**Lab 8**

**PL/SQL Procedure for Fund Transfer**

**Step 1: Create Database Tables**

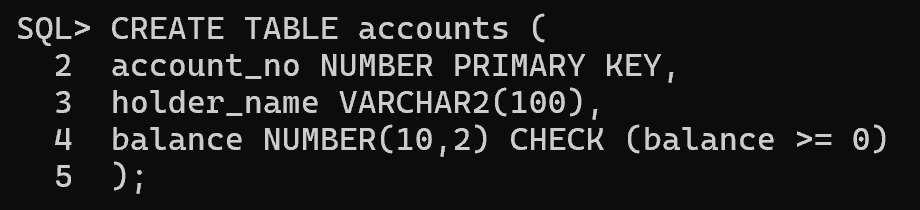
**1.1 Create accounts Table**

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 (

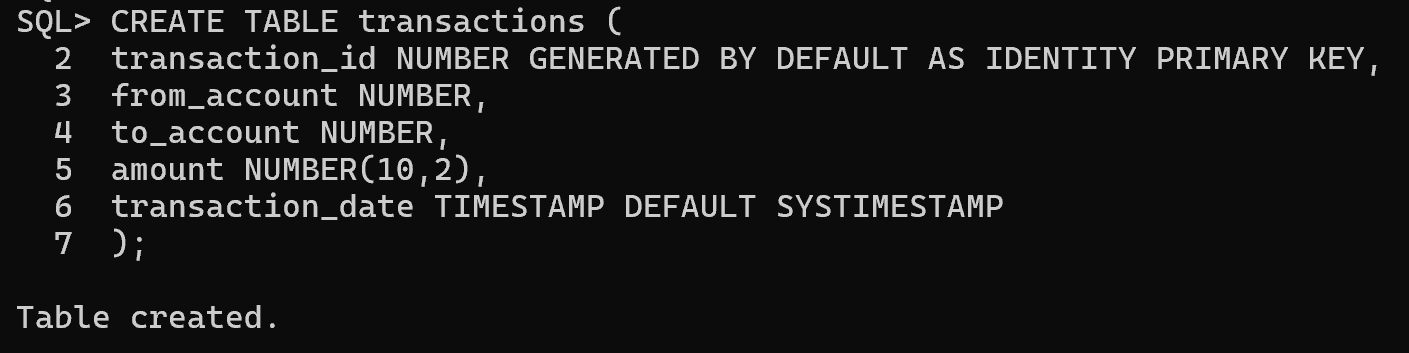
transaction\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

from\_account NUMBER,

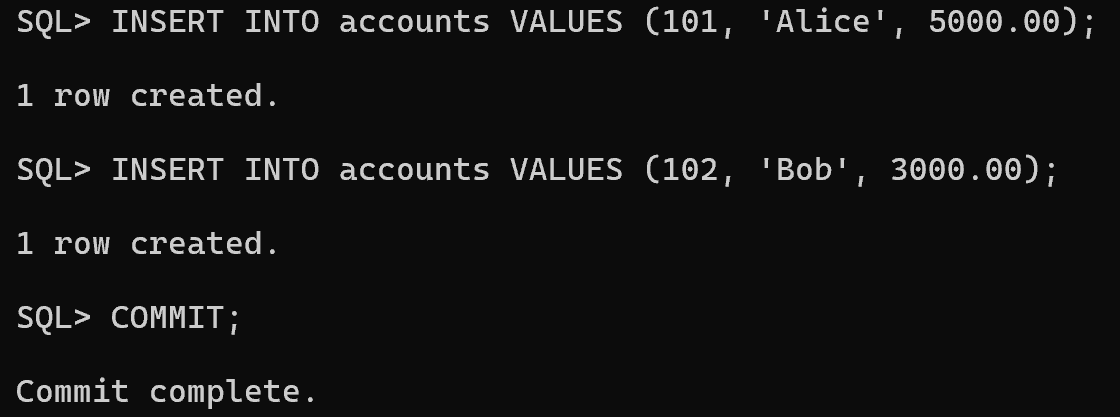
to\_account NUMBER,

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, -- Sender's account number

p\_to\_acc NUMBER, -- Receiver's account number

p\_amount NUMBER -- Amount to be transferred

) AS

v\_balance NUMBER; -- Variable to store sender's current balance

BEGIN

-- Retrieve the balance of the sender's account

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

-- Check if sender has sufficient balance

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance.'); -- Raise an error if funds are insufficient

END IF;

-- Deduct the amount from the sender's account

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

-- Add the amount to the receiver's account

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

-- Log the transaction in the transactions table

INSERT INTO transactions (from\_account, to\_account, amount) VALUES (p\_from\_acc, p\_to\_acc, p\_amount);

-- Commit the transaction to make changes permanent

COMMIT;

-- Print success message

DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

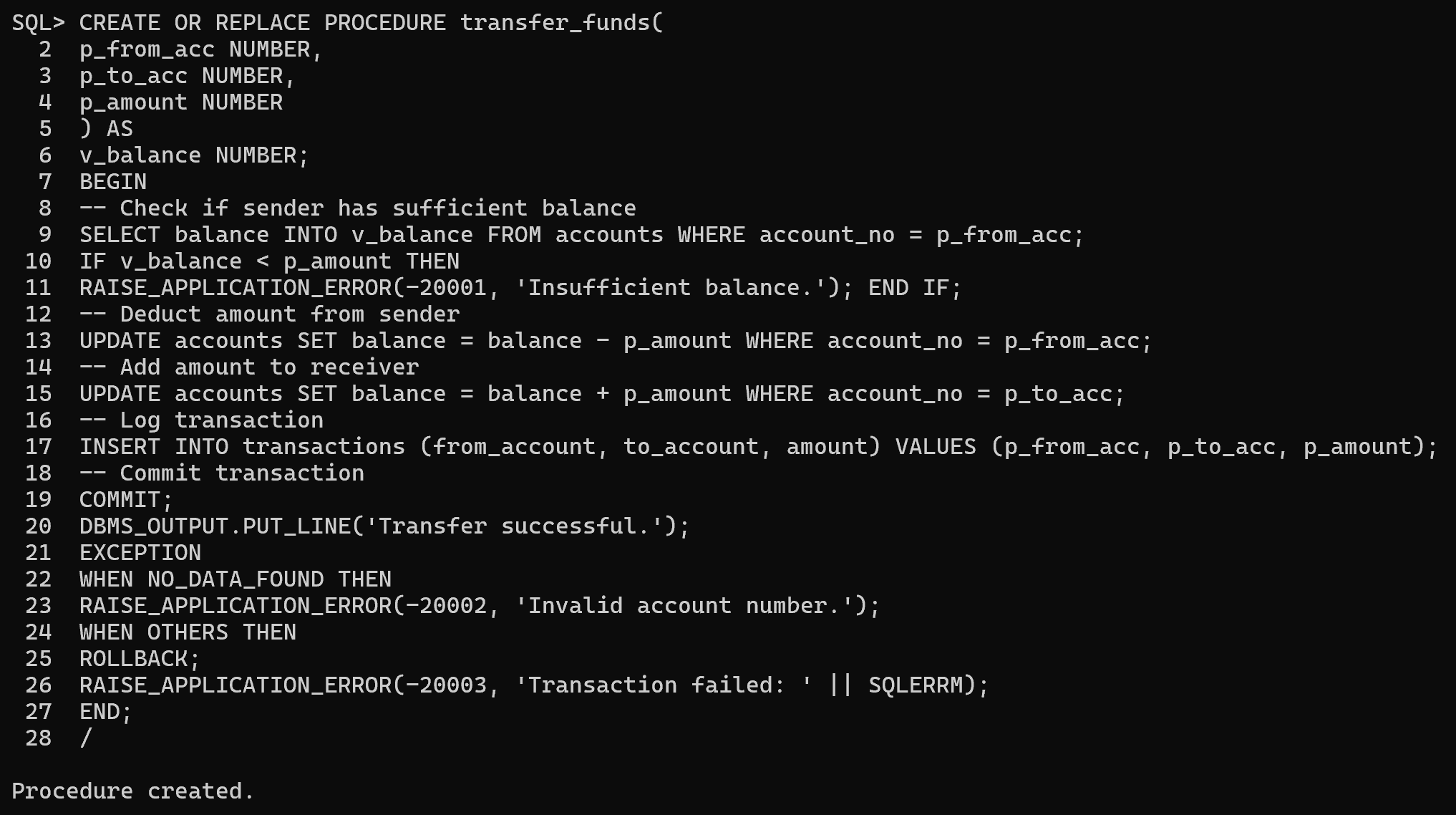
RAISE\_APPLICATION\_ERROR(-20002, 'Invalid account number.'); -- Handle case where sender's account does not exist

WHEN OTHERS THEN

ROLLBACK; -- Rollback transaction in case of any error

RAISE\_APPLICATION\_ERROR(-20003, 'Transaction failed: ' || SQLERRM); -- Raise a generic error with details

END;

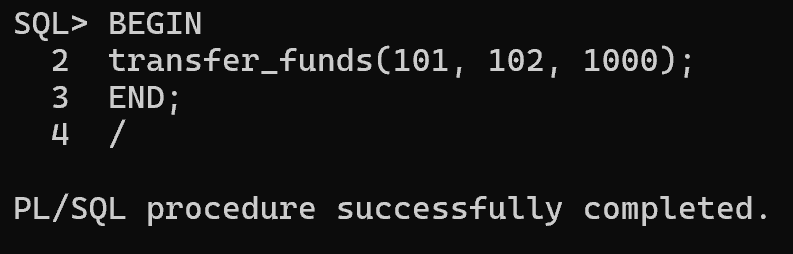
/ 

**Step 4: Execute Procedure**

BEGIN

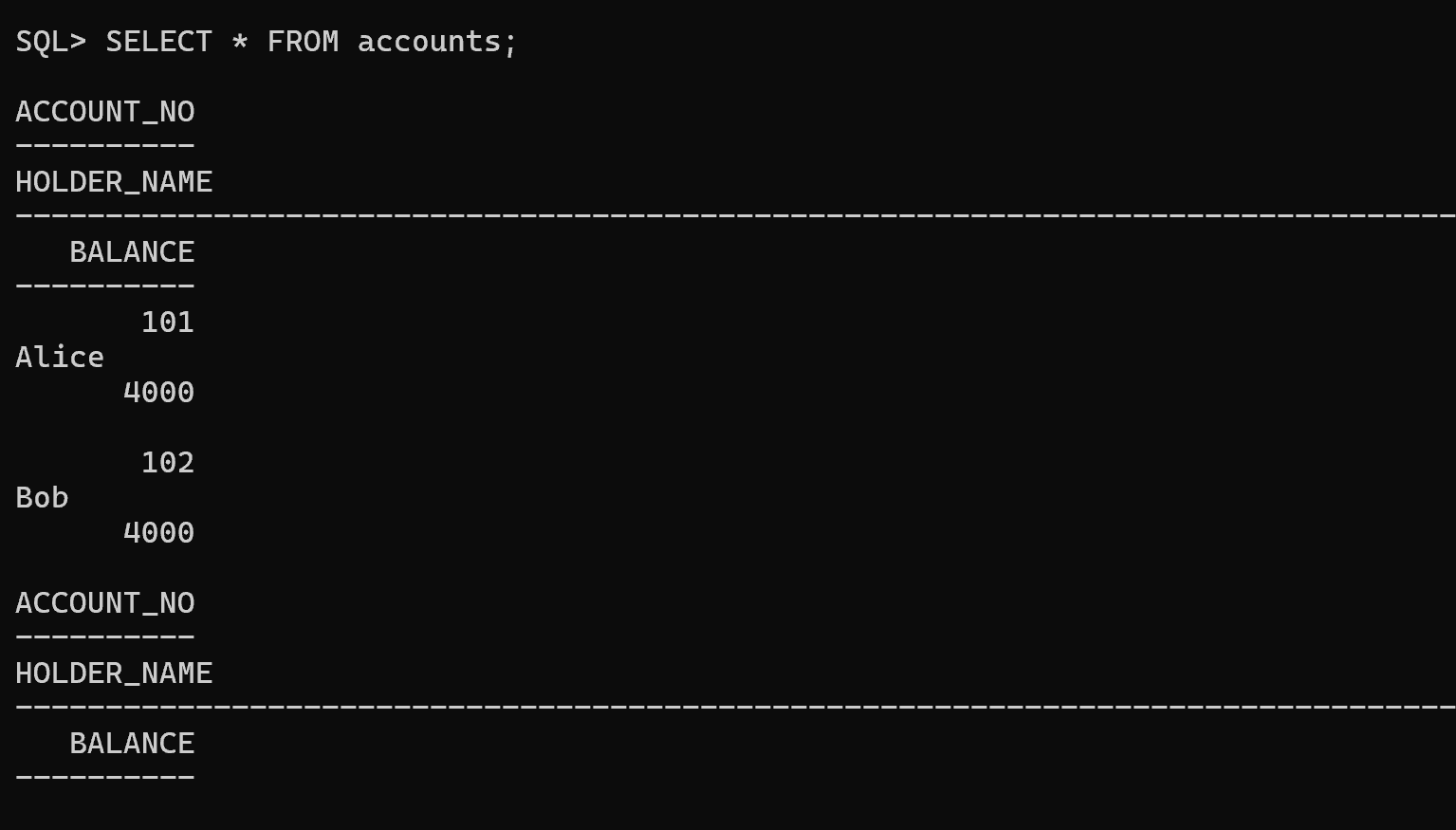
transfer\_funds(101, 102, 1000);

END;

/ 

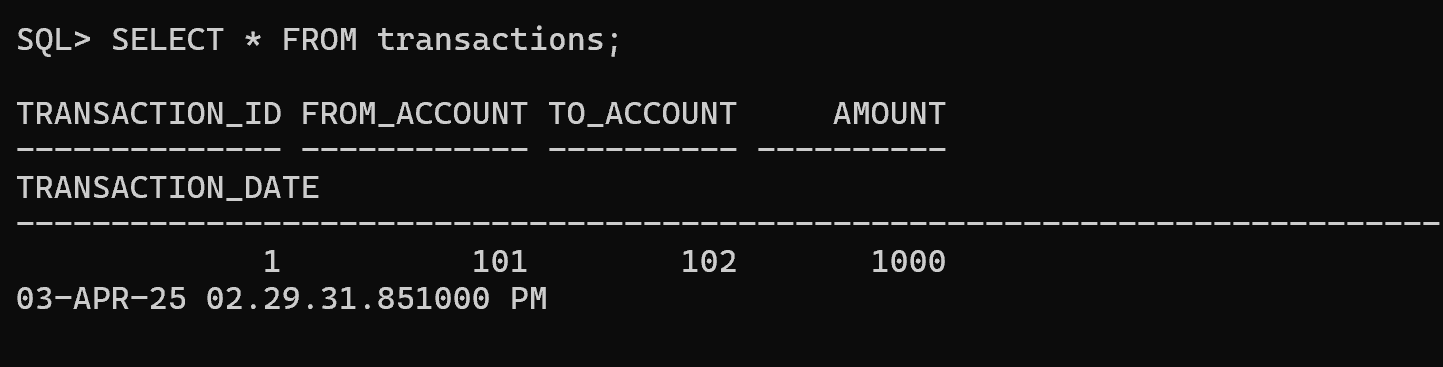
**Step 5: Verify Results**

**Check Account Balances**

SELECT \* FROM accounts; 

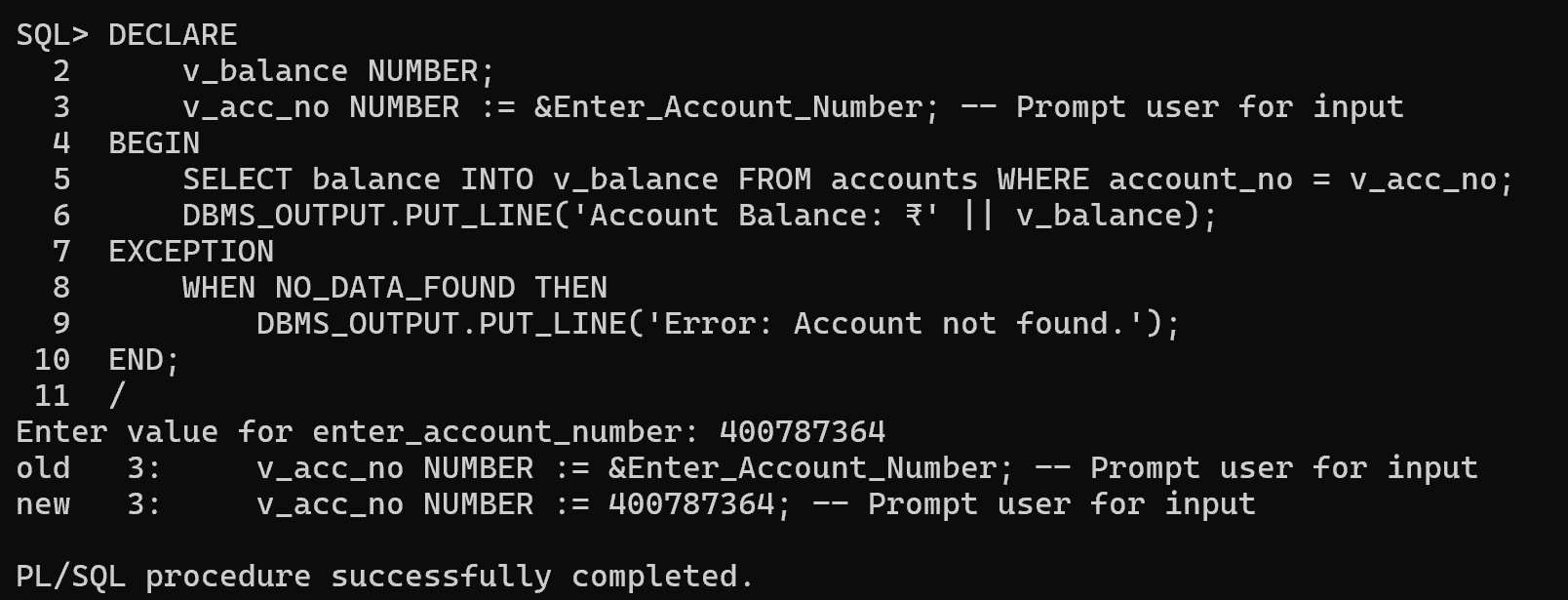
**Check Transactions Log**

SELECT \* FROM transactions;

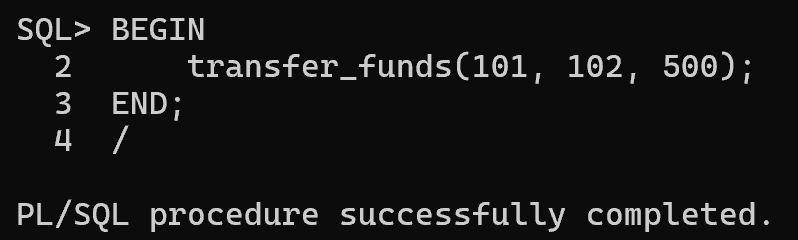


**Task: Fund Transfer Validation and Execution**

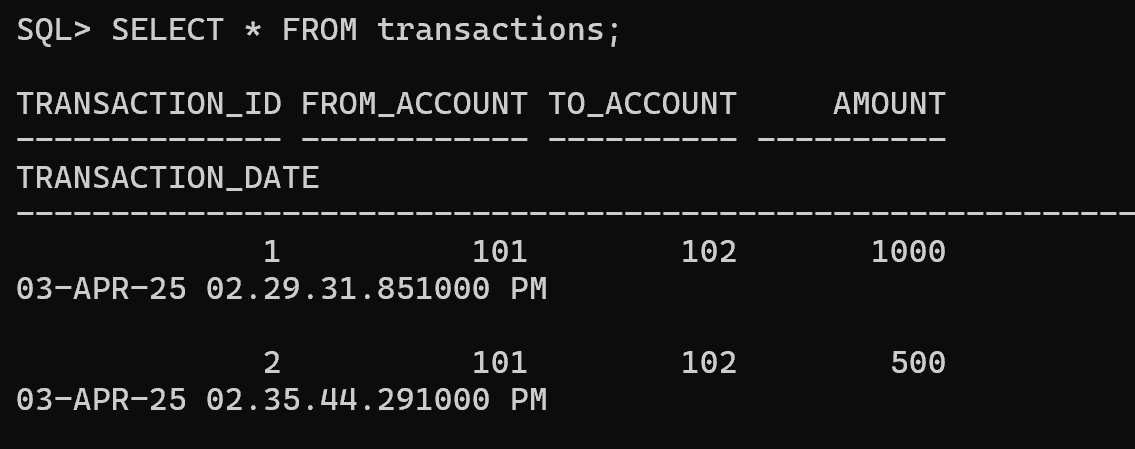
**Task 1: Check Account Balance Before Transfer -** Write a PL/SQL block that takes an account number as input and displays the account balance.

**Hint:** Use SELECT balance INTO inside a PL/SQL block and DBMS\_OUTPUT.PUT\_LINE to display the balance. 

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

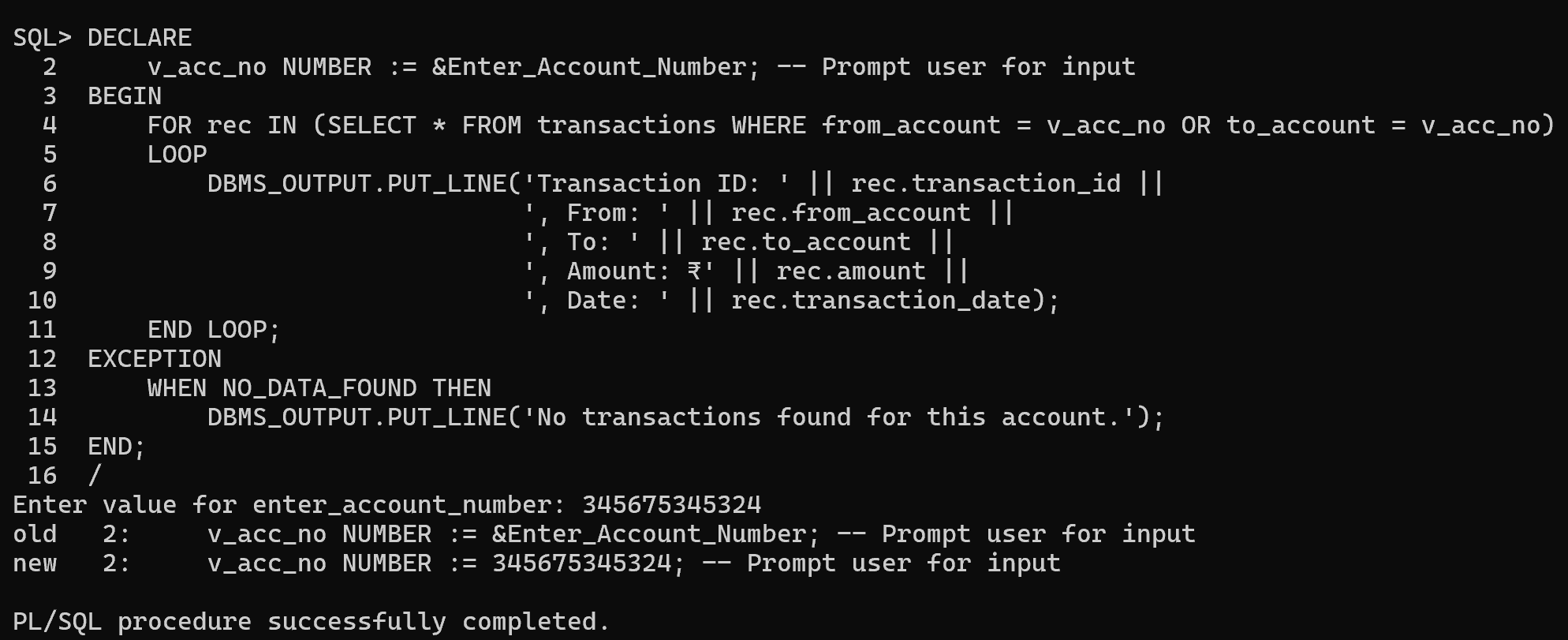
**Hint:** Use the BEGIN...END; block to execute the procedure. 

**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.

**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. 

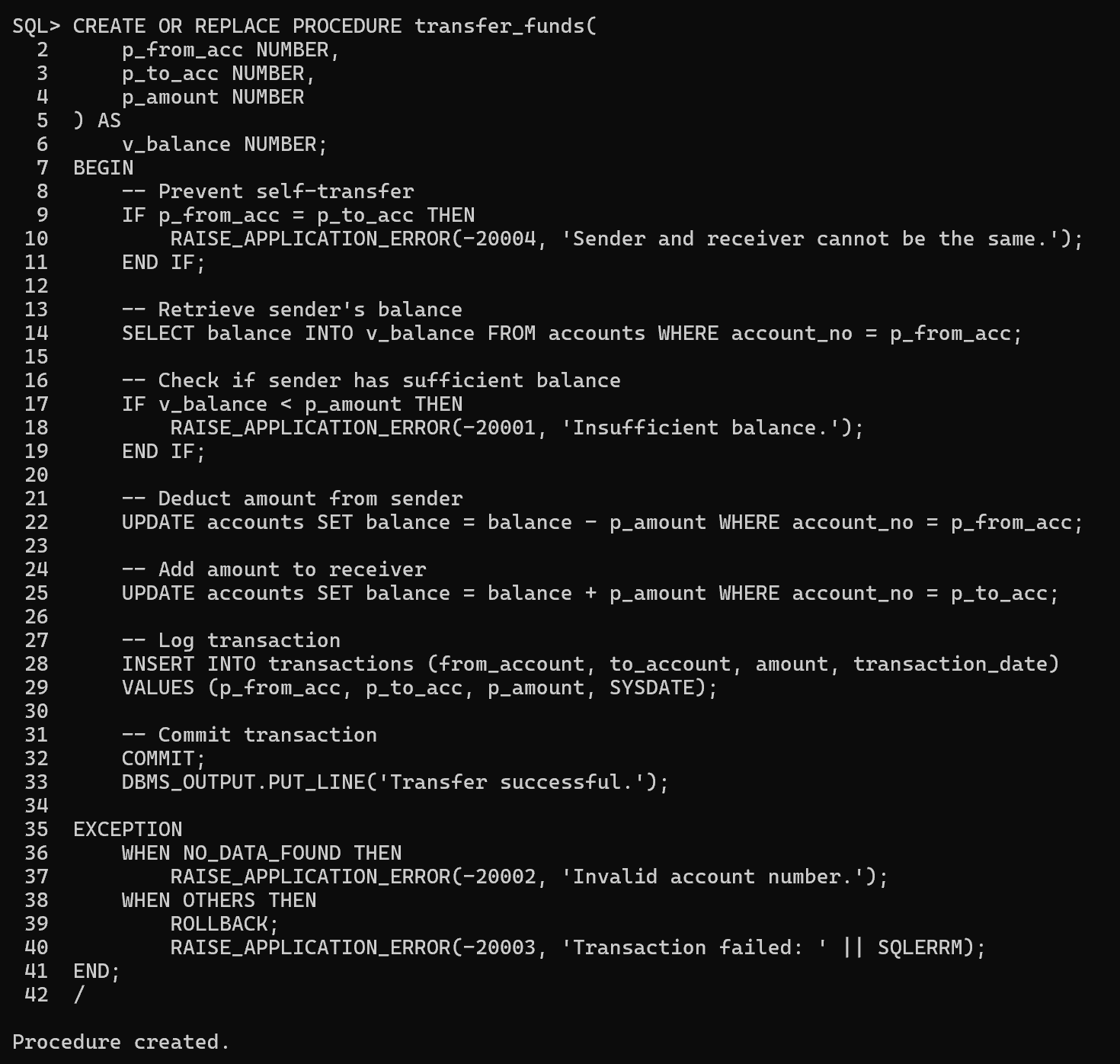
**Task 5: Prevent Self-Transfer**

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; 

**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

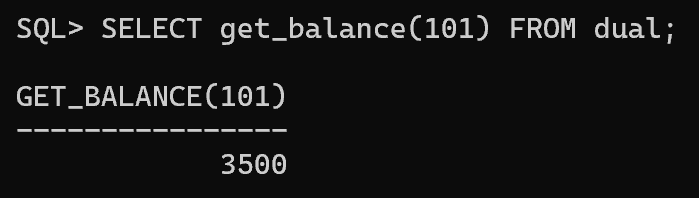
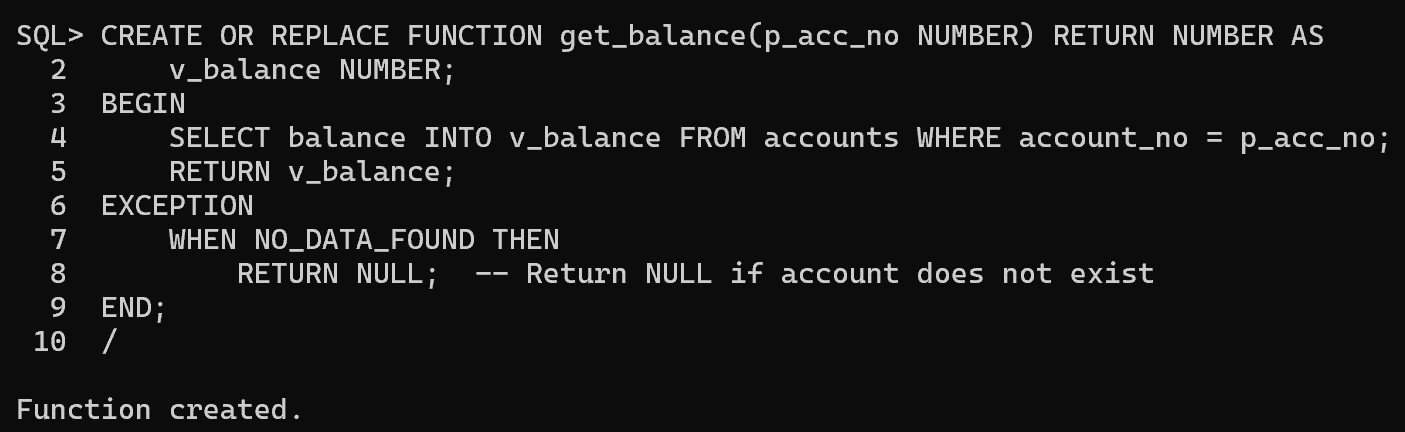
SELECT balance INTO v\_balance FROM accounts WHERE account\_no = p\_acc\_no;

RETURN v\_balance;

END;

/

Call it using:

SELECT get\_balance(101) FROM dual; 

**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; 

**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';