-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Name: Prince Jodhani

-- Student ID: 149455206

-- Date: 10-03-2022

-- Purpose: Lab 5 DBS311

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**-- Question 1 –** Write a stored procedure that gets an integer number and prints The number is divisible by 100. If a number is divisible by 100. Otherwise, it prints The number is not divisible by 100.

**-- Q1 SOLUTION –**

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE divby (num1 IN NUMBER) AS

divable NUMBER;

BEGIN

divable := num1 mod 100;

IF divable = 0

THEN

DBMS\_OUTPUT.PUT\_LINE ('The number is divisible by 100.');

ELSE

DBMS\_OUTPUT.PUT\_LINE ('The number is not divisible by 100');

END IF;

EXCEPTION

WHEN OTHERS

THEN

DBMS\_OUTPUT.PUT\_LINE ('Error!');

END divby;

* I entered 200

Graphical user interface, text, application

Description automatically generated

**-- Question 2 –** Create a stored procedure named find\_employee. This procedure gets an employee number and prints the following employee information: First name Last name Email Phone Hire date

**-- Q2 SOLUTION –**

create or replace PROCEDURE find\_employee (num1 IN NUMBER) AS

m\_first VARCHAR2(50);

m\_last VARCHAR2(50);

m\_email VARCHAR2(50);

m\_phone VARCHAR2(50);

m\_hiredate DATE;

m\_jobid VARCHAR2(50);

BEGIN

SELECT first\_name,last\_name,email,phone,hire\_date,job\_title

INTO m\_first,m\_last,m\_email,m\_phone,m\_hiredate,m\_jobid

FROM employees WHERE employee\_id = num1;

DBMS\_OUTPUT.PUT\_LINE ('First name: ' || m\_first);

DBMS\_OUTPUT.PUT\_LINE ('Last name: '|| m\_first);

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

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

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

DBMS\_OUTPUT.PUT\_LINE ('Job title: '|| m\_jobid);

EXCEPTION

WHEN OTHERS

THEN

DBMS\_OUTPUT.PUT\_LINE ('Error!');

END find\_employee;

* I entered 4

Graphical user interface, text, application

Description automatically generated

**-- Question 3 –** Every year, the company increases the price of all products in one product type. For example, the company wants to increase the list price of products in category type 1 by $5. Write a procedure named update\_price\_category to update the price of all products in the given type and the given amount to be added to the current list price if the price is greater than 0.

**-- Q3 SOLUTION –**

DROP TABLE new\_products;

CREATE TABLE new\_products AS (SELECT \* FROM products);

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE update\_price\_category (ptype IN products.category\_id%TYPE, amount IN products.standard\_cost%TYPE)

AS Rows\_updated NUMBER;

BEGIN

UPDATE new\_products SET standard\_cost = standard\_cost + amount WHERE category\_id = ptype;

IF sql%rowcount = 0

THEN

DBMS\_OUTPUT.PUT\_LINE('No data');

ELSE

Rows\_updated := sql%rowcount;

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

END IF;

EXCEPTION

WHEN OTHERS

THEN

DBMS\_OUTPUT.PUT\_LINE ('Error!');

END update\_price\_category;

Text

Description automatically generated

**-- Question 4 –** Every year, the company increases the list\_price of products by 1 or 2% (Example of 2% -- prod\_sell \* 1.02) based on if the selling price (prod\_sell) is less than the average price of all products. Write a stored procedure named update\_low\_prices\_123456789 where 123456789 is replaced by your student number.

**-- Q4 SOLUTION –**

DROP TABLE new\_products;

CREATE TABLE new\_products AS (SELECT \* FROM products);

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE update\_low\_prices\_149455206

AS Rows\_updated NUMBER; average NUMBER(9,2); added2 NUMBER(9,2); added1 NUMBER(9,2);

BEGIN

SELECT AVG(list\_price) INTO average FROM new\_products;

added2 := (average \* 2) / 100;

added1 := (average \* 1) / 100;

IF average <= 1000 THEN

UPDATE new\_products SET list\_price = list\_price + added2;

ELSIF average > 1000 THEN

UPDATE new\_products SET list\_price = list\_price + added1;

END IF;

Rows\_updated := sql%rowcount;

DBMS\_OUTPUT.PUT\_LINE('\*\*\* OUTPUT update\_low\_prices\_149455206 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 update\_low\_prices\_149455206;

Graphical user interface, text

Description automatically generated

**-- Question 5 –** The company needs a report that shows three categories of products based their prices. The company needs to know if the product price is cheap, fair, or expensive.

**-- Q5 SOLUTION –**

DROP TABLE new\_products;

CREATE TABLE new\_products AS (SELECT \* FROM products);

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE price\_report\_149455206

AS Rows\_updated NUMBER;m\_price1 NUMBER(9,2); m\_low NUMBER(9,2); m\_fair NUMBER(9,2); m\_high NUMBER(9,2); m\_avg NUMBER(9,2); m\_price NUMBER(9,2);

BEGIN

SELECT AVG(list\_price), MIN(list\_price), MAX(list\_price) INTO m\_avg,m\_low,m\_high FROM new\_products;

m\_price := (m\_avg - m\_low) / 2;

IF m\_avg <= m\_price THEN

DBMS\_OUTPUT.PUT\_LINE('Low: ' || m\_price);

ELSIF m\_avg > m\_high THEN

DBMS\_OUTPUT.PUT\_LINE('High: ' || m\_price);

ELSIF m\_avg - m\_low / 2 AND m\_high - m\_avg / 2 THEN

DBMS\_OUTPUT.PUT\_LINE('Fair: ' || m\_price);

END IF;

EXCEPTION

WHEN OTHERS

THEN

DBMS\_OUTPUT.PUT\_LINE ('Error!');

END price\_report\_149455206 ;