REVIEW 3

REPORT FILE

Table used by Prem :- Loan, Payment, Debit Card, Credit Card, Customer Table

Table used by Chiranjeev:- ATM, banktobank, Branch, Insurance, Mobile Banking

Tabke used by Aadit:-Transactions, Account, Merchant, Government Service, Bank Security

PREM LOHIA TABLES

TABLE CUSTOMER

Add a new column

ALTER TABLE customer

ADD Gender CHAR(1);

Modify existing column

ALTER TABLE customer

MODIFY (DOB DATE)

Rename column

ALTER TABLE customer

RENAME COLUMN Email\_id TO Email;

DROP TABLE customer;

TRUNCATE TABLE customer;

REVOKE INSERT, UPDATE, DELETE ON customer FROM user\_name;

TABLE PAYMENT

Add a new column

ALTER TABLE Payment

ADD Description VARCHAR2(25);

Modify column datatype

ALTER TABLE Payment

MODIFY (Payment\_amount NUMBER(12, 2));

Rename table

ALTER TABLE Payment

RENAME TO TransactionDetails;

DROP TABLE Payment;

TRUNCATE TABLE Payment;

REVOKE ALL PRIVILEGES ON Payment FROM user\_name;

TABLE CreditCard:

Add a new column

ALTER TABLE CreditCard

ADD CVV NUMBER(3);

Rename column

ALTER TABLE CreditCard

RENAME COLUMN card\_holder\_name TO holder\_name;

DROP TABLE CreditCard;

TRUNCATE TABLE CreditCard;

REVOKE SELECT ON CreditCard FROM user\_name;

TABLE DebitCard:

Add a new column

ALTER TABLE DebitCard

ADD Bank VARCHAR2(100);

Modify column datatype

ALTER TABLE DebitCard

MODIFY (expiration\_date DATE);

Rename table

ALTER TABLE DebitCard

RENAME TO DebitCardDetails;

DROP TABLE DebitCard;

TRUNCATE TABLE DebitCard;

REVOKE INSERT, UPDATE ON DebitCard FROM user\_name;

CHIRANJEEV KUMAR TABLES

TABLE branch:

Add a new column

ALTER TABLE branch

ADD branch\_manager VARCHAR2(50);

Modify existing column

ALTER TABLE branch

MODIFY IFSC\_code VARCHAR2(11);

Rename column

ALTER TABLE branch

RENAME COLUMN assests TO assets;

DROP TABLE branch;

TRUNCATE TABLE branch;

RENAME branch TO branches;

REVOKE INSERT, UPDATE, DELETE ON branch FROM user\_name;

TABLE Insurance:

Add a new column

ALTER TABLE Insurance

ADD coverage\_amount DECIMAL(10,2);

Modify column datatype

ALTER TABLE Insurance

MODIFY (start\_date DATE, end\_date DATE);

Rename table

ALTER TABLE Insurance

RENAME TO InsurancePolicy;

DROP TABLE InsurancePolicy;

TRUNCATE TABLE InsurancePolicy;

REVOKE ALL PRIVILEGES ON InsurancePolicy FROM user\_name;

TABLE banktobank:

Add a new column

ALTER TABLE banktobank

ADD transaction\_type VARCHAR2(20);

Modify column datatype

ALTER TABLE banktobank

MODIFY (amount DECIMAL(15,2), transaction\_date TIMESTAMP);

DROP TABLE BankTransactions;

TRUNCATE TABLE BankTransactions;

RENAME BankTransactions TO BankTransfers;

REVOKE SELECT ON BankTransactions FROM user\_name;

TABLE ATM\_CARD:

Add a new column

ALTER TABLE ATM\_CARD

ADD ATM\_location VARCHAR2(100);

Modify column datatype

ALTER TABLE ATM\_CARD

MODIFY (AmountWithdrawal DECIMAL(16,2), TransactionID INT);

DROP TABLE ATMTransactions;

TRUNCATE TABLE ATMTransactions;

RENAME ATMTransactions TO ATMTransfers;

REVOKE INSERT, UPDATE ON ATMTransactions FROM user\_name;

TABLE MobileBankingApps:

Add a new column

ALTER TABLE MobileBankingApps

ADD LastLoginDate DATE;

Modify column datatype

ALTER TABLE MobileBankingApps

MODIFY (AppName VARCHAR2(100));

DROP TABLE MobileApps;

TRUNCATE TABLE MobileApps;

RENAME MobileApps TO MobileApplications;

REVOKE ALL PRIVILEGES ON MobileApps FROM user\_name;

AADIT VINAYAK TABLES

TABLE account:

Add a new column

ALTER TABLE account

Modify existing column

ALTER TABLE account

MODIFY (Saving\_account VARCHAR(3), Current\_account VARCHAR(3));

Rename column

ALTER TABLE account

RENAME COLUMN Balance TO AccountBalance;

DROP TABLE account;

TRUNCATE TABLE account;

RENAME account TO accounts;

REVOKE INSERT, UPDATE, DELETE ON account FROM user\_name;

TABLE Transactions:

Add a new column

ALTER TABLE Transactions

ADD Description VARCHAR(255);

Modify column datatype

ALTER TABLE Transactions

MODIFY (TransactionDate TIMESTAMP);

Rename table

ALTER TABLE Transactions

RENAME TO BankTransactions;

DROP TABLE BankTransactions;

TRUNCATE TABLE BankTransactions;

RENAME BankTransactions TO Transactions;

REVOKE ALL PRIVILEGES ON Transactions FROM user\_name;

TABLE merchant:

Add a new column

ALTER TABLE merchant

ADD TransactionDescription VARCHAR(255);

Modify column datatype

ALTER TABLE merchant

MODIFY (phone\_no VARCHAR2(15));

DROP TABLE MerchantTransactions

TRUNCATE TABLE MerchantTransactions;

RENAME MerchantTransactions TO MerchantTrans;

REVOKE SELECT ON MerchantTrans FROM user\_name;

TABLE BankSecurity:

Add a new column

ALTER TABLE BankSecurity

ADD LastPasswordChangeDate DATE;

Modify column datatype

ALTER TABLE BankSecurity

MODIFY (AccountLockStatus VARCHAR2(20));

DROP TABLE UserSecurity;

TRUNCATE TABLE UserSecurity;

RENAME UserSecurity TO UserSec;

REVOKE INSERT, UPDATE ON UserSec FROM user\_name;

TABLE GovernmentServices:

Add a new column

ALTER TABLE GovernmentServices

ADD ServiceCategory VARCHAR(50);

Modify column datatype

ALTER TABLE GovernmentServices

MODIFY (ServiceName VARCHAR2(100));

DROP TABLE GovtServices;

TRUNCATE TABLE GovtServices;

RENAME GovtServices TO GovernmentServices;

REVOKE ALL PRIVILEGES ON GovernmentServices FROM user\_name;

PREM LOHIA

PLSQL CURSOR TRIGGERS

PL/SQL UPDATED CODE

Q1 : Write a PL/SQL block to add 10% more interest on loan type = home loan

ANSWER:NORMAL PLSQL

SET SERVEROUTPUT ON

DECLARE

v\_account\_no loan1.account\_no%TYPE;

v\_loan\_type loan1.loan\_type%TYPE;

v\_loan\_no loan1.loan\_no%TYPE;

v\_amount loan1.amount%TYPE;

v\_interest loan1.interest%TYPE;

BEGIN

-- Update interest for home loans

FOR loan\_rec IN (SELECT \* FROM loan1 WHERE loan\_type = 'Home loan') LOOP

v\_account\_no := loan\_rec.account\_no;

v\_loan\_type := loan\_rec.loan\_type;

v\_loan\_no := loan\_rec.loan\_no;

v\_amount := loan\_rec.amount;

v\_interest := loan\_rec.interest;

-- Calculate 10% more interest for home loans

v\_interest := v\_interest + (v\_interest \* 0.1);

-- Update interest rate in loan1 table

UPDATE loan1

SET interest = v\_interest

WHERE account\_no = v\_account\_no;

-- Output the changes made

DBMS\_OUTPUT.PUT\_LINE('Interest rate for ' || v\_loan\_type || ' with Account No ' || v\_account\_no || ' updated to ' || v\_interest);

END LOOP;

-- Display the updated table

FOR loan\_rec IN (SELECT \* FROM loan1) LOOP

DBMS\_OUTPUT.PUT\_LINE('Account No: ' || loan\_rec.account\_no || ', Loan Type: ' || loan\_rec.loan\_type || ', Loan No: ' || loan\_rec.loan\_no || ', Amount: ' || loan\_rec.amount || ', Interest: ' || loan\_rec.interest);

END LOOP;

-- Commit the changes

COMMIT;

EXCEPTION

WHEN OTHERS THEN

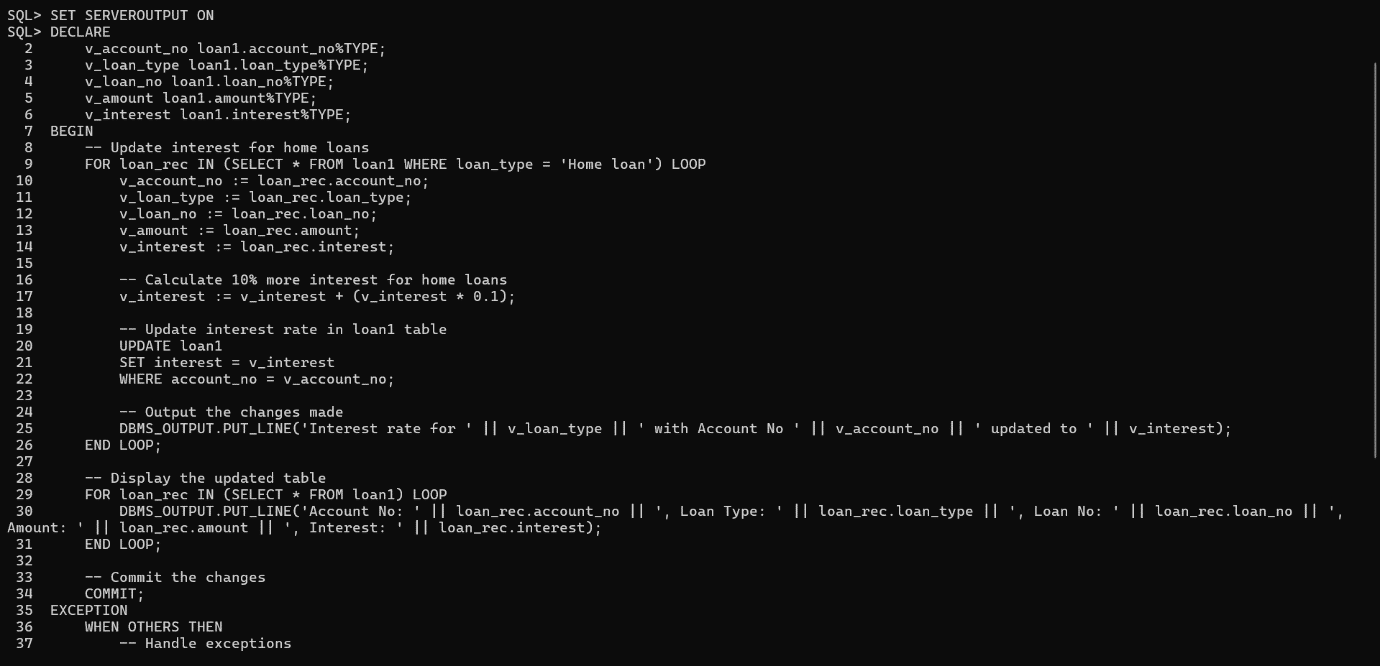
-- Handle exceptions

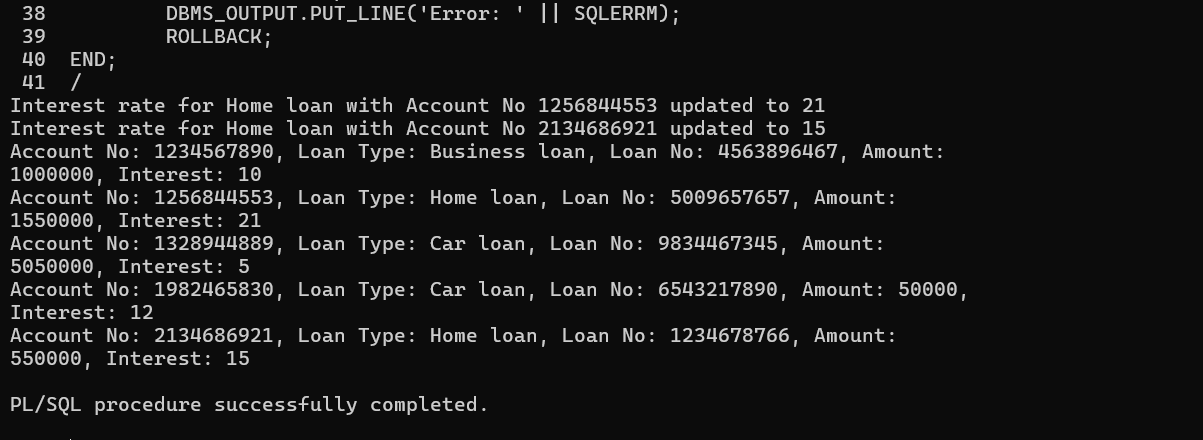
DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

ROLLBACK;

END;

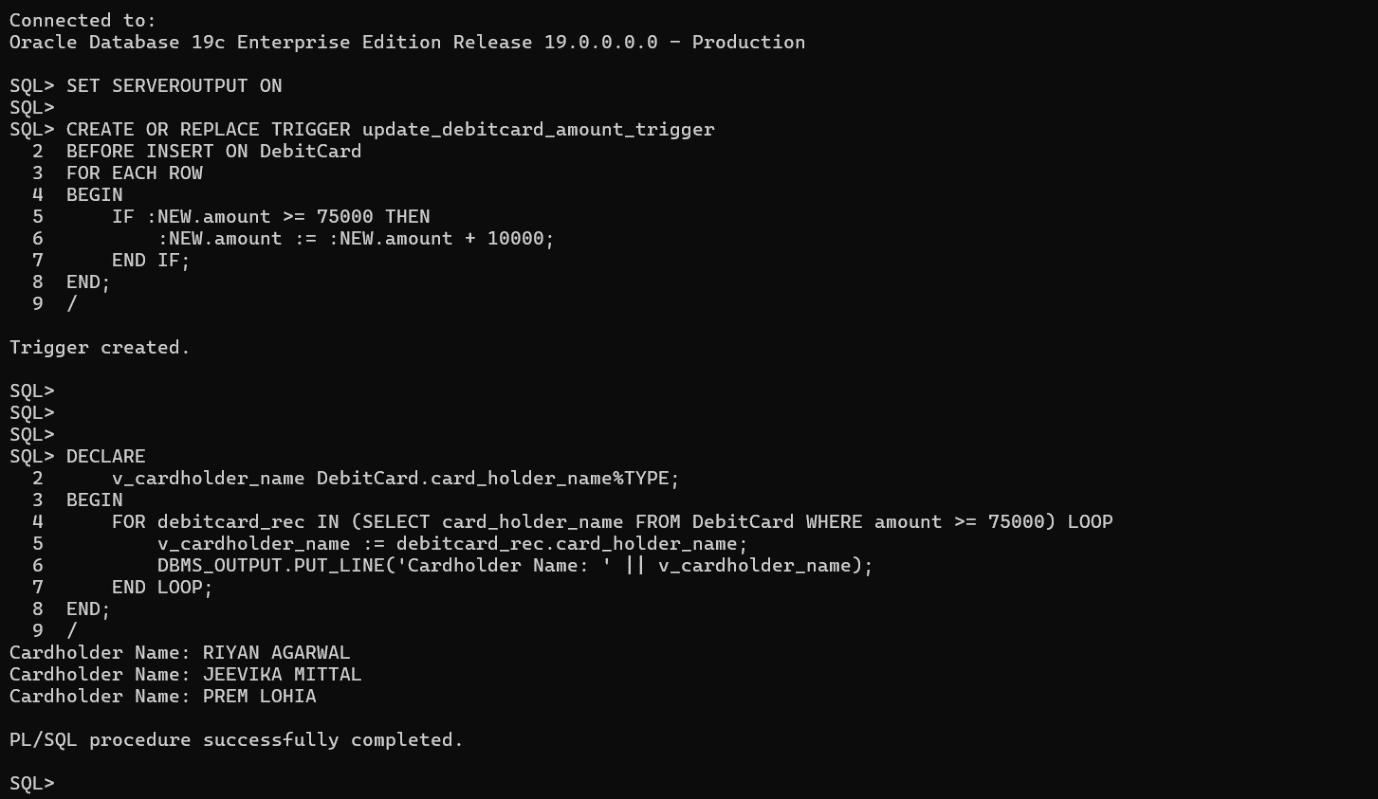
/





Q2 :Write a before trigger to add 10000 ruppee to customer table debit card whose amount is more than 75000

ANSWER: BEFORE INSERT trigger



SET SERVEROUTPUT ON

CREATE OR REPLACE TRIGGER update\_debitcard\_amount\_trigger

BEFORE INSERT ON DebitCard

FOR EACH ROW

BEGIN

IF :NEW.amount >= 75000 THEN

:NEW.amount := :NEW.amount +10000;

END IF;

END;

/

DECLARE

v\_cardholder\_name DebitCard.card\_holder\_name%TYPE;

BEGIN

FOR debitcard\_rec IN (SELECT card\_holder\_name FROM DebitCard WHERE amount >=75000) LOOP

v\_cardholder\_name := debitcard\_rec.card\_holder\_name;

DBMS\_OUTPUT.PUT\_LINE('Cardholder Name: ' || v\_cardholder\_name);

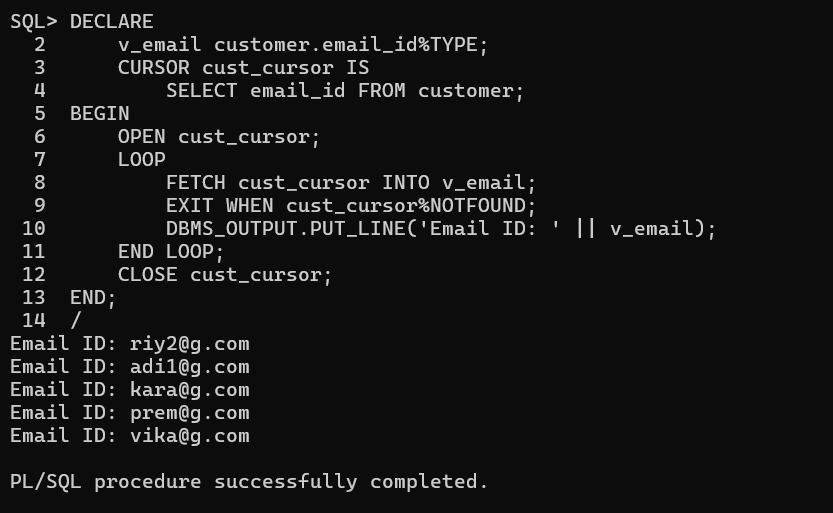
END LOOP;

END;

/

Q3:Write a plsql code of explicit cursor to display all the email\_id af all customer name from table customer

ANSWER: EXPLACIT CURSOR



DECLARE

v\_email customer.email\_id%TYPE;

CURSOR cust\_cursor IS

SELECT email\_id FROM customer;

BEGIN

OPEN cust\_cursor;

LOOP

FETCH cust\_cursor INTO v\_email;

EXIT WHEN cust\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Email ID: ' || v\_email);

END LOOP;

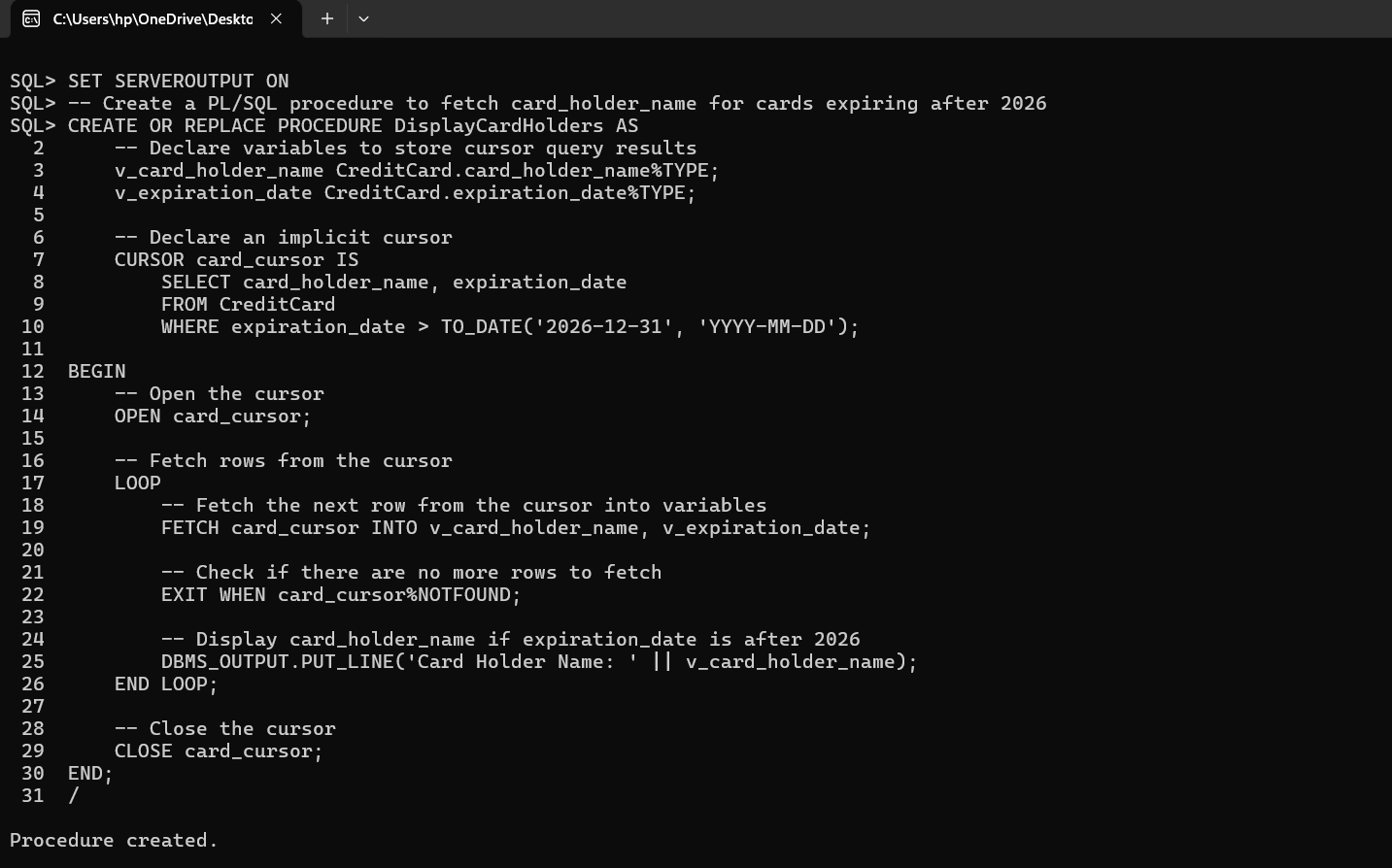
CLOSE cust\_cursor;

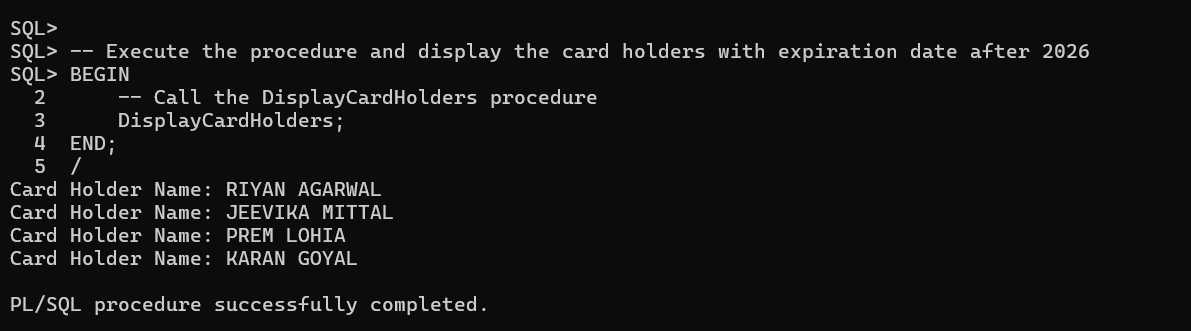
END;

/

Q4:write a plsql code of implicite cursor to diaplay the card\_holder\_name of those people who have expiration\_date after year 2026 and also display result as an output

ANSWER: : IMPLICIT CURSOR





SQL> SET SERVEROUTPUT ON

-- Create a PL/SQL procedure to fetch card\_holder\_name for cards expiring after 2026

SQL> CREATE OR REPLACE PROCEDURE DisplayCardHolders AS

v\_card\_holder\_name CreditCard.card\_holder\_name%TYPE;

v\_expiration\_date CreditCard.expiration\_date%TYPE;

-- Declare an implicit cursor

CURSOR card\_cursor IS

SELECT card\_holder\_name, expiration\_date

FROM CreditCard

WHERE expiration\_date > TO\_DATE('2026-12-31', 'YYYY-MM-DD');

BEGIN

-- Open the cursor

OPEN card\_cursor;

LOOP

-- Fetch the next row from the cursor into variables

FETCH card\_cursor INTO v\_card\_holder\_name, v\_expiration\_date;

-- Check if there are no more rows to fetch

EXIT WHEN card\_cursor%NOTFOUND;

-- Display card holder name if expiration date is after 2026

DBMS\_OUTPUT.PUT\_LINE('Card Holder Name: ' || v\_card\_holder\_name);

END LOOP;

-- Close the cursor

CLOSE card\_cursor;

END;

/

-- Execute the procedure and display the card holders with expiration date after 2026

BEGIN

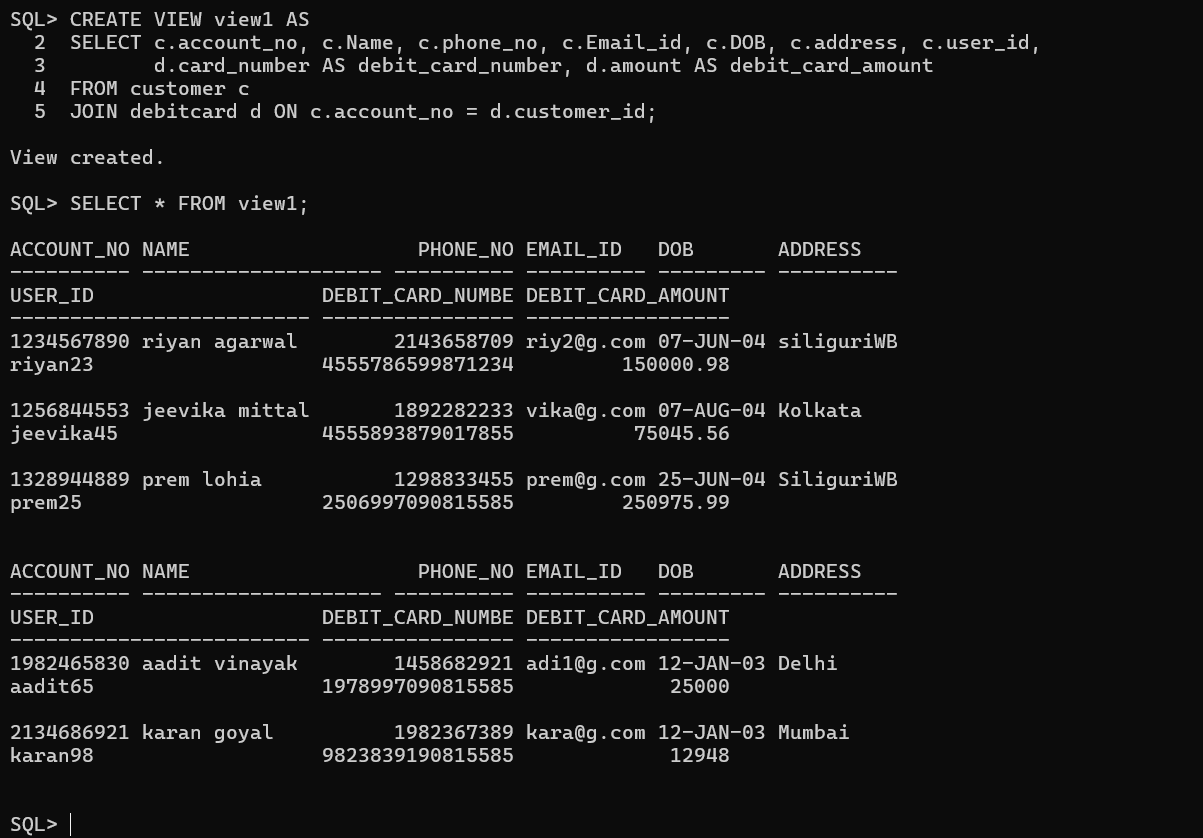
DisplayCardHolders;

END;

/

Q5: create a view and add all column from customer and add card no and amount from debit card and also show the view as output with table name view1

ANSWER: CREATING VIEW1



NORMAL PLSQL ,CURSOR , TRIGGER,VIEW

CHIRANJEEV KUMAR

Q1) write a NORMAL PL/SQL block to change the amount from branch name = navi branch from BanktoBank x table and decrease amount to 1000 from all columns of the table using cursor

SET SERVEROUTPUT ON;

DECLARE

CURSOR c\_bank\_cur IS

SELECT bank\_name, branch, amount, transaction\_date, account\_no

FROM banktobank

WHERE branch = 'NAVI BRANCH'; -- Use 'branch' instead of 'branch\_name'

v\_bank\_name banktobank.bank\_name%TYPE;

v\_branch banktobank.branch%TYPE; -- Correct variable name for branch

v\_amount DECIMAL(15, 2); -- Correct data type declaration for amount

v\_timestamp TIMESTAMP;

v\_account\_number banktobank.account\_no%TYPE; -- Use correct column name

BEGIN

FOR bank\_rec IN c\_bank\_cur LOOP

v\_bank\_name := bank\_rec.bank\_name;

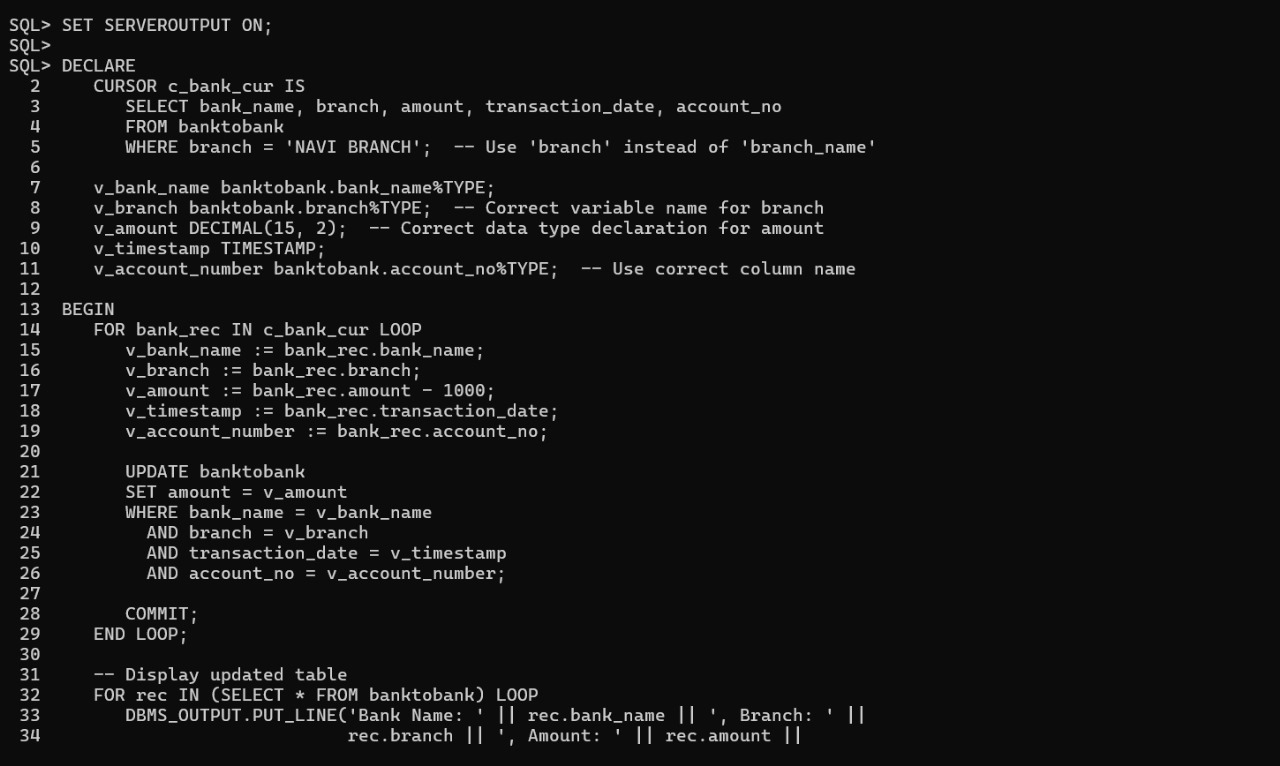
v\_branch := bank\_rec.branch;

v\_amount := bank\_rec.amount - 1000;

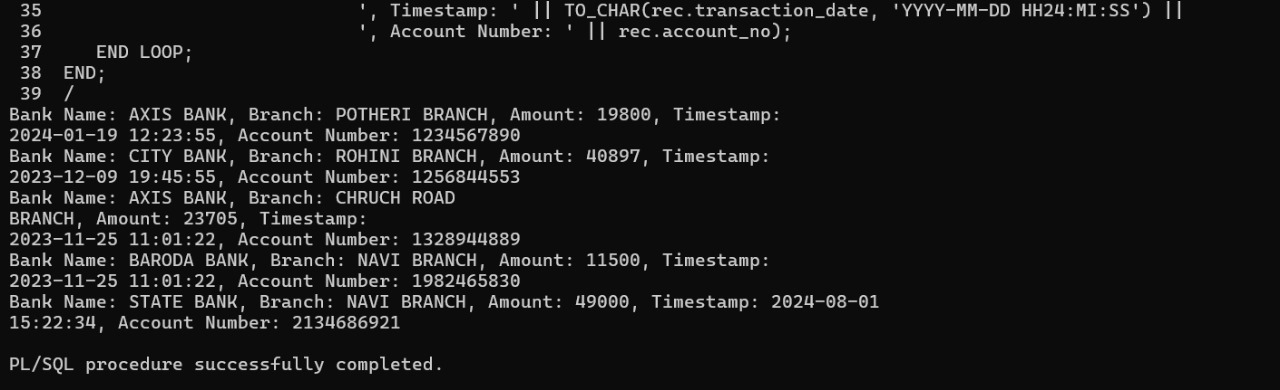
v\_timestamp := bank\_rec.transaction\_date;

v\_account\_number := bank\_rec.account\_n

CODE



OUTPUT



Q2) write a PL/SQL block to change the insurancex table, premium\_amount who have coverage\_details of Hospital bills , Accident bills will increase premium amount by 20 percent more by using trigger

BEFORE trigger

SET SERVEROUTPUT ON;

CREATE OR REPLACE TRIGGER adjust\_premium\_trigger

BEFORE INSERT OR UPDATE ON insurance

FOR EACH ROW

BEGIN

IF :NEW.coverage\_details = 'hospital bills' OR :NEW.coverage\_details = 'Accidents' THEN

:NEW.premium\_amount := :NEW.premium\_amount \* 1.20; -- Increase premium amount

by 20%

END IF;

END;

/

DECLARE

BEGIN

-- Display updated table

FOR rec IN (SELECT \* FROM insurance) LOOP

DBMS\_OUTPUT.PUT\_LINE('Policy Number: ' || rec.policy\_number || ', Customer ID: ' ||

rec.customer\_id || ', Type: ' || rec.insurance\_type || ', Premium Amount: ' ||

rec.premium\_amount || ', Start Date: ' || TO\_CHAR(rec.start\_date, 'YYYY-MM-DD') || ', End

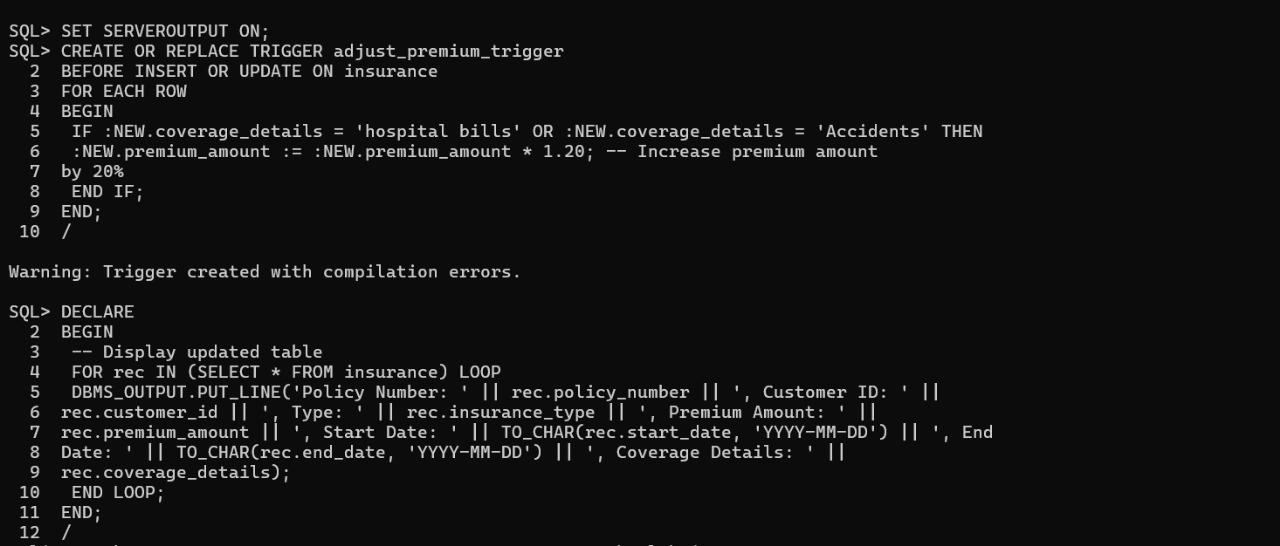
Date: ' || TO\_CHAR(rec.end\_date, 'YYYY-MM-DD') || ', Coverage Details: ' ||

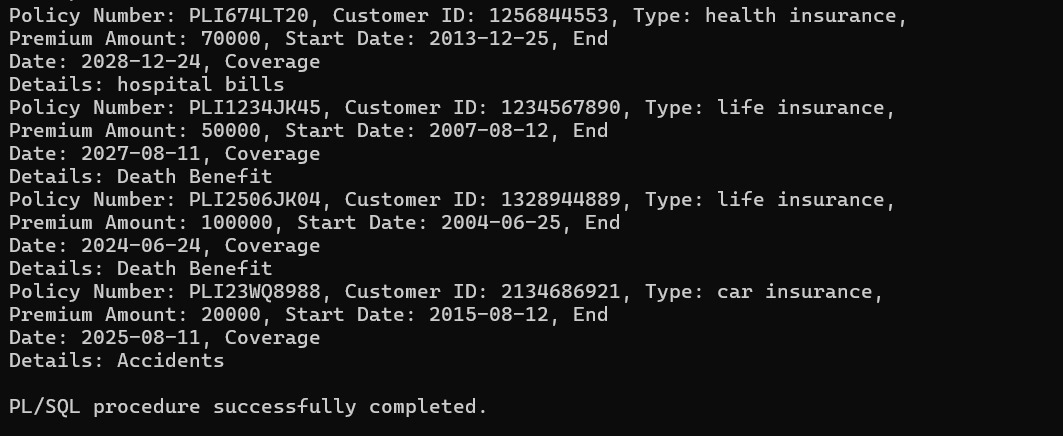
rec.coverage\_details);

END LOOP;

END;

\





Q3) write a plsql code of implicite cursor to diaplay the account \_noe of those who have transaction\_date after year 2023

implicit cursor

SET SERVEROUTPUT ON;

DECLARE

v\_account\_no banktobank.account\_no%TYPE;

BEGIN

-- Open cursor to fetch account numbers with transaction dates after 2023

FOR rec IN (SELECT account\_no FROM banktobank WHERE EXTRACT(YEAR FROM transaction\_date) > 2023) LOOP

v\_account\_no := rec.account\_no;

-- Display account numbers

DBMS\_OUTPUT.PUT\_LINE('Account Number: ' || v\_account\_no);

END LOOP;

END;

/

