**NAME:Purva Tagde**

**PRN:23070521111**

**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) );

**CREATED**

## 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 );

**CREATED**

# Step 2: Insert Sample Data

INSERT INTO accounts VALUES (101, 'Alice', 5000.00);

INSERT INTO accounts VALUES (102, 'Bob', 3000.00); COMMIT;

**done**

# 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

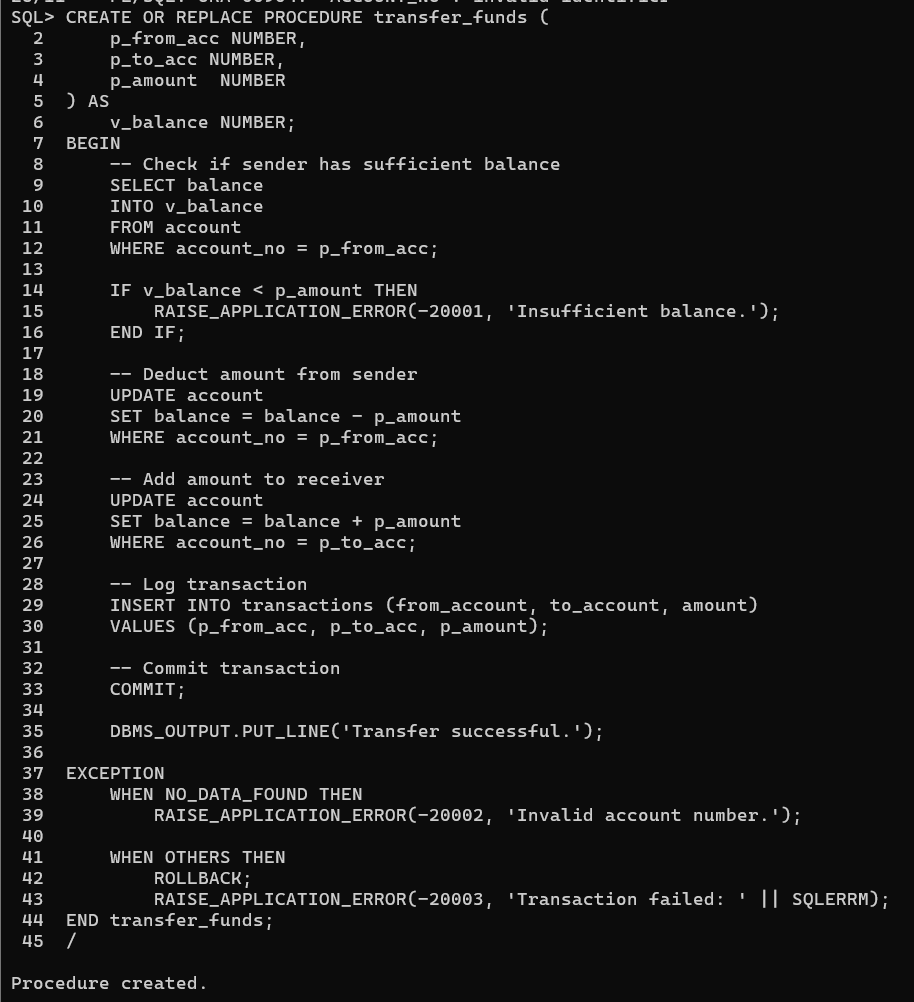
RAISE\_APPLICATION\_ERROR(-20002, 'Invalid account number.'); WHEN OTHERS THEN

ROLLBACK;

RAISE\_APPLICATION\_ERROR(-20003, 'Transaction failed: ' ||

SQLERRM);

END; /

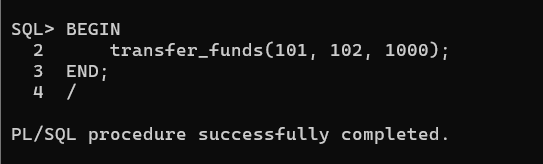


# Step 4: Execute Procedure

BEGIN

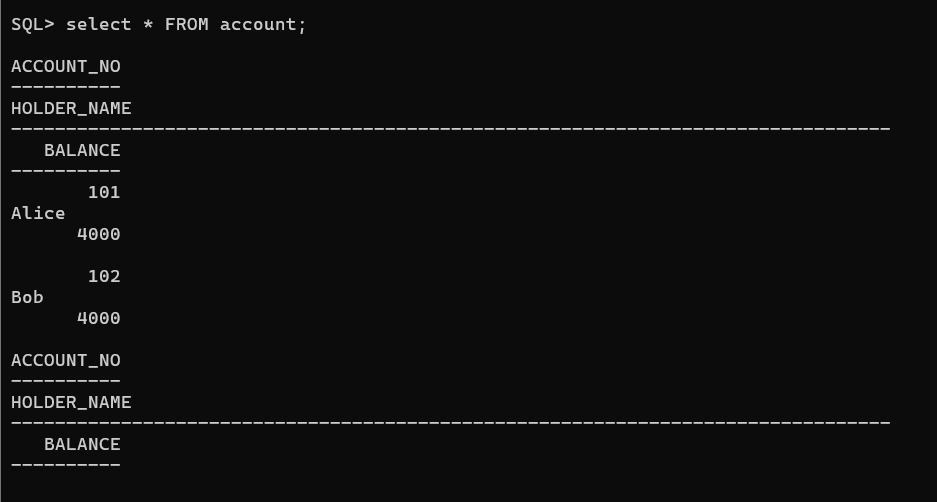
transfer\_funds(101, 102, 1000);

END; /



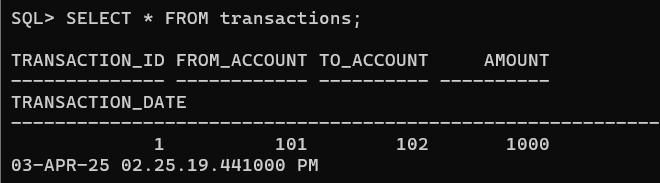
# Step 5: Verify Results

**Check Account Balances** SELECT \* FROM accounts;



**Check Transactions Log**

SELECT \* FROM transactions;

****

## 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.

DECLARE

v\_acc\_no NUMBER := 101; -- **variable acc\_no is created with datatype number** **to take** **input**

v\_balance NUMBER; -- **Variable to store balance with data type number**

BEGIN

-- **Retrieve balance from account table**

SELECT balance INTO v\_balance FROM account WHERE account\_no = v\_acc\_no;

-- **Display / printing the account balance**

DBMS\_OUTPUT.PUT\_LINE('Account Balance for ' || v\_acc\_no || ': ₹' || v\_balance);

EXCEPTION

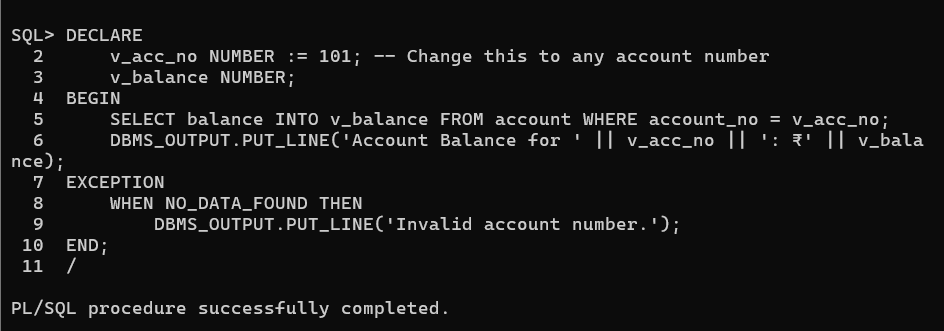
WHEN NO\_DATA\_FOUND THEN

-- **If account number is invalid, then print an message**

DBMS\_OUTPUT.PUT\_LINE('Invalid account number.');

END;

/



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

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

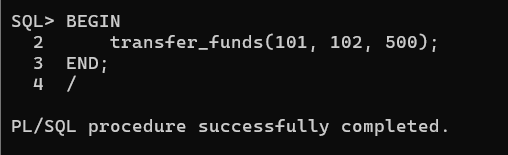
BEGIN

**-- Calling transfer\_funds with parameters** **Sender Account: 101, Receiver Account: 102, Amount: ₹500**

transfer\_funds(101, 102, 500);

END;

/

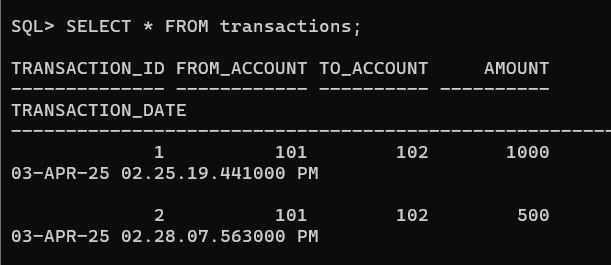
****

**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**.**

**-- Showing /display all rows from transactions table**

SELECT \* FROM transactions;

****

### 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.

**Hint:** Use SELECT \* FROM transactions WHERE from\_account = acc\_no OR to\_account = acc\_no; inside a PL/SQL block.

DECLARE

v\_acc\_no NUMBER := 101; **-- Variable acc\_no of datatype number**

BEGIN

**-- display all transactions involving this account**

FOR rec IN (

SELECT \* FROM transactions

WHERE from\_account = v\_acc\_no OR to\_account = v\_acc\_no

) LOOP

**-- Display/print transaction details**

DBMS\_OUTPUT.PUT\_LINE('Transaction ID: ' || rec.transaction\_id ||

', From: ' || rec.from\_account ||

', To: ' || rec.to\_account ||

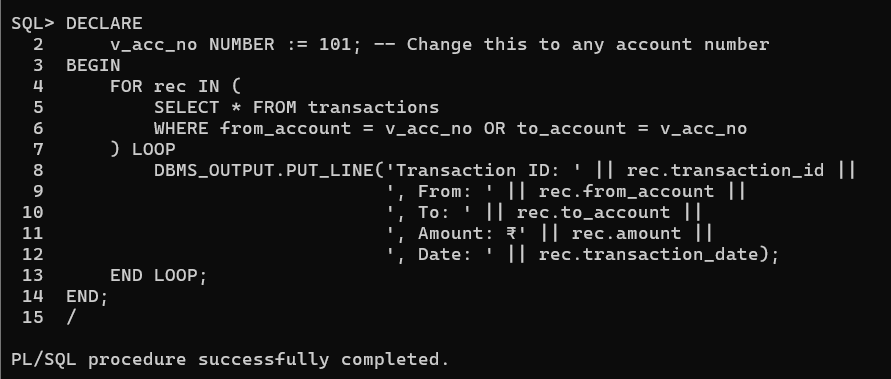
', Amount: ₹' || rec.amount ||

', Date: ' || rec.transaction\_date);

END LOOP;

END;

/



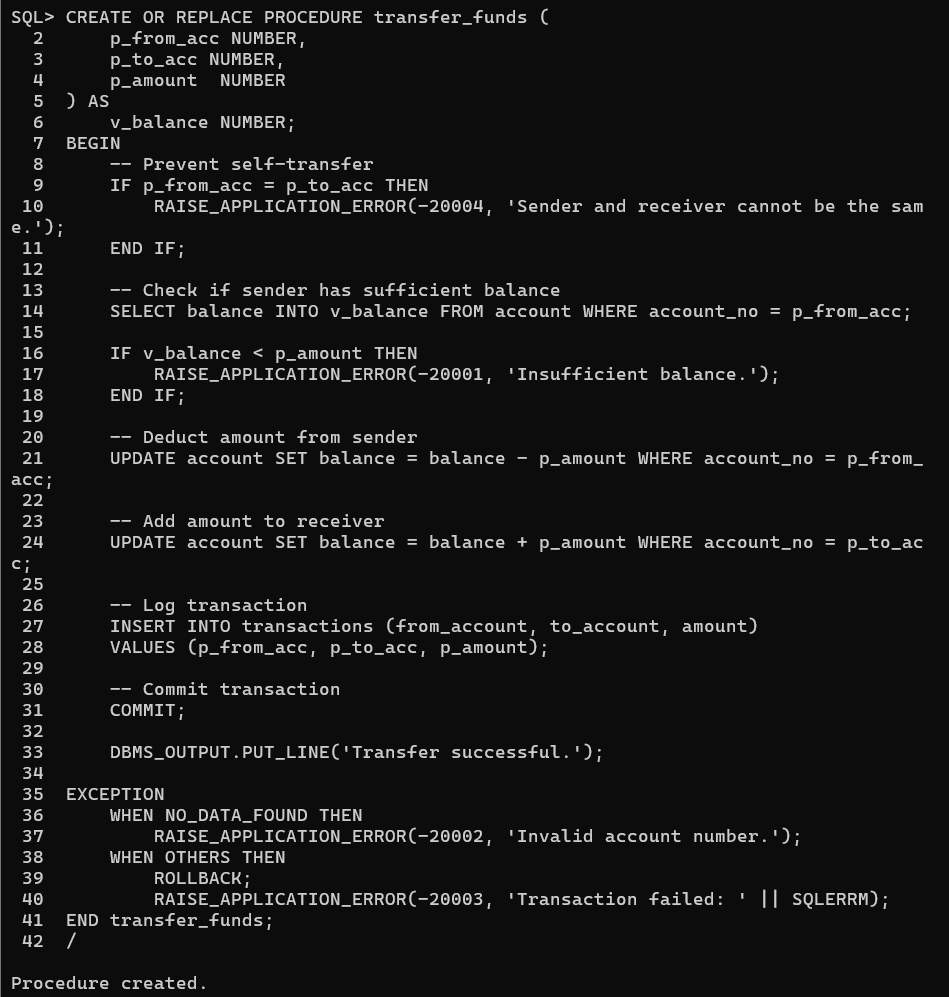
### Task 5: Prevent Self-Transfer

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;

****

CREATE OR REPLACE PROCEDURE transfer\_funds (

p\_from\_acc NUMBER, **-- Sender account**

p\_to\_acc NUMBER, **-- Receiver account**

p\_amount NUMBER **-- Amount to transfer**

) AS

v\_balance NUMBER; **-- Variable to store sender’s balance of number datatype**

BEGIN

IF p\_from\_acc = p\_to\_acc THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Sender and receiver cannot be the same.');

END IF;

**-- Check if sender has enough balance**

SELECT balance INTO v\_balance FROM account WHERE account\_no = p\_from\_acc;

**-- If sender’s balance is less than the transfer amount, show an error**

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance.');

END IF;

**-- Deduct amount from sender**

UPDATE account SET balance = balance - p\_amount WHERE account\_no = p\_from\_acc;

**-- Add amount to receiver**

UPDATE account SET balance = balance + p\_amount WHERE account\_no = p\_to\_acc;

**-- transaction details**

INSERT INTO transactions (from\_account, to\_account, amount)

VALUES (p\_from\_acc, p\_to\_acc, p\_amount);

**-- Save the changes**

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 transfer\_funds;

/

### 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.

**Hint:**

CREATE OR REPLACE FUNCTION get\_balance(p\_acc\_no NUMBER) RETURN NUMBER AS

v\_balance NUMBER;

BEGIN

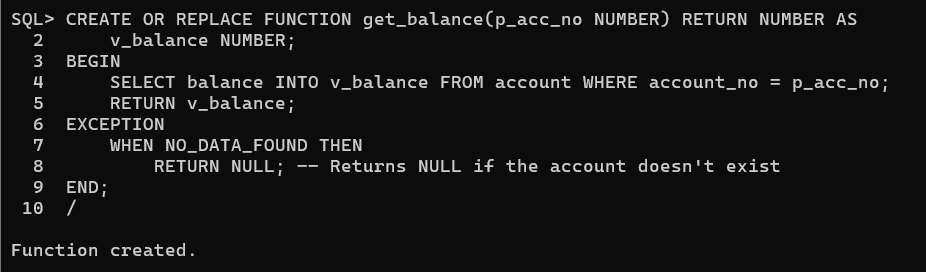
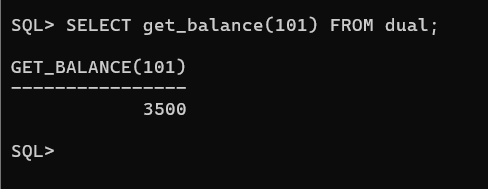
SELECT balance INTO v\_balance FROM accounts WHERE account\_no = p\_acc\_no;

RETURN v\_balance;

END; /

Call it using:

SELECT get\_balance(101) FROM dual;

CREATE OR REPLACE FUNCTION get\_balance(p\_acc\_no NUMBER) RETURN NUMBER AS

v\_balance NUMBER; **-- Variable to store balance of datatype number**

BEGIN

**-- Retrieve account balance from account table**

SELECT balance INTO v\_balance FROM account WHERE account\_no = p\_acc\_no;

**-- Return the account balance**

RETURN v\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

**-- Return NULL if the account number is invalid**

RETURN NULL;

END;

/

### 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.

**Hint:** Add a condition:

IF p\_amount > 10000 THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Transfer amount exceeds the limit of ₹10,000.');

END IF;



### CREATE OR REPLACE PROCEDURE transfer\_funds (

### p\_from\_acc NUMBER, -- Sender account

### p\_to\_acc NUMBER, -- Receiver account

### p\_amount NUMBER -- Amount to transfer

### ) AS

### v\_balance NUMBER; -- Variable to store sender’s balance

### BEGIN

### IF p\_from\_acc = p\_to\_acc THEN

### RAISE\_APPLICATION\_ERROR(-20004, 'Sender and receiver cannot be the same.');

### END IF;

### 

### IF p\_amount > 10000 THEN

### RAISE\_APPLICATION\_ERROR(-20005, 'Transfer amount exceeds the limit of ₹10,000.');

### END IF;

### -- Check if sender has enough balance

### SELECT balance INTO v\_balance FROM account WHERE account\_no = p\_from\_acc;

### -- If sender’s balance is less than the transfer amount, show an error

### IF v\_balance < p\_amount THEN

### RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance.');

### END IF;

### 

### UPDATE account SET balance = balance - p\_amount WHERE account\_no = p\_from\_acc;

### -- Add amount to receiver

### UPDATE account SET balance = balance + p\_amount WHERE account\_no = p\_to\_acc;

### 

### INSERT INTO transactions (from\_account, to\_account, amount)

### VALUES (p\_from\_acc, p\_to\_acc, p\_amount);

### -- save changes 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 transfer\_funds;

### /

### Task 8: Generate a Monthly Statement

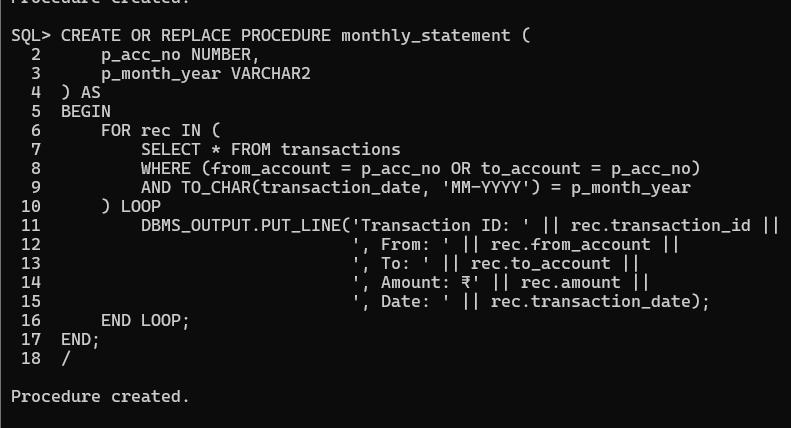
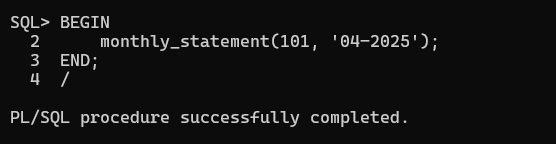
Write a PL/SQL procedure that takes an account number and a month-year (e.g., 04-2025) as input and displays all transactions for that month.

**Hint:** Use TO\_CHAR(transaction\_date, 'MM-YYYY') in the WHERE clause:

SELECT \* FROM transactions

WHERE (from\_account = acc\_no OR to\_account = acc\_no)

AND TO\_CHAR(transaction\_date, 'MM-YYYY') = '04-2025';

CREATE OR REPLACE PROCEDURE monthly\_statement (

p\_acc\_no NUMBER,

p\_month\_year VARCHAR2

) AS

BEGIN

FOR rec IN (

SELECT \* FROM transactions

WHERE (from\_account = p\_acc\_no OR to\_account = p\_acc\_no)

AND TO\_CHAR(transaction\_date, 'MM-YYYY') = p\_month\_year

) LOOP

**-- Display transaction details**

DBMS\_OUTPUT.PUT\_LINE('Transaction ID: ' || rec.transaction\_id ||

', From: ' || rec.from\_account ||

', To: ' || rec.to\_account ||

', Amount: ₹' || rec.amount ||

', Date: ' || rec.transaction\_date);

END LOOP;

END;

/