Week 2 – PL/SQL PROGRAMMING

Exercise 3:Stored procedure

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, ‘Shalini', 1000.00);
INSERT INTO savings_accounts VALUES (2, 'varshini', 2000.00);
INSERT INTO savings_accounts VALUES (3, 'Charu', 3000.00);
```

```
INSERT INTO employees VALUES (101, 'Emma', 'SALES', 5000.00);
INSERT INTO employees VALUES (102, 'Janani', 'HR', 4500.00);
INSERT INTO employees VALUES (103, 'Diya', 'SALES', 4800.00);
INSERT INTO bank_accounts VALUES (201, 'shalini', 2500.00);
INSERT INTO bank_accounts VALUES (202, 'varshini', 1800.00);
```

Scenario 1:

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

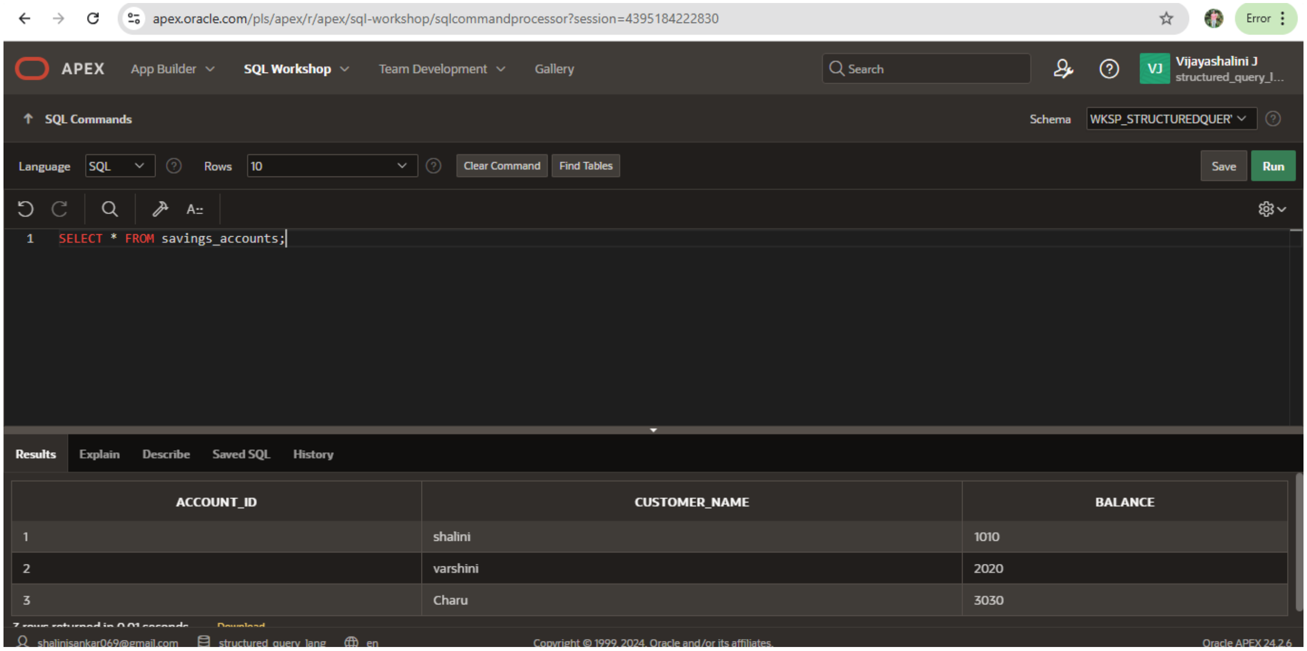
Question: Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
  UPDATE savings_accounts
  SET balance = balance * 1.01;

  COMMIT;
END;
```

Statement:

```
BEGIN
  ProcessMonthlyInterest;
END;
```

OUTPUT:



```
SELECT * FROM savings_accounts;
```

ACCOUNT_ID	CUSTOMER_NAME	BALANCE
1	shalini	1010
2	varshini	2020
3	Charu	3030

3 rows returned in 0.01 seconds

SCENARIO 2:

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

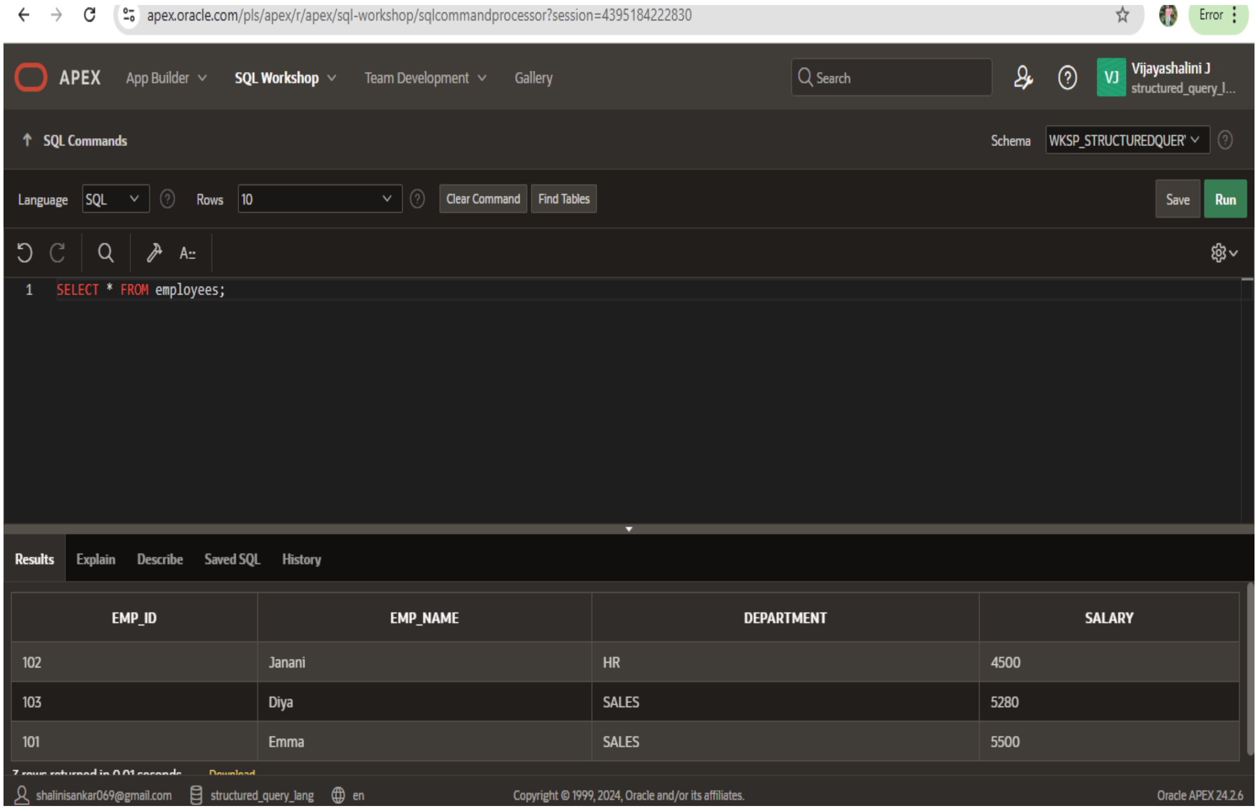
Question: Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(
  p_department IN VARCHAR2,
  p_bonus_percent IN NUMBER
) IS
BEGIN
  UPDATE employees
  SET salary = salary + (salary * p_bonus_percent / 100)
  WHERE department = p_department;


  COMMIT;
END;
```

CALL IT:

```
BEGIN
  UpdateEmployeeBonus('SALES', 10);  -- 10% bonus to SALES department
END;
```

OUTPUT:



```
SELECT * FROM employees;
```

EMP_ID	EMP_NAME	DEPARTMENT	SALARY
102	Janani	HR	4500
103	Diya	SALES	5280
101	Emma	SALES	5500

7 rows returned in 0.01 seconds

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.

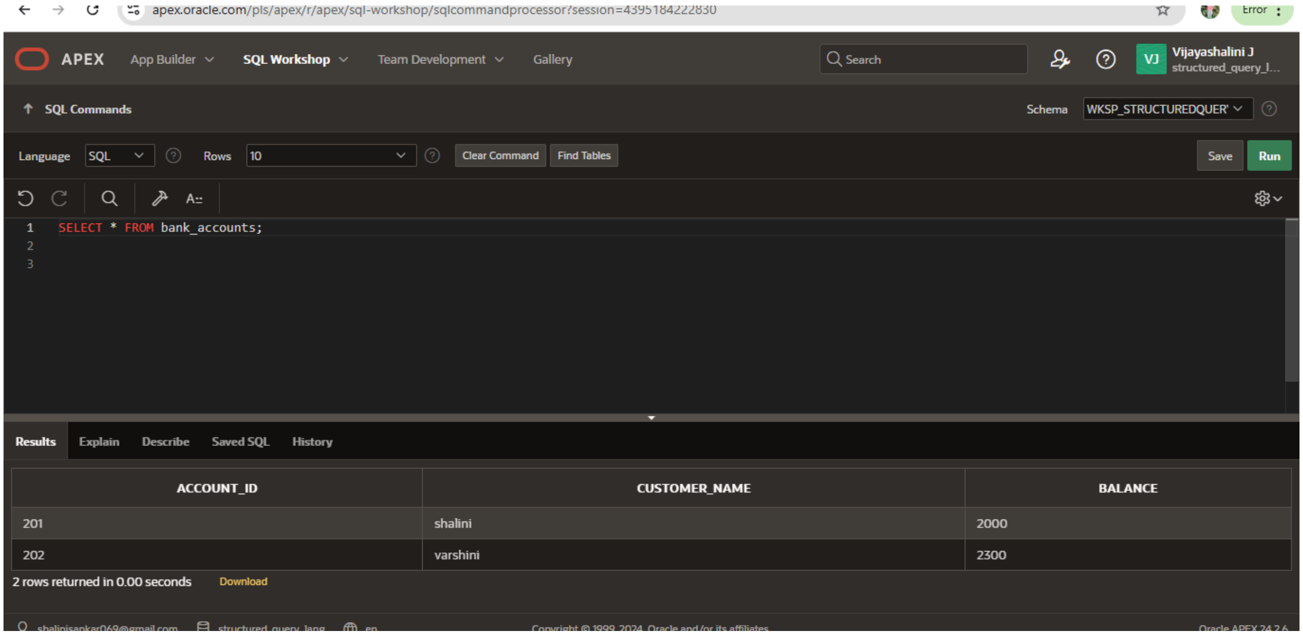
```
CREATE OR REPLACE PROCEDURE TransferFunds(
  p_from_account_id IN NUMBER,
  p_to_account_id IN NUMBER,
  p_amount IN NUMBER
) IS
  v_balance NUMBER;
BEGIN
  SELECT balance INTO v_balance
  FROM bank_accounts
  WHERE account_id = p_from_account_id
  FOR UPDATE;

  IF v_balance < p_amount THEN
    RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance.');
  END IF;

  UPDATE bank_accounts
  SET balance = balance - p_amount
  WHERE account_id = p_from_account_id;

  UPDATE bank_accounts
  SET balance = balance + p_amount
  WHERE account_id = p_to_account_id;

  COMMIT;
END;
```

CALL IT:

```
BEGIN
  TransferFunds(201, 202, 500);
```

```
END;
```

Output:



```sql
SELECT * FROM bank_accounts;
```

ACCOUNT_ID	CUSTOMER_NAME	BALANCE
201	shalini	2000
202	varshini	2300

2 rows returned in 0.00 seconds