WEEK 2 : PL/SQL

Question 1:

**Exercise 1: Control Structures**

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* + **Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* + **Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* + **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

Answer:

CREATE table customer(customer\_id int primary key,

name varchar (50),age number,

balance number,isvip VARCHAR(5) default 'false'

);

create table loan(loan\_id int primary key,

customer\_id references customers(customer\_id),

interest\_rate number(7,2),

due\_date date);

INSERT INTO customer(customer\_id, name, age, balance, isvip)

VALUES (105, 'balaji', 73, 2500.00, 'false');

INSERT INTO customer(customer\_id, name, age, balance, isvip)

VALUES (106, 'suresh', 62, 20000.00, 'false');

INSERT INTO customer(customer\_id, name, age, balance, isvip)

VALUES (107, 'karthik', 30, 15000.00, 'true');

INSERT INTO customer(customer\_id, name, age, balance, isvip)

VALUES (108, 'priya', 28, 3000.00, 'true');

insert into loan(loan\_id, customer\_id, interest\_rate, due\_date)

values (201, 105, 5.5, SYSDATE + 10);

insert into loan(loan\_id, customer\_id, interest\_rate, due\_date)

values (202, 106, 6.0, SYSDATE + 30);

insert into loan(loan\_id, customer\_id, interest\_rate, due\_date)

values (203, 107, 4.5, SYSDATE + 35);

insert into loan(loan\_id, customer\_id, interest\_rate, due\_date)

values (204, 108, 3.5, SYSDATE + 28);

--scenario 1

begin

for rec in (

    select l.loan\_id,l.interest\_rate from loan l JOIN customer c on

    l.customer\_id = c.customer\_id where c.age > 60

)LOOP

update loan set interest\_rate =rec. interest\_rate - 1 where loan\_id = rec.loan\_id;

end loop;

COMMIT;

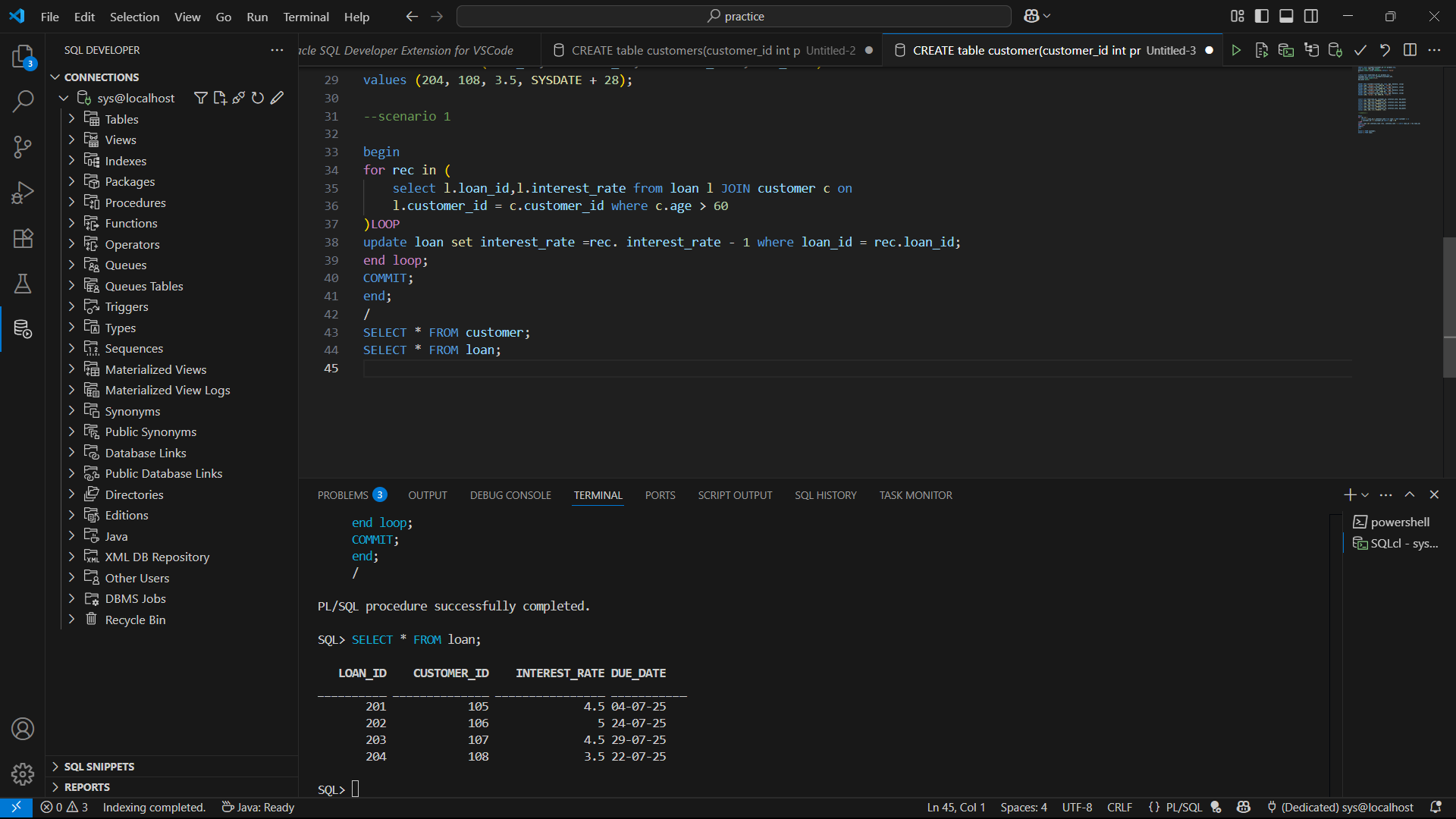
end;

/

SELECT \* FROM customer;

SELECT \* FROM loan;

OUTPUT(scenario 1)



--scenario 2

begin

for rec in (

    Select customer\_id from customer

)LOOP

    update  customer set isvip = 'true' where balance > 10000;

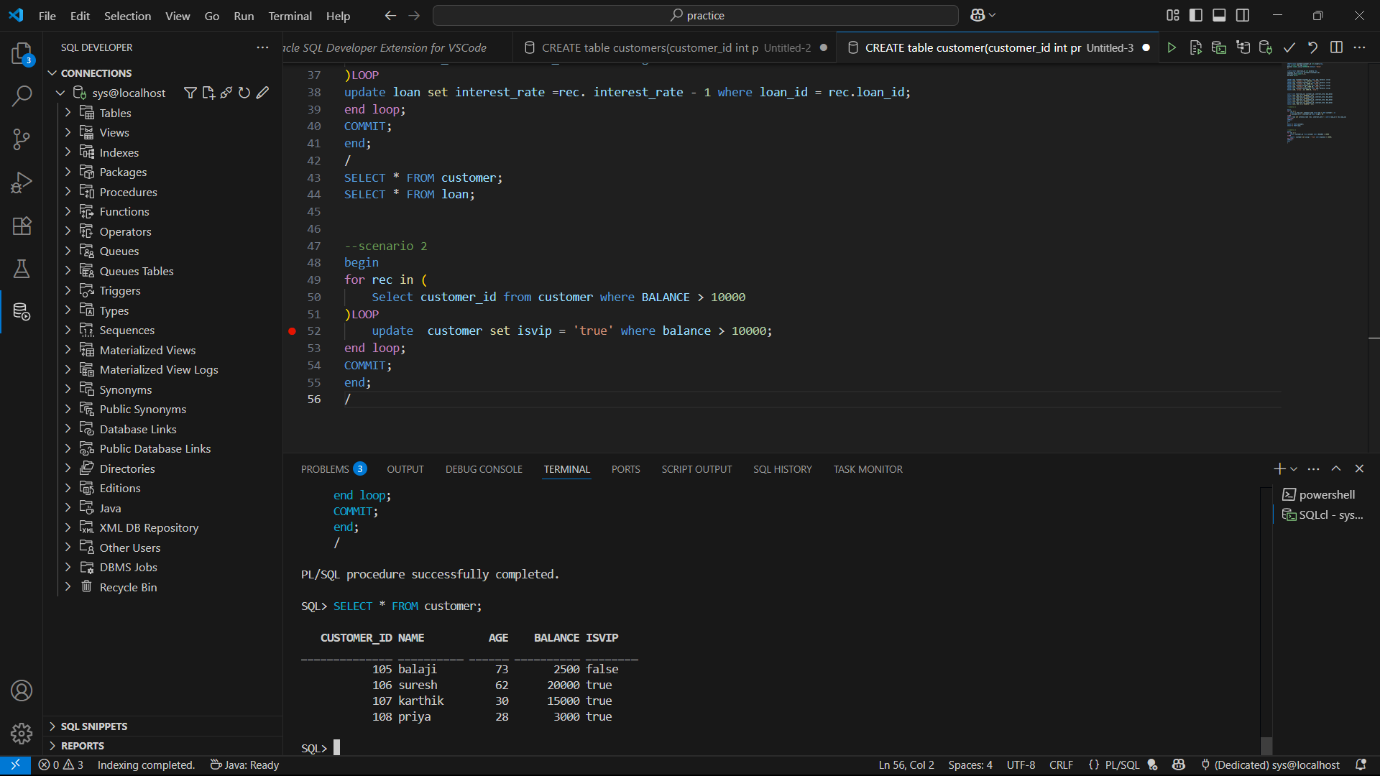
end loop;

COMMIT;

end;

/

OUTPUT(scenario 2)



--scenario 3

begin

for obj in (

    select l.due\_date,c.name from loan l JOIN customer c on l.CUSTOMER\_ID = c.customer\_id

    where l.due\_date < SYSDATE + 30

)LOOP

DBMS\_OUTPUT.PUT\_LINE('remainder: '||obj.name||' loans due in the next date on '||obj.due\_date );

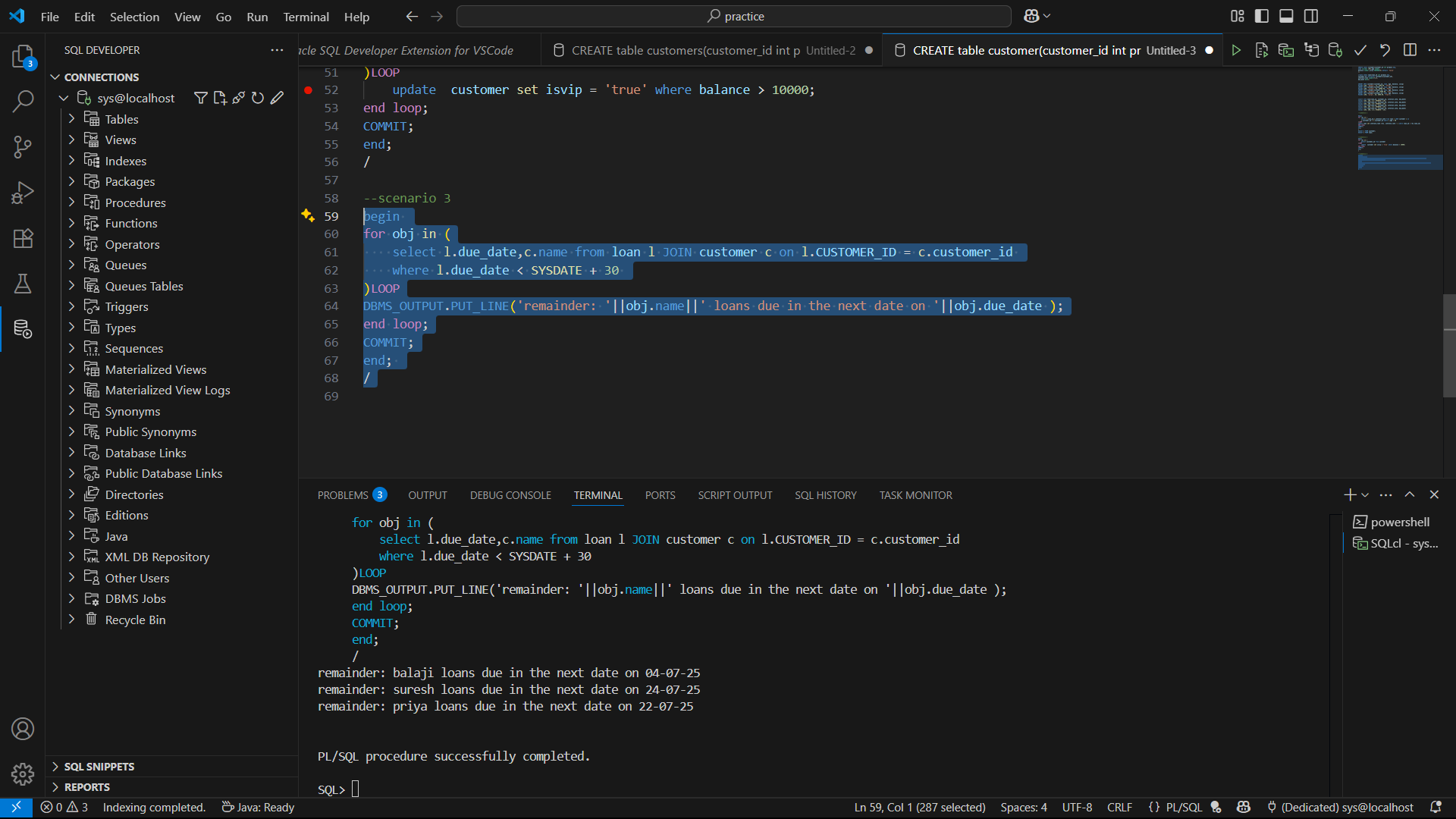
end loop;

COMMIT;

end;

/

OUTPUT(scenario 3)



Question 2:

**Exercise 3: Stored Procedures**

**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.

**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.

**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.

**Answer:**

create table bank(

account\_id number(10) not null,

account\_type varchar2(20) not null,

balance number(15,2) default 0 not null

);

insert into bank values (10001, 'savings', 15000.00);

insert into bank values (10002, 'current', 50000.00);

insert into bank values (10003, 'savings', 20000.00);

insert into bank values (10004, 'current', 75000.00);

--scenario 1

create or replace PROCEDURE ProcessMonthlyInterest IS

begin

update bank set balance=balance+balance\*0.01 where account\_type='savings';

COMMIT;

end ;

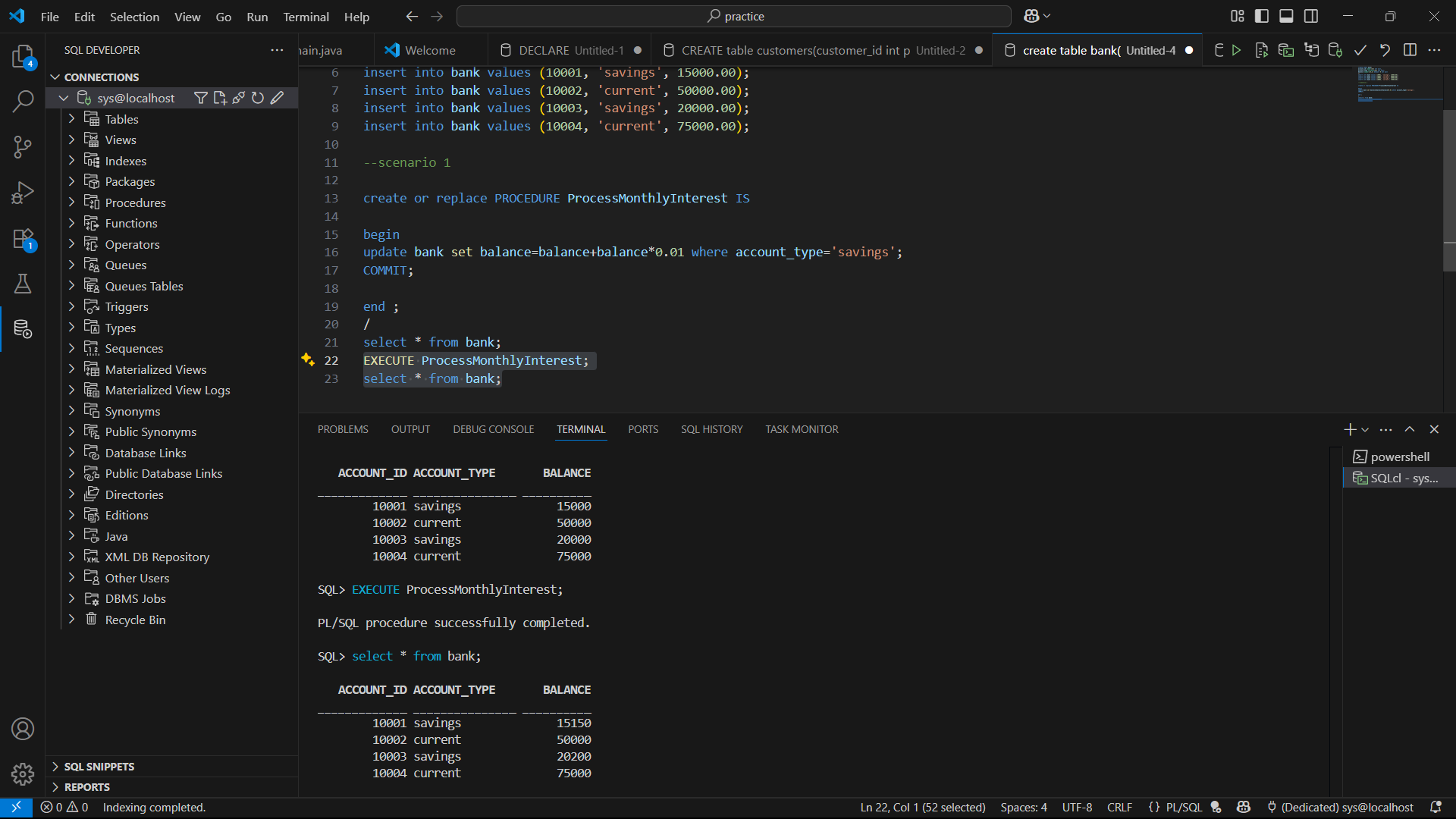
/

select \* from bank;

EXECUTE ProcessMonthlyInterest;

select \* from bank;

Output:



--scenario 2

create table employee(

employee\_id number(10) primary key,

name varchar2(50),

department varchar2(30),

salary number

);

INSERT into employee values (101, 'balaji', 'HR', 50000);

INSERT into employee values (102, 'suresh', 'IT', 60000);

INSERT into employee values (103, 'ramesh', 'Finance', 55000);

INSERT into employee values (104, 'karthik', 'IT', 70000);

create or replace procedure UpdateEmployeeBonus(bonus\_percentage  NUMBER,department  VARCHAR) is

begin

update employee set salary =salary + (salary \* bonus\_percentage/100)  where department = department;

commit;

end UpdateEmployeeBonus;

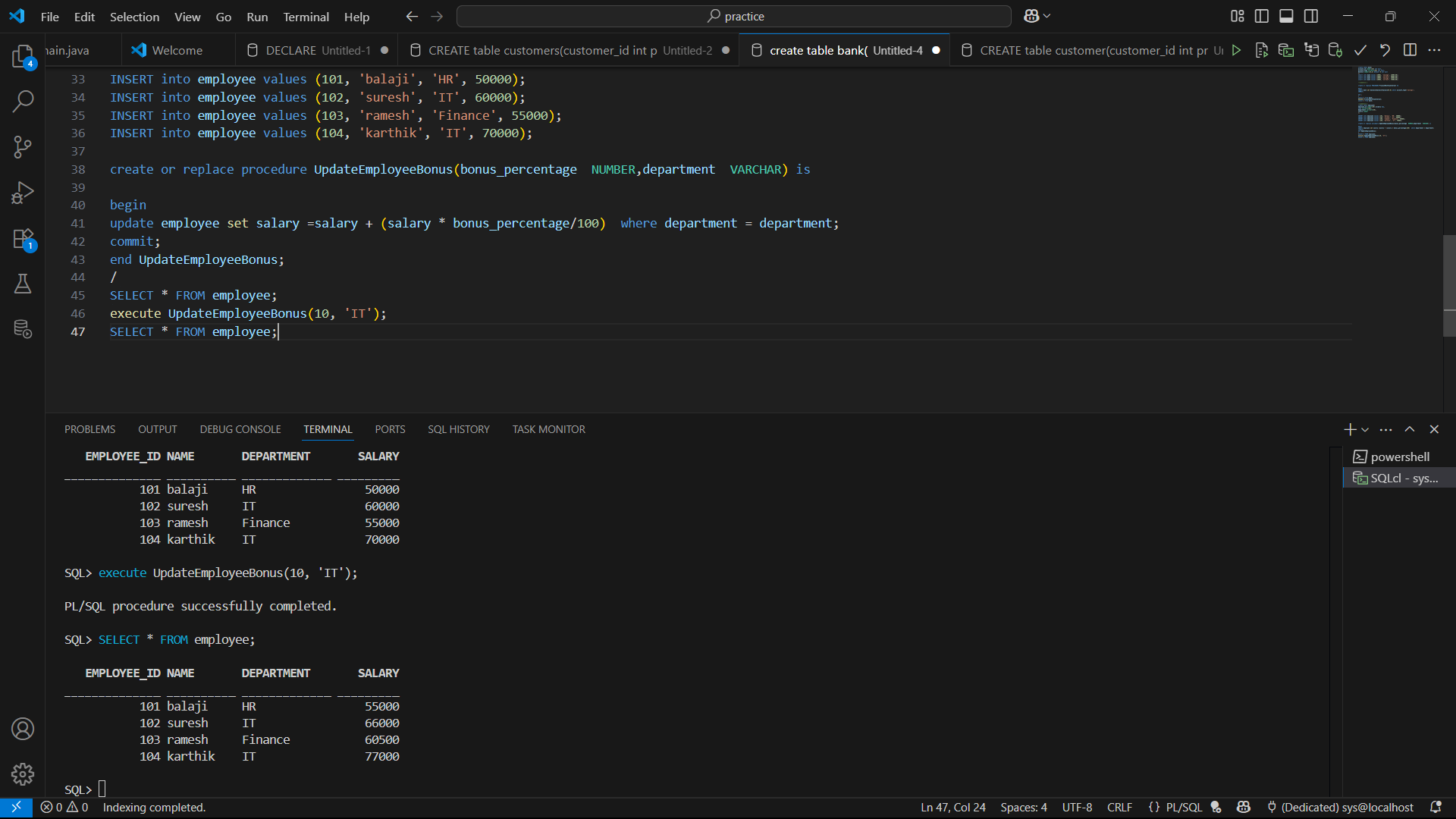
/

SELECT \* FROM employee;

execute UpdateEmployeeBonus(10, 'IT');

SELECT \* FROM employee;

Output:



--scenario 3

create or replace procedure TransferFunds(

    faccount\_id IN NUMBER,

    taccount\_id IN NUMBER,

    amount IN NUMBER

) IS

    fbal NUMBER;

begin

select balance into fbal from bank where account\_id = faccount\_id

for update;

if fbal < amount then

    raise\_application\_error(-20001, 'Insufficient funds in the sender account');

end if;

update bank set balance = balance - amount where account\_id = faccount\_id;

update bank set balance = balance + amount where account\_id = taccount\_id;

commit;

end ;

/

select \* from bank;

execute TransferFunds(10002, 10003, 50000);

select \* from bank;

Output:

