

## **WEEK 2 – PL/SQL PROGRAMMING**

### **Exercise 3: Stored Procedures**

#### **TABLE CREATION:**

##### **savings\_accounts:**

```
CREATE TABLE savings_accounts (  
    account_id NUMBER PRIMARY KEY,  
    customer_name VARCHAR2(100),  
    balance NUMBER(10,2)  
);
```

##### **Employees:**

```
CREATE TABLE employees (  
    emp_id NUMBER PRIMARY KEY,  
    emp_name VARCHAR2(100),  
    department VARCHAR2(50),  
    salary NUMBER(10,2)  
);
```

##### **bank\_accounts:**

```
CREATE TABLE bank_accounts (  
    account_id NUMBER PRIMARY KEY,  
    customer_name VARCHAR2(100),  
    balance NUMBER(10,2)  
);
```

#### **DATA INSERTION:**

```
INSERT INTO savings_accounts VALUES (1, 'Alice', 10000);
```

```
INSERT INTO savings_accounts VALUES (2, 'Bob', 15000);
```

```
INSERT INTO employees VALUES (101, 'John', 'HR', 30000);
```

```
INSERT INTO employees VALUES (102, 'Jane', 'HR', 32000);
```

```
INSERT INTO employees VALUES (103, 'Mark', 'IT', 40000);
```

```
INSERT INTO bank_accounts VALUES (201, 'Alice', 5000);
INSERT INTO bank_accounts VALUES (202, 'Bob', 7000);
COMMIT;
```

## **SCENARIO 1:**

The bank needs to process monthly interest for all savings accounts.

- Question: Write a stored procedure  
ProcessMonthlyInterest that calculates and updates the  
balance of all savings accounts by applying an interest rate  
of 1% to the current balance.

## **CODE:**

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
```

```
BEGIN
```

```
    UPDATE savings_accounts
```

```
    SET balance = balance + (balance * 0.01);
```

```
    COMMIT;
```

```
END;
```

## **OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, displaying a SQL command: `SELECT * FROM savings_accounts;`. The 'Run' button is highlighted in green. Below the command editor, the 'Results' tab is selected, showing a table with three columns: 'ACCOUNT\_ID', 'CUSTOMER\_NAME', and 'BALANCE'. The table contains three rows of data.

ACCOUNT_ID	CUSTOMER_NAME	BALANCE
1	Asha	10000
2	Sheela	15000
3	Tamil	20000

## **SCENARIO 2:**

Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.

- Question: Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

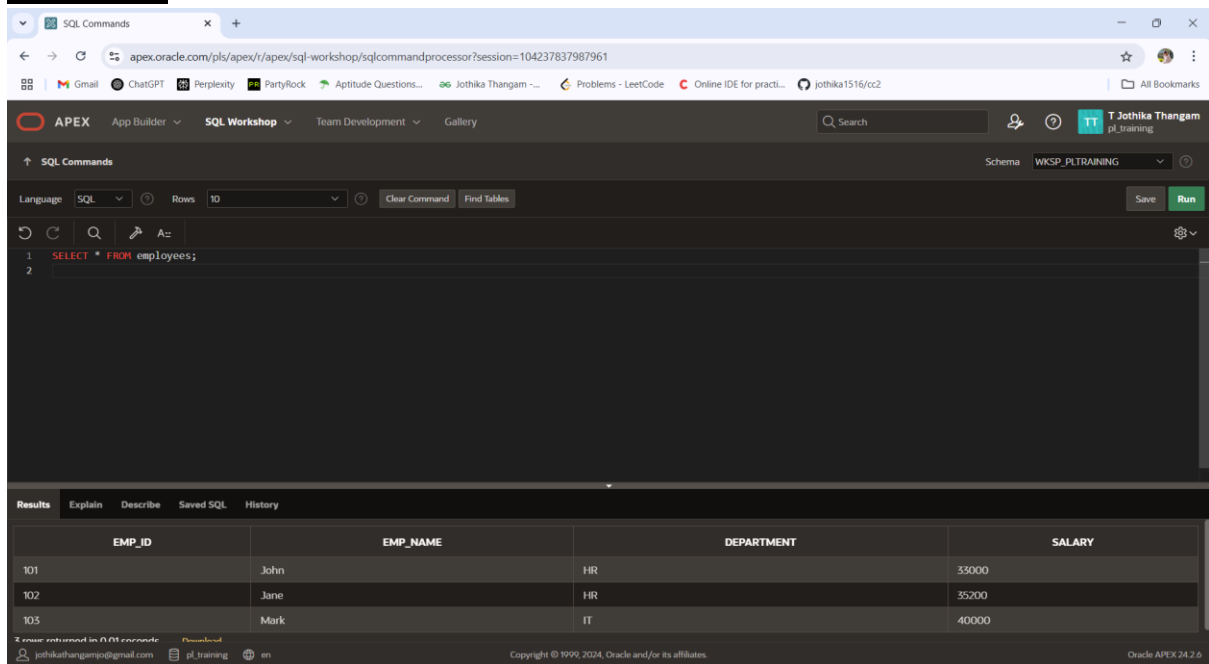
## **CODE:**

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (  
    dept_name IN VARCHAR2,  
    bonus_pct IN NUMBER  
) AS  
BEGIN  
    UPDATE employees  
    SET salary = salary + (salary * bonus_pct / 100)  
    WHERE department = dept_name;  
    COMMIT;  
END;
```

## **STATEMENT:**

```
BEGIN  
    UpdateEmployeeBonus('HR', 10);  
END;
```

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The main area displays the 'SQL Commands' tab with a schema dropdown set to 'WKSP\_PLTRAINING'. The SQL editor contains the query: `SELECT * FROM employees;`. Below the editor, the 'Results' tab is active, showing a table with 3 rows and 4 columns: EMP\_ID, EMP\_NAME, DEPARTMENT, and SALARY. The data rows are: (101, John, HR, 33000), (102, Jane, HR, 35200), and (103, Mark, IT, 40000). The bottom status bar indicates 'Oracle APEX 24.2.6'.

EMP_ID	EMP_NAME	DEPARTMENT	SALARY
101	John	HR	33000
102	Jane	HR	35200
103	Mark	IT	40000

## SCENARIO 3:

Scenario 3: Customers should be able to transfer funds between their accounts.

- Question: Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

## CODE:

```
CREATE OR REPLACE PROCEDURE TransferFunds (  
    from_account IN NUMBER,  
    to_account IN NUMBER,  
    amount IN NUMBER  
) AS  
    from_balance NUMBER;  
BEGIN
```

```
SELECT balance INTO from_balance FROM bank_accounts WHERE  
account_id = from_account;
```

```
IF from_balance < amount THEN
```

```
    RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds in source  
account.');
```

```
END IF;
```

```
UPDATE bank_accounts SET balance = balance - amount WHERE  
account_id = from_account;
```

```
UPDATE bank_accounts SET balance = balance + amount WHERE  
account_id = to_account;
```

```
COMMIT;
```

```
END;
```

## **STATEMENT:**

```
BEGIN
```

```
TRANSFERFUNDS(201,202,1000);
```

```
END;
```

## **OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is `SELECT * FROM bank_accounts;`. The results are displayed in a table with the following data:

ACCOUNT_ID	CUSTOMER_NAME	BALANCE
202	Sheela	8000
201	Asha	4000

2 rows returned in 0.00 seconds

