);

END;

**Question 3----update prod\_sell by product\_type**

CREATE OR REPLACE PROCEDURE update\_price\_fortype(a\_prod\_type IN VARCHAR2, amount IN NUMBER)

AS

Rows\_updated NUMBER;

BEGIN

UPDATE newProducts

SET prod\_sell = prod\_sell + amount

WHERE UPPER(prod\_type) LIKE UPPER(a\_prod\_type)

AND prod\_sell > 0;

Rows\_updated := sql%rowcount;

dbms\_output.put\_line(Rows\_updated || ' rows updated!');

EXCEPTION

WHEN OTHERS

THEN

dbms\_output.put\_line('ERROR');

END;

BEGIN

update\_price\_fortype('Tents', 5);

END;

**Question 4----update prod\_sell if the prod\_sell is lower than average**

CREATE OR REPLACE PROCEDURE update\_low\_prices\_15

AS

average NUMBER;

Rows\_updated NUMBER;

BEGIN

SELECT ROUND(AVG(prod\_sell), 2)

INTO average

FROM newProducts;

IF average <= 1000

THEN UPDATE newProducts

SET prod\_sell = prod\_sell \* 1.02

WHERE prod\_sell < average;

ELSIF average > 1000

THEN UPDATE newProducts

SET prod\_sell = prod\_sell \* 1.01

WHERE prod\_sell < average;

END IF;

Rows\_updated := sql%rowcount;

dbms\_output.put\_line('\*\*\* OUTPUT update\_low\_prices\_99 STARTED \*\*\*');

dbms\_output.put\_line('Number of updates: ' || Rows\_updated);

dbms\_output.put\_line('----ENDED --------');

EXCEPTION

WHEN OTHERS

THEN

dbms\_output.put\_line('ERROR');

END;

BEGIN

update\_low\_prices\_15;

END;

**Question 5----Display the number of each product price category**

CREATE OR REPLACE PROCEDURE price\_report\_15

AS

avg\_price NUMBER;

min\_price NUMBER;

max\_price NUMBER;

low\_count NUMBER := 0;

fair\_count NUMBER := 0;

high\_count NUMBER := 0;

BEGIN

SELECT ROUND(AVG(prod\_sell), 2), MIN(prod\_sell), MAX(prod\_sell)

INTO avg\_price, min\_price, max\_price

FROM newProducts;

SELECT COUNT(prod\_sell)

INTO low\_count

FROM newProducts

WHERE prod\_sell < (avg\_price - min\_price) / 2;

SELECT COUNT(prod\_sell)

INTO high\_count

FROM newProducts

WHERE prod\_sell > (max\_price - avg\_price) / 2;

SELECT COUNT(prod\_sell)

INTO fair\_count

FROM newProducts

WHERE prod\_sell >= (avg\_price - min\_price) / 2

AND prod\_sell <= (max\_price - avg\_price) / 2;

dbms\_output.put\_line('Low: ' || low\_count);

dbms\_output.put\_line('Fair: ' || fair\_count);

dbms\_output.put\_line('High: ' || high\_count);

EXCEPTION

WHEN OTHERS

THEN

dbms\_output.put\_line('ERROR');

END;

BEGIN

price\_report\_15;

END;