**--Oracle Assignments**

**Business Requirement / User Story :** As a eCommerce  company we should be able to store & query customer-order information using oracle database package.

* Please create Oracle Database tables & package-

Package should contain below procedures/functions.

1. InsertCustomer – Create new customer. Return Customer id.
2. GetCustomerDetails – Input Customer id, Return customer record.
3. UpdateCustomer – Input Customer id , Name, Birth Date, Address.  Return success/failure. Return proper error message id customer not found.
4. SearchCustomer – Input Customer Name. Return matching customer id’s as PLSQL Index by Table (array).
5. CreateOrder – Input Customer Id, Product Id. Insert row in order details. Return success/failure .
6. GetOrderByDate – Input Order Date, Return cursor with Customers Name , Product Name & Oder id for order created in given date

* Create product, customer, and order tables with required columns.
* Define primary keys, foreign keys & not null columns.
* Define Exception handling & Error logging.
* Create anonymous PLSQL scripts to demonstrate call to above procedure/functions with sample data.

***--Solution of the given statement:***

--Creating Tables for Customer, Product & Order\_Details

CREATE TABLE Customer (

CustomerId NUMBER PRIMARY KEY NOT NULL,

Name VARCHAR2(50) NOT NULL,

Phone Number NOT NULL,

Address VARCHAR2(500) NOT NULL

);

CREATE TABLE Product (

ProductId NUMBER PRIMARY KEY,

ProductName VARCHAR2(100) NOT NULL

);

CREATE TABLE OrderDetails (

OrderId NUMBER PRIMARY KEY,

CustomerId NUMBER,

ProductId NUMBER,

OrderDate DATE,

CONSTRAINT fk\_customer FOREIGN KEY (CustomerId) REFERENCES Customer(CustomerId),

CONSTRAINT fk\_product FOREIGN KEY (ProductId) REFERENCES Product (ProductId)

);

--SEQUENCE

CREATE SEQUENCE SEQ1;

--PACKAGE (INSERTING CUSTOMER)

CREATE OR REPLACE PACKAGE Business AS

FUNCTION Insert Customer ( c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN int;

END Business;

/

--PACKAGE BODY (INSERTING CUSTOMER)

CREATE OR REPLACE PACKAGE BODY Business AS

FUNCTION InsertCustomer ( c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN int IS

BEGIN

insert into customer(CustomerId, name, Phone, address)values(seq1.nextval, c\_Name,c\_Phone,c\_Address);

RETURN seq1.currval;

END InsertCustomer;

END Business;

--Calling InsertCustomer

DECLARE

v\_CustomerId1 INT;

v\_CustomerId2 INT;

v\_CustomerId3 INT;

BEGIN

v\_CustomerId1 := Business.InsertCustomer('Daniel Craig', 6492301, 'NewYork');

v\_CustomerId2 := Business.InsertCustomer('Emma Watson', 6492302, 'LosAngeles');

v\_CustomerId3 := Business.InsertCustomer('John Doe', 6492303, 'Chicago');

DBMS\_OUTPUT.PUT\_LINE('New Customer Id 1: ' || TO\_CHAR(v\_CustomerId1));

DBMS\_OUTPUT.PUT\_LINE('New Customer Id 2: ' || TO\_CHAR(v\_CustomerId2));

DBMS\_OUTPUT.PUT\_LINE('New Customer Id 3: ' || TO\_CHAR(v\_CustomerId3));

END;

/

SELECT \* FROM CUSTOMER

--GetCustomerDetails

-- PACKAGE

CREATE OR REPLACE PACKAGE Business AS

FUNCTION InsertCustomer(c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN NUMBER;

FUNCTION GetCustomerDetails(c\_CustomerId Customer.CustomerId%Type) RETURN Customer%ROWTYPE;

END Business;

/

-- PACKAGE BODY

CREATE OR REPLACE PACKAGE BODY Business AS

FUNCTION InsertCustomer(c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN NUMBER IS

BEGIN

INSERT INTO customer(CustomerId, name, Phone, address) VALUES (seq1.nextval, c\_Name, c\_Phone, c\_Address);

RETURN seq1.currval;

END InsertCustomer;

FUNCTION GetCustomerDetails(c\_CustomerId Customer.CustomerId%Type) RETURN Customer%ROWTYPE IS

v\_CustomerDetails Customer%ROWTYPE;

BEGIN

SELECT \* INTO v\_CustomerDetails FROM Customer WHERE CustomerId = c\_CustomerId;

RETURN v\_CustomerDetails;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Customer with ID ' || c\_CustomerId || ' Data Not Found.');

RETURN NULL;

END GetCustomerDetails;

END Business;

/

--Call Customer Details

DECLARE

v\_CustomerId NUMBER := 2;

v\_CustomerDetails Customer%ROWTYPE;

BEGIN

v\_CustomerDetails := Business.GetCustomerDetails(v\_CustomerId);

IF v\_CustomerDetails.CustomerId IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Customer Details for ID ' || v\_CustomerId || ':');

DBMS\_OUTPUT.PUT\_LINE('Customer Name: ' || v\_CustomerDetails.Name);

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

DBMS\_OUTPUT.PUT\_LINE('Address: ' || v\_CustomerDetails.Address);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Customer with ID ' || v\_CustomerId || ' Data Not Found.');

END IF;

END;

/

-- Updating Customer

-- Package for Updating Customer

CREATE OR REPLACE PACKAGE Business AS

FUNCTION InsertCustomer(c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN NUMBER;

FUNCTION GetCustomerDetails(c\_CustomerId Customer.CustomerId%Type) RETURN Customer%ROWTYPE;

FUNCTION UpdateCustomer(c\_CustomerId Customer.CustomerId%Type, c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN VARCHAR2;

END Business;

/

--Package Body for Updating Customer

CREATE OR REPLACE PACKAGE BODY Business AS

FUNCTION InsertCustomer(c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN NUMBER IS

BEGIN

INSERT INTO customer(CustomerId, name, Phone, address) VALUES (seq1.nextval, c\_Name, c\_Phone, c\_Address);

RETURN seq1.currval;

END InsertCustomer;

FUNCTION GetCustomerDetails(c\_CustomerId Customer.CustomerId%Type) RETURN Customer%ROWTYPE IS

v\_CustomerDetails Customer%ROWTYPE;

BEGIN

SELECT \* INTO v\_CustomerDetails FROM Customer WHERE CustomerId = c\_CustomerId;

RETURN v\_CustomerDetails;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Customer with ID ' || c\_CustomerId || ' Data Not Found.');

RETURN NULL;

END GetCustomerDetails;

FUNCTION UpdateCustomer(c\_CustomerId Customer.CustomerId%Type, c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN VARCHAR2 IS

BEGIN

UPDATE Customer

SET Name = c\_Name, Phone = c\_Phone, Address = c\_Address

WHERE CustomerId = c\_CustomerId;

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Customer with ID ' || c\_CustomerId || ' Not Found. Update Failed.');

RETURN 'FAILURE: Customer not found';

ELSE

RETURN 'SUCCESS: Customer updated successfully';

END IF;

END UpdateCustomer;

END Business;

/

--Calling To Update Customer

DECLARE

v\_UpdateResult VARCHAR2(100);

BEGIN

v\_UpdateResult := Business.UpdateCustomer(3, 'Peter Parker', 4301837, 'Mexico');

DBMS\_OUTPUT.PUT\_LINE(v\_UpdateResult);

END;

/

SELECT \* FROM CUSTOMER

--Search Customer

--PACKAGE For Searching Customers

CREATE OR REPLACE PACKAGE Business AS

FUNCTION InsertCustomer(c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN NUMBER;

FUNCTION GetCustomerDetails(c\_CustomerId Customer.CustomerId%Type) RETURN Customer%ROWTYPE;

FUNCTION UpdateCustomer(c\_CustomerId Customer.CustomerId%Type, c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN VARCHAR2;

FUNCTION SearchCustomer(c\_Name Customer.Name%Type) RETURN SYS\_REFCURSOR;

END Business;

/

--Package Body of Search\_Customer

CREATE OR REPLACE PACKAGE BODY Business AS

FUNCTION InsertCustomer(c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN NUMBER IS

BEGIN

INSERT INTO customer(CustomerId, name, Phone, address) VALUES (seq1.nextval, c\_Name, c\_Phone, c\_Address);

RETURN seq1.currval;

END InsertCustomer;

FUNCTION GetCustomerDetails(c\_CustomerId Customer.CustomerId%Type) RETURN Customer%ROWTYPE IS

v\_CustomerDetails Customer%ROWTYPE;

BEGIN

SELECT \* INTO v\_CustomerDetails FROM Customer WHERE CustomerId = c\_CustomerId;

RETURN v\_CustomerDetails;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Customer with ID ' || c\_CustomerId || ' Data Not Found.');

RETURN v\_CustomerDetails; -- or NULL if you prefer

END GetCustomerDetails;

FUNCTION UpdateCustomer(c\_CustomerId Customer.CustomerId%Type, c\_Name Customer.Name%Type, c\_Phone Customer.Phone%Type, c\_Address Customer.Address%Type) RETURN VARCHAR2 IS

BEGIN

UPDATE Customer

SET Name = c\_Name, Phone = c\_Phone, Address = c\_Address

WHERE CustomerId = c\_CustomerId;

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Customer with ID ' || c\_CustomerId || ' Not Found. Update Failed.');

RETURN 'FAILURE: Customer not found';

ELSE

RETURN 'SUCCESS: Customer updated successfully';

END IF;

END UpdateCustomer;

FUNCTION SearchCustomer(c\_Name Customer.Name%Type) RETURN SYS\_REFCURSOR IS

v\_Cursor SYS\_REFCURSOR;

BEGIN

OPEN v\_Cursor FOR

SELECT CustomerId

FROM Customer

WHERE UPPER(Name) LIKE '%' || UPPER(c\_Name) || '%';

RETURN v\_Cursor;

END SearchCustomer;

END Business;

/

--Search\_Customer function using a PL/SQL block

DECLARE

v\_CustomerName VARCHAR2(50) := 'PETER';

v\_Cursor SYS\_REFCURSOR;

v\_CustomerId NUMBER;

BEGIN

v\_Cursor := Business.SearchCustomer(v\_CustomerName);

LOOP

FETCH v\_Cursor INTO v\_CustomerId;

EXIT WHEN v\_Cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Matching Customer ID: ' || TO\_CHAR(v\_CustomerId));

END LOOP;

CLOSE v\_Cursor;

END;

/

SELECT \* FROM CUSTOMER

--Creating Order

-- PACKAGE

CREATE OR REPLACE PACKAGE Business AS

FUNCTION CreateOrder(c\_CustomerId Customer.CustomerId%Type, c\_ProductId Product.ProductId%Type) RETURN VARCHAR2;

END Business;

/

-- PACKAGE BODY

CREATE OR REPLACE PACKAGE BODY Business AS

FUNCTION CreateOrder(c\_CustomerId Customer.CustomerId%Type, c\_ProductId Product.ProductId%Type) RETURN VARCHAR2 IS

BEGIN

DECLARE

v\_CustomerExists NUMBER;

v\_ProductExists NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_CustomerExists FROM Customer WHERE CustomerId = c\_CustomerId;

SELECT COUNT(\*) INTO v\_ProductExists FROM Product WHERE ProductId = c\_ProductId;

IF v\_CustomerExists = 0 THEN

RETURN 'FAILURE: Customer with ID ' || c\_CustomerId || ' not found.';

ELSIF v\_ProductExists = 0 THEN

RETURN 'FAILURE: Product with ID ' || c\_ProductId || ' not found.';

END IF;

END;

-- Insert the order details

BEGIN

INSERT INTO OrderDetails(OrderId, CustomerId, ProductId, OrderDate)

VALUES (SEQ1.NEXTVAL, c\_CustomerId, c\_ProductId, SYSDATE);

EXCEPTION

WHEN OTHERS THEN

RETURN 'FAILURE: Error inserting order details - ' || SQLERRM;

END;

RETURN 'SUCCESS: Order created successfully.';

END CreateOrder;

END Business;

/

-- CALLING CreateOrder

DECLARE

v\_CustomerId NUMBER := 1;

v\_ProductId NUMBER := 1;

v\_Result VARCHAR2(100);

BEGIN

v\_Result := Business.CreateOrder(v\_CustomerId, v\_ProductId);

DBMS\_OUTPUT.PUT\_LINE(v\_Result);

END;

/

--GetOrderByDate

-- PACKAGE

CREATE OR REPLACE PACKAGE Business AS

FUNCTION GetOrderByDate(c\_OrderDate DATE) RETURN SYS\_REFCURSOR;

END Business;

/

-- PACKAGE BODY for GetOrder

CREATE OR REPLACE PACKAGE BODY Business AS

FUNCTION GetOrderByDate(c\_OrderDate DATE) RETURN SYS\_REFCURSOR IS

v\_Cursor SYS\_REFCURSOR;

BEGIN

OPEN v\_Cursor FOR

SELECT c.Name AS CustomerName, p.ProductName, od.OrderId

FROM OrderDetails od

JOIN Customer c ON od.CustomerId = c.CustomerId

JOIN Product p ON od.ProductId = p.ProductId

WHERE TRUNC(od.OrderDate) = TRUNC(c\_OrderDate);

RETURN v\_Cursor;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No orders found for the given date: ' || TO\_CHAR(c\_OrderDate));

RETURN NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error getting orders by date - ' || SQLERRM);

RETURN NULL;

END GetOrderByDate;

END Business;

/

-- CALLING GetOrderByDate

DECLARE

v\_OrderDate DATE := TO\_DATE('2024-01-02', 'YYYY-MM-DD');

v\_ResultCursor SYS\_REFCURSOR;

v\_CustomerName VARCHAR2(50);

v\_ProductName VARCHAR2(100);

v\_OrderId NUMBER;

BEGIN

v\_ResultCursor := Business.GetOrderByDate(v\_OrderDate);

LOOP

FETCH v\_ResultCursor INTO v\_CustomerName, v\_ProductName, v\_OrderId;

EXIT WHEN v\_ResultCursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Customer Name: ' || v\_CustomerName || ', Product Name: ' || v\_ProductName || ', Order ID: ' || TO\_CHAR(v\_OrderId));

END LOOP;

CLOSE v\_ResultCursor;

END;

/

SELECT \* FROM OrderDetails

SELECT \* FROM CUSTOMER