Name-Harshit Selarka
Prn-23070521131
Section-B2

Lab 8
PL/SQL Procedure for Fund Transfer

Step 1: Create Database Tables

1.1 Create `accounts` Table

```
CREATE TABLE accounts (
    account_no NUMBER PRIMARY KEY,
    holder_name VARCHAR2(100),
    balance NUMBER(10,2) CHECK (balance >= 0)
);
```

1.2 Create `transactions` Table

```
CREATE TABLE transactions (
    transaction_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY
KEY,
    from_account NUMBER,
    to_account NUMBER,
    amount NUMBER(10,2),
    transaction_date TIMESTAMP DEFAULT SYSTIMESTAMP
);
```

Step 2: Insert Sample Data

```
INSERT INTO accounts VALUES (101, 'Alice', 5000.00);
INSERT INTO accounts VALUES (102, 'Bob', 3000.00);
COMMIT;
```

Step 3: Write PL/SQL Procedure

```
CREATE OR REPLACE PROCEDURE transfer_funds(
    p_from_acc NUMBER,
    p_to_acc NUMBER,
    p_amount NUMBER
) AS
    v_balance NUMBER;
BEGIN
    -- Check if sender has sufficient balance
    SELECT balance INTO v_balance FROM accounts WHERE account_no =
p_from_acc;

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

    -- Deduct amount from sender
    UPDATE accounts SET balance = balance - p_amount WHERE account_no
= p_from_acc;

    -- Add amount to receiver
    UPDATE accounts SET balance = balance + p_amount WHERE account_no
= p_to_acc;

    -- Log transaction
    INSERT INTO transactions (from_account, to_account, amount)
    VALUES (p_from_acc, p_to_acc, p_amount);
    -- Commit transaction
    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Transfer successful.');
```

```
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20002, 'Invalid account number.');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20003, 'Transaction failed: ' ||
SQLERRM);
END;
/
```

```
SQL> CREATE OR REPLACE PROCEDURE transfer_funds(
  2  p_from_acc NUMBER,
  3  p_to_acc NUMBER,
  4  p_amount NUMBER
  5  ) AS
  6  v_balance NUMBER;
  7  BEGIN
  8  -- Check if sender has sufficient balance
  9  SELECT balance INTO v_balance FROM accounts WH
 10  IF v_balance < p_amount THEN
 11  RAISE_APPLICATION_ERROR(-20001, 'Insufficient
 12  -- Deduct amount from sender
 13  UPDATE accounts SET balance = balance - p_amou
 14  -- Add amount to receiver
 15  UPDATE accounts SET balance = balance + p_amou
 16  -- Log transaction
 17  INSERT INTO transactions (from_account, to_acc
 18  -- Commit transaction
 19  COMMIT;
 20  DBMS_OUTPUT.PUT_LINE('Transfer successful.');
 21  EXCEPTION
 22  WHEN NO_DATA_FOUND THEN
 23  RAISE_APPLICATION_ERROR(-20002, 'Invalid accou
 24  WHEN OTHERS THEN
 25  ROLLBACK;
 26  RAISE_APPLICATION_ERROR(-20003, 'Transaction f
 27  END;
 28  /

Procedure created.
```

This procedure facilitates fund transfers between two bank accounts.

It verifies if the sender has enough balance before deducting the amount

The receiver's account is then credited, and the transaction is recorded

It incorporates error handling for low balance, incorrect account details, and other potential issues.

Step 4: Execute Procedure

```
BEGIN
    transfer_funds(101, 102, 1000);
END;
/
```

```
SQL> BEGIN
  2  transfer_funds(101, 102, 1000);
  3  END;
  4  /

PL/SQL procedure successfully completed.
```

- This block initiates the previously created procedure with the necessary input.

Step 5: Verify Results

Check Account Balances

```
SELECT * FROM accounts;
```

```
SQL> SELECT * FROM accounts;

ACCOUNT_NO
----------
HOLDER_NAME
------------------------------
   BALANCE
----------
       101
Alice
      4000

       102
Bob
      4000

ACCOUNT_NO
----------
HOLDER_NAME
------------------------------
   BALANCE
----------
```

Check Transactions Log

```
SELECT * FROM transactions;
```

```
SQL> SELECT * FROM transactions;

TRANSACTION_ID FROM_ACCOUNT TO_ACCOUNT     AMOUNT
-------------- ------------ ---------- ----------
TRANSACTION_DATE
---------------------------------------------------
             1          101        102       1000
03-APR-25 02.28.47.845000 PM
```

Task: Fund Transfer Validation and Execution

Task 1: Check Account Balance Before Transfer - Write a PL/SQL block that takes an account number as input and displays the account balance.

Hint: Use `SELECT balance INTO` inside a PL/SQL block and `DBMS_OUTPUT.PUT_LINE` to display the balance.

```
SQL> DECLARE
  2      v_account_no  NUMBER := &account_number;
  3      v_balance     NUMBER;
  4  BEGIN
  5      SELECT balance INTO v_balance
  6      FROM accounts
  7      WHERE account_no = v_account_no;
  8
  9      DBMS_OUTPUT.PUT_LINE('Account Number: ' || v_account_no);
 10      DBMS_OUTPUT.PUT_LINE('Current Balance: ' || v_balance);
 11
 12  EXCEPTION
 13      WHEN NO_DATA_FOUND THEN
 14          DBMS_OUTPUT.PUT_LINE('Error: Account not found.');
 15      WHEN OTHERS THEN
 16          DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
 17  END;
 18  /
Enter value for account_number: 101
old   2:      v_account_no  NUMBER := &account_number;
new   2:      v_account_no  NUMBER := 101;
Account Number: 101
Current Balance: 4000

PL/SQL procedure successfully completed.
```

Task 2: Execute Fund Transfer Procedure - Call the `transfer_funds` procedure to transfer **₹500 from account 101 to account 102**.

```
  6      v_from_balance NUMBER;
  7  BEGIN
  8      SELECT balance INTO v_from_balance
  9      FROM accounts
 10      WHERE account_no = p_from_account;
 11
 12      IF v_from_balance < p_amount THEN
 13          RAISE_APPLICATION_ERROR(-20001, 'Insufficient fur
 14      END IF;
 15
 16      UPDATE accounts
 17      SET balance = balance - p_amount
 18      WHERE account_no = p_from_account;
 19
 20      UPDATE accounts
 21      SET balance = balance + p_amount
 22      WHERE account_no = p_to_account;
 23
 24      COMMIT;
 25
 26      DBMS_OUTPUT.PUT_LINE('Transfer successful: ₹' || p_am
 27  EXCEPTION
 28      WHEN NO_DATA_FOUND THEN
 29          DBMS_OUTPUT.PUT_LINE('Error: One or both accounts
 30      WHEN OTHERS THEN
 31          DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
 32  END;
 33  /

Procedure created.

SQL> BEGIN
  2      transfer_funds(101, 102, 500);
  3  END;
  4  /
Transfer successful: ?500 from 101 to 102
```

Hint: Use the `BEGIN...END;` block to execute the procedure.

Task 3: Validate Transaction Log - After executing the transfer, write an SQL query to display all transactions recorded in the `transactions` table.

Hint: Use `SELECT * FROM transactions;` to verify the transaction details.

```
SQL> SELECT * FROM transactions;

TRANSACTION_ID FROM_ACCOUNT TO_ACCOUNT     AMOUNT
-------------- ------------ ---------- ----------
TRANSACTION_DATE
---------------------------------------------------------------------------
             1          101        102       1000
03-APR-25 02.29.31.851000 PM

             2          101        102        500
03-APR-25 02.35.44.291000 PM
```

Task 4: Check Transaction History for a Specific Account

Write a PL/SQL block that takes an account number as input and displays all transactions (both sent and received) related to that account.

```
SQL> DECLARE
  2      v_acc_no NUMBER := &Enter_Account_Number;
  3  BEGIN
  4      FOR rec IN (SELECT * FROM transactions WHERE from_account = v_acc_no OR to_account = v_acc_no)
  5      LOOP
  6          DBMS_OUTPUT.PUT_LINE('Transaction ID: ' || rec.transaction_id ||
  7                               ', From: ' || rec.from_account ||
  8                               ', To: ' || rec.to_account ||
  9                               ', Amount: ₹' || rec.amount ||
 10                               ', Date: ' || rec.transaction_date);
 11      END LOOP;
 12  EXCEPTION
 13      WHEN NO_DATA_FOUND THEN
 14          DBMS_OUTPUT.PUT_LINE('No transactions found for this account.');
 15  END;
 16  /
Enter value for enter_account_number: 101
old   2:      v_acc_no NUMBER := &Enter_Account_Number;
new   2:      v_acc_no NUMBER := 101;
Transaction ID: 1, From: 101, To: 102, Amount: ₹1000, Date: 03-APR-25
02.28.47.845000 PM

PL/SQL procedure successfully completed.

SQL> |
```

Hint: Use `SELECT * FROM transactions WHERE from_account = acc_no OR to_account = acc_no;` inside a PL/SQL block.

Task 5: Prevent Self-Transfer

Modify the `transfer_funds` procedure to prevent an account from transferring money to itself. If the sender and receiver accounts are the same, raise an error message.

Hint: Add a condition inside the procedure:

```
IF p_from_acc = p_to_acc THEN
   RAISE_APPLICATION_ERROR(-20004, 'Sender and receiver cannot be the
```

```
same.');
END IF;
```

```
SQL> CREATE OR REPLACE PROCEDURE transfer_funds(
  2       p_from_acc NUMBER,
  3       p_to_acc NUMBER,
  4       p_amount NUMBER
  5   ) AS
  6       v_balance NUMBER;
  7   BEGIN
  8       IF p_from_acc = p_to_acc THEN
  9           RAISE_APPLICATION_ERROR(-20004, 'Sender and re
 10       END IF;
 11
 12       SELECT balance INTO v_balance FROM accounts WHERE
 13
 14       IF v_balance < p_amount THEN
 15           RAISE_APPLICATION_ERROR(-20001, 'Insufficient
 16       END IF;
 17
 18       UPDATE accounts SET balance = balance - p_amount W
 19       UPDATE accounts SET balance = balance + p_amount W
 20
 21       INSERT INTO transactions (from_account, to_account
 22       VALUES (p_from_acc, p_to_acc, p_amount, SYSDATE);
 23
 24       COMMIT;
 25       DBMS_OUTPUT.PUT_LINE('Transfer successful.');
 26   EXCEPTION
 27       WHEN NO_DATA_FOUND THEN
```

Task 6: Create a Function to Check Account Balance

Write a PL/SQL function named `get_balance` that takes an account number as input and returns the current balance.

```
SQL> CREATE OR REPLACE FUNCTION get_balance(p_acc_no NUMBER) RETURN NUMBER AS
  2       v_balance NUMBER;
  3   BEGIN
  4       SELECT balance INTO v_balance FROM accounts WHERE account_no = p_acc_no;
  5       RETURN v_balance;
  6   EXCEPTION
  7       WHEN NO_DATA_FOUND THEN
  8           RETURN NULL;  -- Return NULL if account does not exist
  9   END;
 10   /

Function created.
```

Hint:

```
CREATE OR REPLACE FUNCTION get_balance(p_acc_no NUMBER) RETURN NUMBER
AS
    v_balance NUMBER;
BEGIN
    SELECT balance INTO v_balance FROM accounts WHERE account_no =
p_acc_no;
    RETURN v_balance;
END;
/
```

Call it using:

```
SELECT get_balance(101) FROM dual;
```

Task 7: Implement a Transfer Limit

Modify the `transfer_funds` procedure to set a maximum transfer limit of ₹10,000 per transaction. If a user tries to transfer more than this amount, raise an error.

```
SQL> CREATE OR REPLACE PROCEDURE transfer_funds(
  2      p_from_acc NUMBER,
  3      p_to_acc NUMBER,
  4      p_amount NUMBER
  5  ) AS
  6      v_balance NUMBER;
  7  BEGIN
  8      -- Prevent self-transfer
  9      IF p_from_acc = p_to_acc THEN
 10          RAISE_APPLICATION_ERROR(-20004, 'Sender and receiver cannot be the same.');
 11      END IF;
 12
 13      -- Check transfer limit
 14      IF p_amount > 10000 THEN
 15          RAISE_APPLICATION_ERROR(-20005, 'Transfer amount exceeds the limit of ₹10,000.');
 16      END IF;
 17
 18      -- Retrieve sender's balance
 19      SELECT balance INTO v_balance FROM accounts WHERE account_no = p_from_acc;
 20
 21      -- Check if sender has sufficient balance
 22      IF v_balance < p_amount THEN
 23          RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance.');
 24      END IF;
 25
 26      -- Deduct amount from sender
 27      UPDATE accounts SET balance = balance - p_amount WHERE account_no = p_from_acc;
 28
 29      -- Add amount to receiver
 30      UPDATE accounts SET balance = balance + p_amount WHERE account_no = p_to_acc;
 31
 32      -- Log transaction
 33      INSERT INTO transactions (from_account, to_account, amount, transaction_date)
 34      VALUES (p_from_acc, p_to_acc, p_amount, SYSDATE);
 35
 36      -- Commit transaction
 37      COMMIT;
 38      DBMS_OUTPUT.PUT_LINE('Transfer successful.');
 39
 40  EXCEPTION
 41      WHEN NO_DATA_FOUND THEN
 42          RAISE_APPLICATION_ERROR(-20002, 'Invalid account number.');
 43      WHEN OTHERS THEN
 44          ROLLBACK;
 45          RAISE_APPLICATION_ERROR(-20003, 'Transaction failed: ' || SQLERRM);
 46  END;
 47  /

Procedure created.
```

Hint: Add a condition:

```
IF p_amount > 10000 THEN
   RAISE_APPLICATION_ERROR(-20005, 'Transfer amount exceeds the limit
of ₹10,000.');
END IF;
```

Task 8: Generate a Monthly Statement

Write a PL/SQL procedure that takes an account number and a month-year (e.g., `04-2025`) as input and displays all transactions for that month.

Hint: Use `TO_CHAR(transaction_date, 'MM-YYYY')` in the `WHERE`

clause: `SELECT * FROM transactions`

`WHERE(from_account=acc_noORto_account=acc_no)`

```
ANDTO_CHAR(transaction_date, 'MM-YYYY')= '04-2025';
```

```
SQL> CREATE OR REPLACE PROCEDURE monthly_statement(
  2      p_acc_no NUMBER,
  3      p_month_year VARCHAR2
  4  ) AS
  5  BEGIN
  6      FOR rec IN (SELECT * FROM transactions
  7                  WHERE (from_account = p_acc_no OR to_account = p_acc_no)
  8                  AND TO_CHAR(transaction_date, 'MM-YYYY') = p_month_year)
  9      LOOP
 10          DBMS_OUTPUT.PUT_LINE('Transaction ID: ' || rec.transaction_id ||
 11                               ', From: ' || rec.from_account ||
 12                               ', To: ' || rec.to_account ||
 13                               ', Amount: ₹' || rec.amount ||
 14                               ', Date: ' || rec.transaction_date);
 15      END LOOP;
 16  EXCEPTION
 17      WHEN NO_DATA_FOUND THEN
 18          DBMS_OUTPUT.PUT_LINE('No transactions found for the given period.');
 19  END;
 20  /

Procedure created.
```

```
SQL> BEGIN
  2      monthly_statement(101, '04-2025');
  3  END;
  4  /

PL/SQL procedure successfully completed.
```