DBMS ASSIGNMENT: 2

Name: Poojan Gandhi

Enroll No: AU1940125

Faculty: Shefali Naik

- One txt file is also submitted.
- The txt file and this doc both contains data manipulation commands for all questions so we can refer any of them.
- This doc also have images of execution part for each question.
- The pdf (which is same as doc) is also submitted so that it will be more convenient to read.

Bank Table creation and data insertion

create table branch(branch_name varchar(50) primary key,branch_city varchar(50),assets int);

create table account(account_number varchar(5) primary key ,branch_name varchar(50) references branch(branch_name),balance int);

create table loan(loan_number varchar(4) primary key ,branch_name varchar(50) references branch(branch_name),amount int);

create table customer(customer_name varchar(50) primary key,customer_street varchar(50),customer_city varchar(50));

create table borrower(customer_name varchar(50) references customer(customer_name),loan_number varchar(4) references loan(loan_number),primary key(customer_name,loan_number));

create table depositor(customer_name varchar(50) references customer(customer_name),account_number varchar(5) references account(account_number),primary key(customer_name,account_number));

BRANCH TABLE

```
INSERT INTO branch VALUES ('Brighton', 'Brooklyn', 7100000);
INSERT INTO branch VALUES ('Downtown', 'Brooklyn', 9000000);
INSERT INTO branch VALUES ('Mianus', 'Horseneck', 400000);
INSERT INTO branch VALUES ('North Town', 'Rye', 3700000);
INSERT INTO branch VALUES ('Perryridge', 'Horseneck', 1700000);
INSERT INTO branch VALUES ('Pownal', 'Bennington', 300000);
INSERT INTO branch VALUES ('Redwood', 'Palo Alto', 2100000);
INSERT INTO branch VALUES ('Round Hill', 'Horseneck', 8000000);
```

LOAN TABLE

```
INSERT INTO loan VALUES ('L-11', 'Round Hill', 900);
INSERT INTO loan VALUES ('L-14', 'Downtown', 1500);
INSERT INTO loan VALUES ('L-15', 'Perryridge', 1500);
INSERT INTO loan VALUES ('L-16', 'Perryridge', 1300);
INSERT INTO loan VALUES ('L-17', 'Downtown', 1000);
INSERT INTO loan VALUES ('L-23', 'Redwood', 2000);
INSERT INTO loan VALUES ('L-93', 'Mianus', 500);
```

CUSTOMER TABLE

```
INSERT INTO customer VALUES ('Adams', 'Spring', 'Pittsfield');
INSERT INTO customer VALUES ('Brooks', 'Senator', 'Brooklyn');
INSERT INTO customer VALUES ('Curry', 'North', 'Rye');
INSERT INTO customer VALUES ('Glenn', 'Sand Hill', 'Woodside');
INSERT INTO customer VALUES ('Green', 'Walnut', 'Stamford');
INSERT INTO customer VALUES ('Hayes', 'Main', 'Harrison');
INSERT INTO customer VALUES ('Johnson', 'Alma', 'Palo Alto');
INSERT INTO customer VALUES ('Jones', 'Main', 'Harrison');
```

INSERT INTO customer VALUES ('Lindsay', 'Park', 'Pittsfield');

INSERT INTO customer VALUES ('Smith', 'North', 'Rye');

INSERT INTO customer VALUES ('Turner', 'Putnam', 'Stamford');

INSERT INTO customer VALUES ('Williams', 'Nassau', 'Princeton');

BORROWER TABLE

INSERT INTO borrower VALUES ('Adams', 'L-16');

INSERT INTO borrower VALUES ('Curry', 'L-93');

INSERT INTO borrower VALUES ('Hayes', 'L-15');

INSERT INTO borrower VALUES ('Johnson', 'L-14');

INSERT INTO borrower VALUES ('Jones', 'L-17');

INSERT INTO borrower VALUES ('Smith', 'L-11');

INSERT INTO borrower VALUES ('Smith', 'L-23');

INSERT INTO borrower VALUES ('Williams', 'L-17');

ACCOUNT TABLE

INSERT INTO account VALUES ('A-101', 'Downtown', 500);

INSERT INTO account VALUES ('A-102', 'Perryridge', 400);

INSERT INTO account VALUES ('A-201', 'Brighton', 900);

INSERT INTO account VALUES ('A-215', 'Mianus', 700);

INSERT INTO account VALUES ('A-217', 'Brighton', 750);

INSERT INTO account VALUES ('A-222', 'Redwood', 700);

INSERT INTO account VALUES ('A-305', 'Round Hill', 350);

DEPOSITOR TABLE

INSERT INTO depositor VALUES ('Hayes', 'A-102');

INSERT INTO depositor VALUES ('Johnson', 'A-101');

INSERT INTO depositor VALUES ('Johnson', 'A-201');

INSERT INTO depositor VALUES ('Jones', 'A-217');

INSERT INTO depositor VALUES ('Lindsay', 'A-222');

```
INSERT INTO depositor VALUES ('Smith', 'A-215');

INSERT INTO depositor VALUES ('Turner', 'A-305');
```

1. Create a procedure which will display loan details of borrowers in the following format.

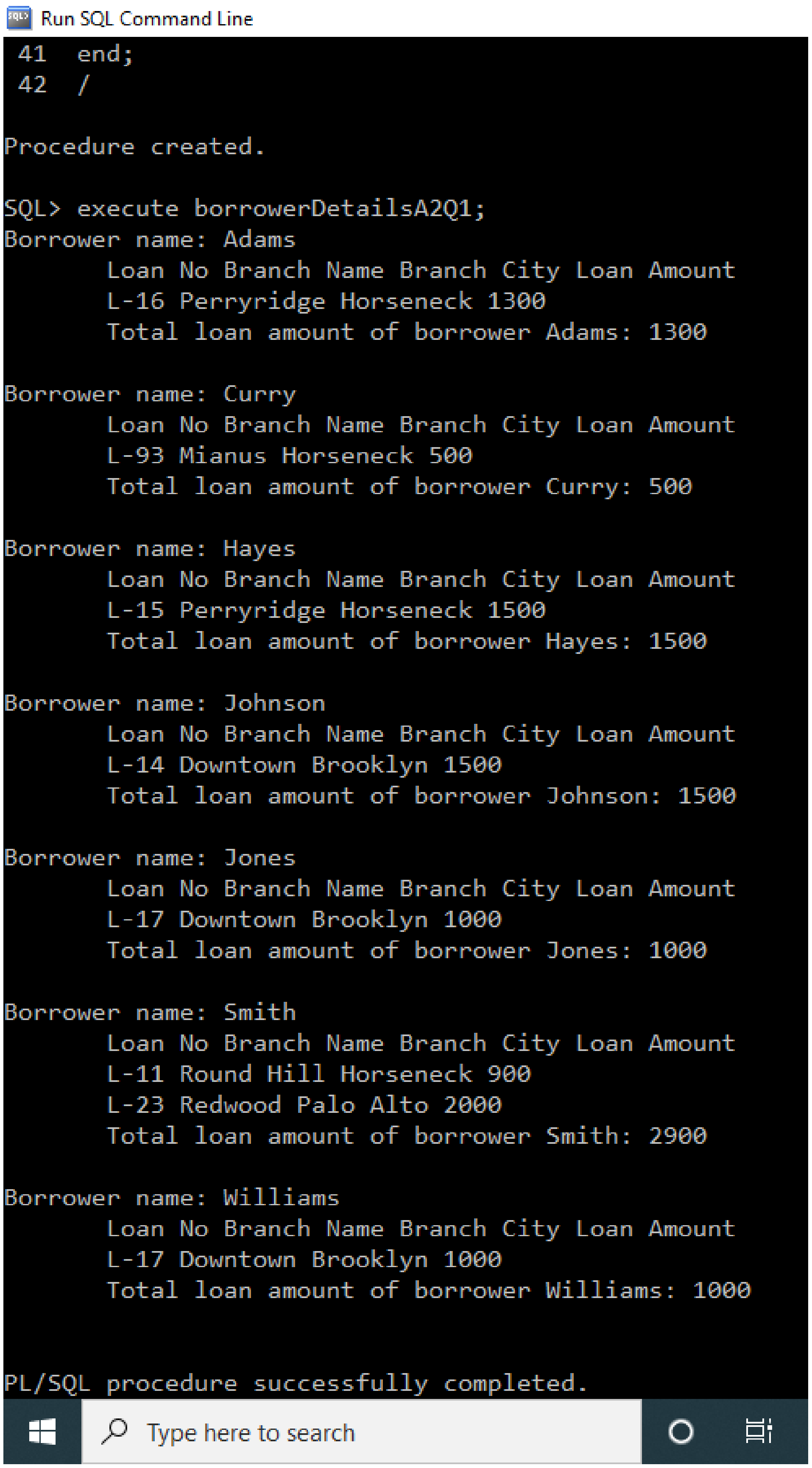
Procedure

```
create or replace procedure borrowerDetailsA2Q1 as

        totalLoanAmount int:=0;

        cursor cur_borrower is select distinct customer_name from borrower;

        rec_borrower cur_borrower%rowtype;

        cursor cur_loanNumber(cname customer.customer_name%type) is select * from borrower where customer_name=cname;

        rec_loanNumber cur_loanNumber%rowtype;

        cursor cur_loan(ln loan.loan_number%type) is select * from loan where loan_number=ln;

        rec_loan cur_loan%rowtype;

        cursor cur_branchCity(bn branch.branch_name%type) is select branch_city from branch where branch_name=bn;

        rec_branchCity cur_branchCity%rowtype;

begin

        for rec_borrower in cur_borrower loop

                totalLoanAmount:=0;

                dbms_output.put_line('Borrower name: '||rec_borrower.customer_name);

                dbms_output.put_line('      Loan No                    Branch Name                    Branch City
Loan Amount');

                for rec_loanNumber in cur_loanNumber(rec_borrower.customer_name) loop

                        for rec_loan in cur_loan(rec_loanNumber.loan_number) loop

                                --BRANCH_NAME IS PRIMAY KEY IN BRANCH THEREFORE IT IS UNIQUE

                                for rec_branchCity in cur_branchCity(rec_loan.branch_name) loop

                                        dbms_output.put_line('      '||rec_loan.loan_number||' '||rec_loan.branch_name||'
'||rec_branchCity.branch_city||' '|| rec_loan.amount);

                                end loop;

                                totalLoanAmount:=totalLoanAmount+rec_loan.amount;

                        end loop;
                end loop;

                if(totalLoanAmount!=0) then

                        dbms_output.put_line('      Total loan amount of borrower '||rec_borrower.customer_name||':
'||totalLoanAmount||chr(10));

                end if;

        end loop;

end;

/
```

Execution

Execute borrowerDetailsA2Q1;

IMAGE



Run SQL Command Line

```
 41  end;
 42  /

Procedure created.

SQL> execute borrowerDetailsA2Q1;
Borrower name: Adams
       Loan No Branch Name Branch City Loan Amount
       L-16 Perryridge Horseneck 1300
       Total loan amount of borrower Adams: 1300

Borrower name: Curry
       Loan No Branch Name Branch City Loan Amount
       L-93 Mianus Horseneck 500
       Total loan amount of borrower Curry: 500

Borrower name: Hayes
       Loan No Branch Name Branch City Loan Amount
       L-15 Perryridge Horseneck 1500
       Total loan amount of borrower Hayes: 1500

Borrower name: Johnson
       Loan No Branch Name Branch City Loan Amount
       L-14 Downtown Brooklyn 1500
       Total loan amount of borrower Johnson: 1500

Borrower name: Jones
       Loan No Branch Name Branch City Loan Amount
       L-17 Downtown Brooklyn 1000
       Total loan amount of borrower Jones: 1000

Borrower name: Smith
       Loan No Branch Name Branch City Loan Amount
       L-11 Round Hill Horseneck 900
       L-23 Redwood Palo Alto 2000
       Total loan amount of borrower Smith: 2900

Borrower name: Williams
       Loan No Branch Name Branch City Loan Amount
       L-17 Downtown Brooklyn 1000
       Total loan amount of borrower Williams: 1000


PL/SQL procedure successfully completed.
```

2. Create a procedure which will display city-wise branch-wise loan details of borrowers in the following format.

Procedure

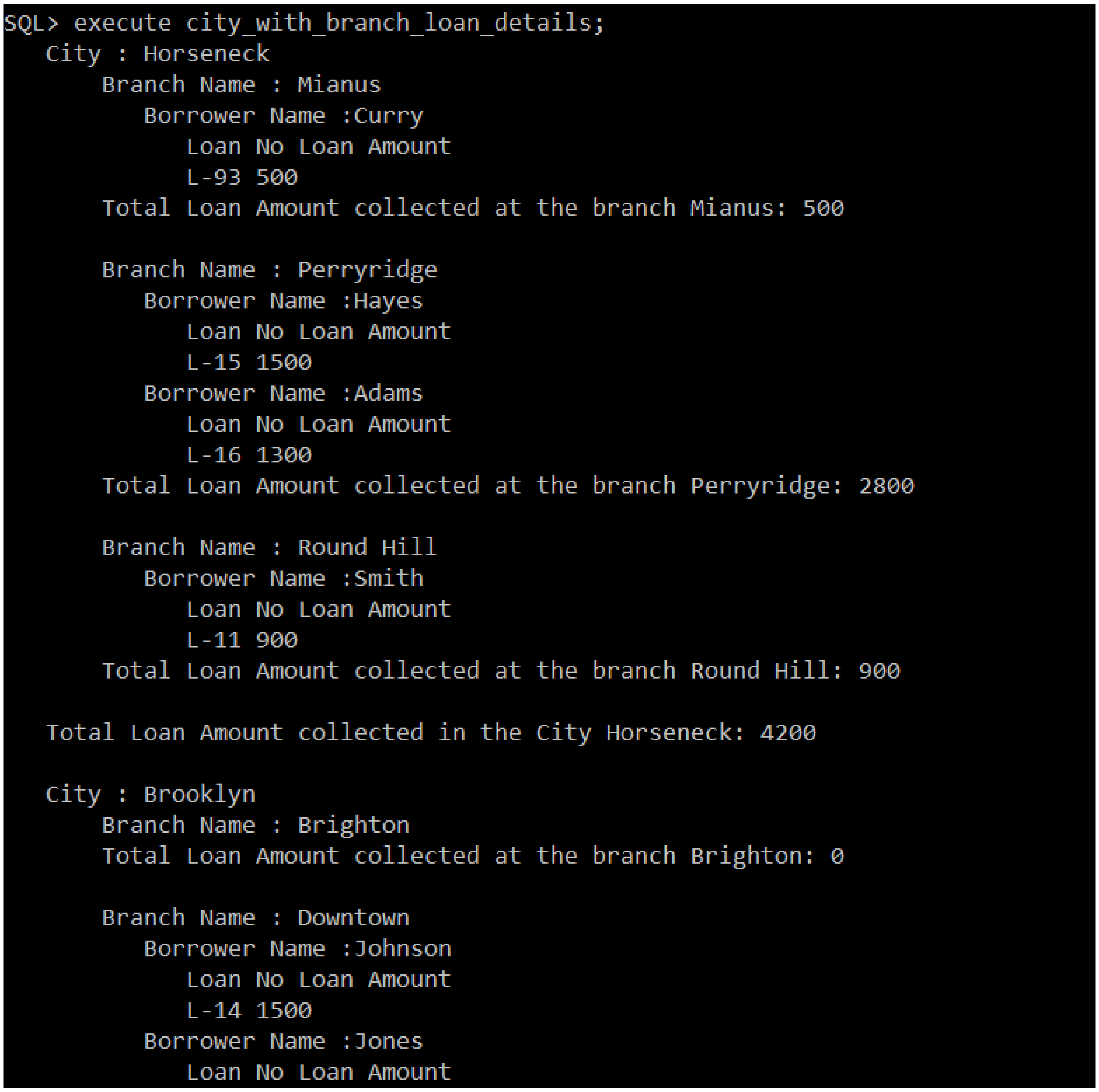
```
create or replace procedure city_with_branch_loan_details as

        cityAmount int:=0;

        branchAmount int:=0;

        totalAmount int:=0;

        cursor cur_city is select distinct branch_city from branch;

        rec_city cur_city%rowtype;

        cursor cur_branch(ct branch.branch_city%type) is select branch_name from branch where branch_city=ct;

        rec_branch cur_branch%rowtype;

        cursor cur_customer(bn branch.branch_name%type) is select b.customer_name,b.loan_number from borrower b,loan l where
l.branch_name=bn and l.loan_number=b.loan_number;

        rec_customer cur_customer%rowtype;

        cursor cur_loanNumber(ln loan.loan_number%type) is select * from loan where loan_number=ln;

        rec_loanNumber cur_loanNumber%rowtype;

begin

        for rec_city in cur_city loop

                dbms_output.put_line(rpad(' ',3)||'City : '|| rec_city.branch_city);

                cityAmount:=0;

                for rec_branch in cur_branch(rec_city.branch_city) loop

                        branchAmount:=0;

                        dbms_output.put_line(rpad(' ',7)||'Branch Name : '|| rec_branch.branch_name);

                        for rec_customer in cur_customer(rec_branch.branch_name) loop

                                        dbms_output.put_line(rpad(' ',10)||'Borrower Name :'||
rec_customer.customer_name);

                                        dbms_output.put_line(rpad(' ',13)||'Loan No         Loan Amount');

                                        for rec_loanNumber in cur_loanNumber(rec_customer.loan_number) loop

                                                dbms_output.put_line(rpad('
',13)||rec_loanNumber.loan_number||' ' || rec_loanNumber.amount);
```

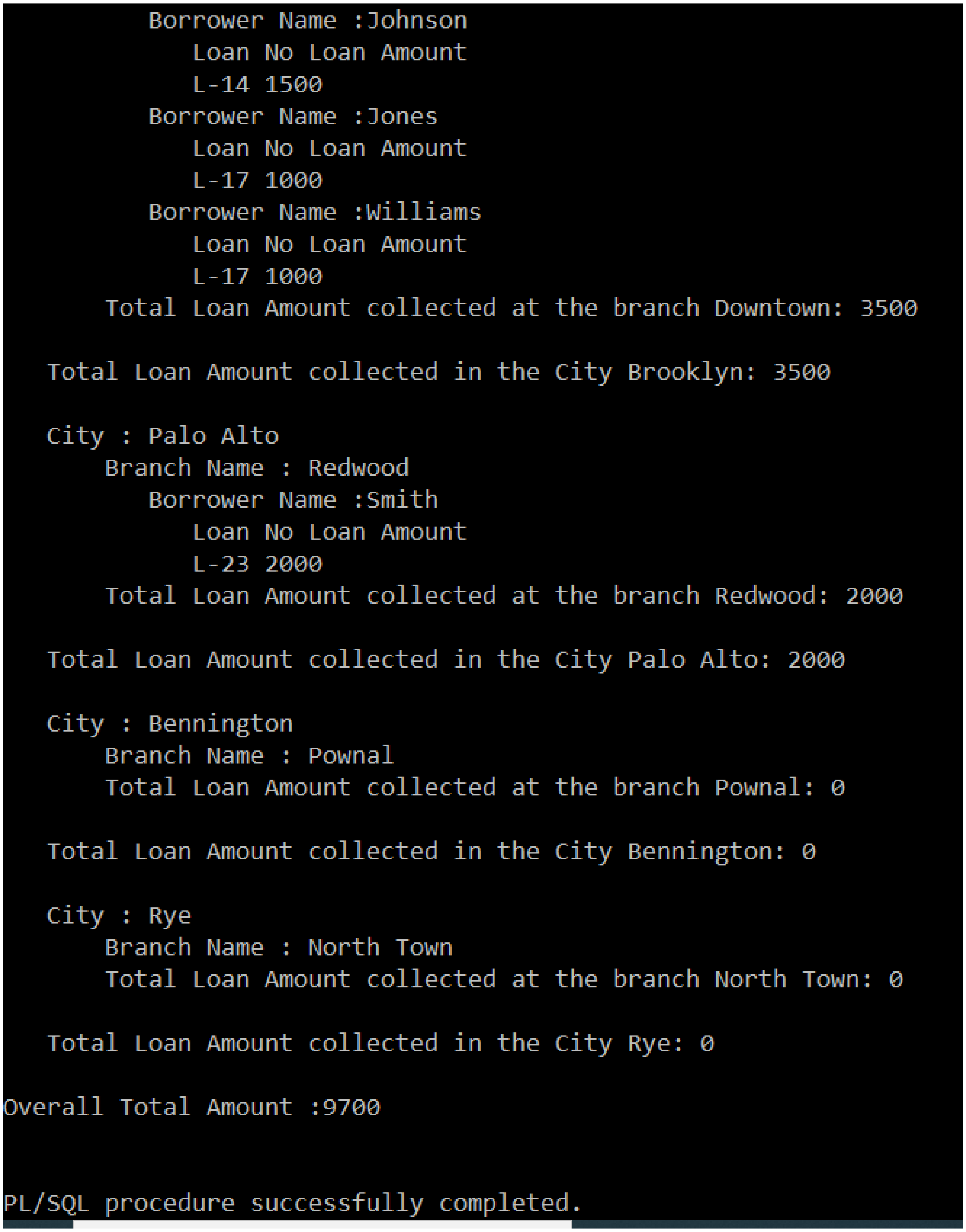
```
                                                                        branchAmount:= branchAmount+rec_loanNumber.amount;


                                                    end loop;

                                    end loop;

                                    dbms_output.put_line(rpad(' ',7)||'Total Loan Amount collected at the branch
'||rec_branch.branch_name||': '|| branchAmount||chr(10));

                                    cityAmount:=cityAmount+branchAmount;

                        end loop;

                        dbms_output.put_line(rpad(' ',3)||'Total Loan Amount collected in the City '||rec_city.branch_city||': '||
cityAmount||chr(10));

                        totalAmount:=totalAmount+cityAmount;

            end loop;

            dbms_output.put_line('Overall Total Amount :'||totalAmount||chr(10));

end;

/
```

Execution

```
execute city_with_branch_loan_details;
```

Image


```
SQL> execute city_with_branch_loan_details;
   City : Horseneck
       Branch Name : Mianus
          Borrower Name :Curry
             Loan No Loan Amount
             L-93 500
       Total Loan Amount collected at the branch Mianus: 500

       Branch Name : Perryridge
          Borrower Name :Hayes
             Loan No Loan Amount
             L-15 1500
          Borrower Name :Adams
             Loan No Loan Amount
             L-16 1300
       Total Loan Amount collected at the branch Perryridge: 2800

       Branch Name : Round Hill
          Borrower Name :Smith
             Loan No Loan Amount
             L-11 900
       Total Loan Amount collected at the branch Round Hill: 900

   Total Loan Amount collected in the City Horseneck: 4200

   City : Brooklyn
       Branch Name : Brighton
       Total Loan Amount collected at the branch Brighton: 0

       Branch Name : Downtown
          Borrower Name :Johnson
             Loan No Loan Amount
             L-14 1500
          Borrower Name :Jones
             Loan No Loan Amount
```



```
         Borrower Name :Johnson
            Loan No Loan Amount
            L-14 1500
         Borrower Name :Jones
            Loan No Loan Amount
            L-17 1000
         Borrower Name :Williams
            Loan No Loan Amount
            L-17 1000
      Total Loan Amount collected at the branch Downtown: 3500

   Total Loan Amount collected in the City Brooklyn: 3500

   City : Palo Alto
      Branch Name : Redwood
         Borrower Name :Smith
            Loan No Loan Amount
            L-23 2000
      Total Loan Amount collected at the branch Redwood: 2000

   Total Loan Amount collected in the City Palo Alto: 2000

   City : Bennington
      Branch Name : Pownal
      Total Loan Amount collected at the branch Pownal: 0

   Total Loan Amount collected in the City Bennington: 0

   City : Rye
      Branch Name : North Town
      Total Loan Amount collected at the branch North Town: 0

   Total Loan Amount collected in the City Rye: 0

Overall Total Amount :9700


PL/SQL procedure successfully completed.
```

3. Create a procedure with parameters city name and branch name. Display records of borrowers of that city and branch in the following format.

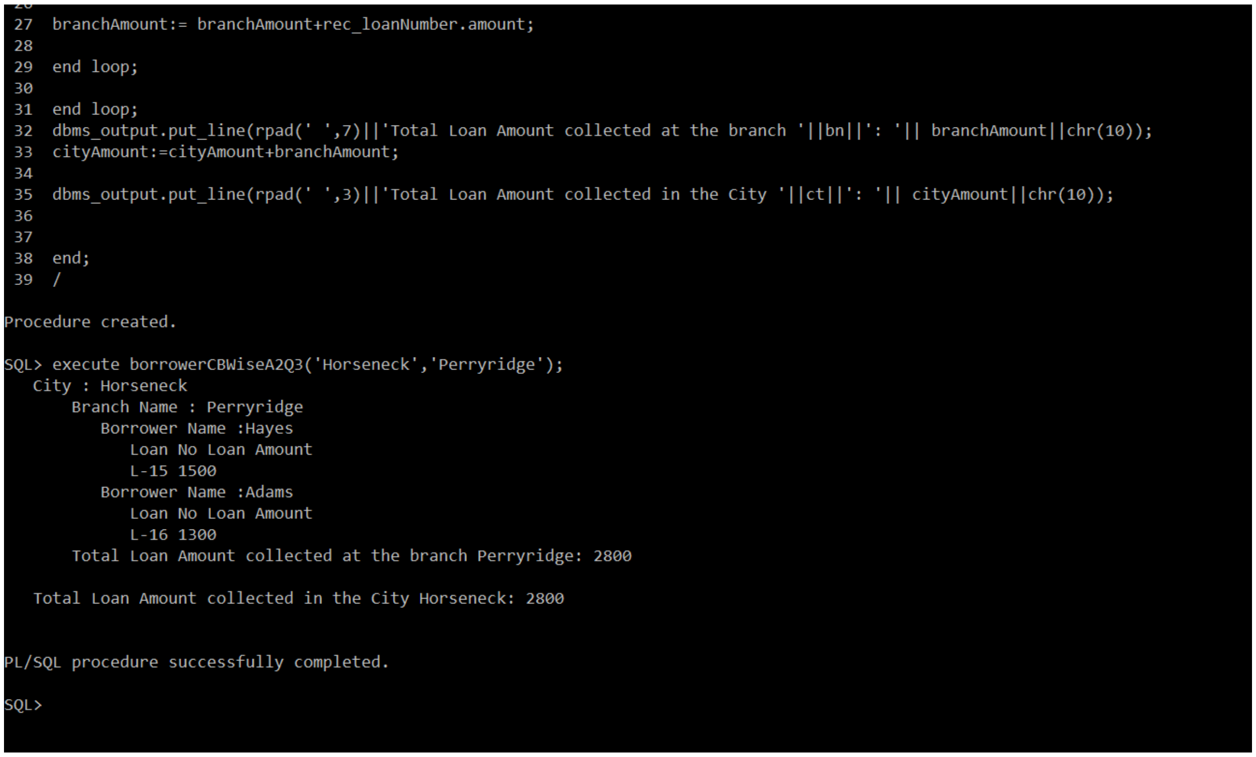
Procedure

```
create or replace procedure borrowerCBWiseA2Q3(ct branch.branch_city%type, bn branch.branch_name%type) as
        cityAmount int:=0;
        branchAmount int:=0;
        cursor cur_customer is select b.customer_name,b.loan_number from borrower b,loan l where l.branch_name=bn and
l.loan_number=b.loan_number;
        rec_customer cur_customer%rowtype;
        cursor cur_loanNumber(ln loan.loan_number%type) is select * from loan where loan_number=ln;
        rec_loanNumber cur_loanNumber%rowtype;
begin
                dbms_output.put_line(rpad(' ',3)||'City : '|| ct);
                cityAmount:=0;
                branchAmount:=0;
                dbms_output.put_line(rpad(' ',7)||'Branch Name : '|| bn);
                for rec_customer in cur_customer loop
                        dbms_output.put_line(rpad(' ',10)||'Borrower Name :'|| rec_customer.customer_name);
                        dbms_output.put_line(rpad(' ',13)||'Loan No          Loan Amount');
                        for rec_loanNumber in cur_loanNumber(rec_customer.loan_number) loop
                                dbms_output.put_line(rpad(' ',13)||rec_loanNumber.loan_number||' ' ||
rec_loanNumber.amount);
                                branchAmount:= branchAmount+rec_loanNumber.amount;
                        end loop;
                end loop;
                dbms_output.put_line(rpad(' ',7)||'Total Loan Amount collected at the branch '||bn||': '|| branchAmount||chr(10));
                cityAmount:=cityAmount+branchAmount;
                dbms_output.put_line(rpad(' ',3)||'Total Loan Amount collected in the City '||ct||': '|| cityAmount||chr(10));
end;
/
```

Execution

```
execute borrowerCBWiseA2Q3('Horseneck','Perryridge');
```

Image



```
branchAmount:= branchAmount+rec_loanNumber.amount;

end loop;

end loop;
dbms_output.put_line(rpad(' ',7)||'Total Loan Amount collected at the branch '||bn||': '|| branchAmount||chr(10));
cityAmount:=cityAmount+branchAmount;

dbms_output.put_line(rpad(' ',3)||'Total Loan Amount collected in the City '||ct||': '|| cityAmount||chr(10));


end;
/

Procedure created.

SQL> execute borrowerCBWiseA2Q3('Horseneck','Perryridge');
   City : Horseneck
       Branch Name : Perryridge
          Borrower Name :Hayes
             Loan No Loan Amount
             L-15 1500
          Borrower Name :Adams
             Loan No Loan Amount
             L-16 1300
       Total Loan Amount collected at the branch Perryridge: 2800

   Total Loan Amount collected in the City Horseneck: 2800


PL/SQL procedure successfully completed.

SQL>
```

4. Write a procedure to display details of the customers who are depositors as well as borrowers

Procedure

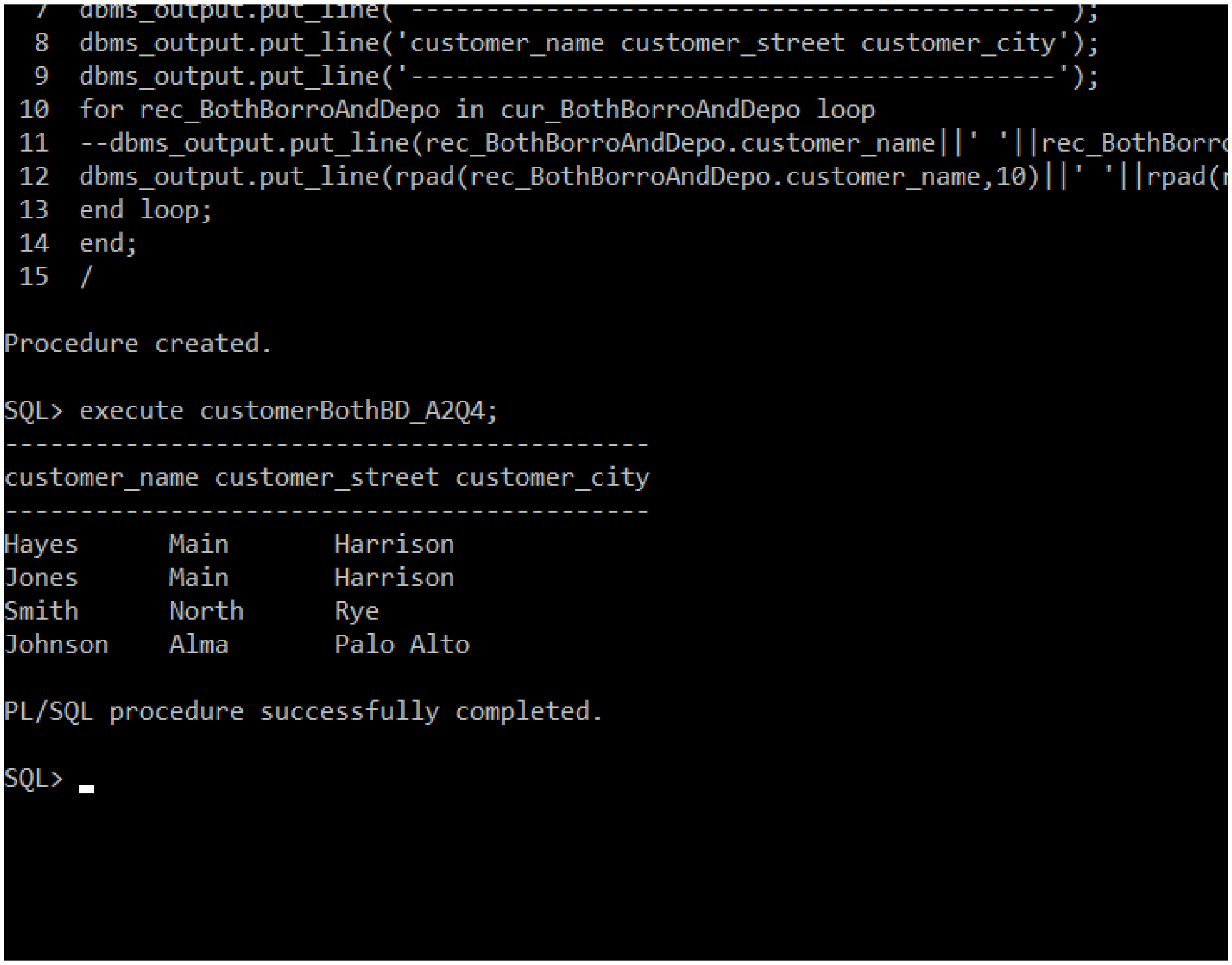
```
create or replace procedure customerBothBD_A2Q4 as

        i int:=1;

        cursor cur_BothBorroAndDepo is select distinct c.customer_name,c.customer_street,c.customer_city from customer c,borrower b,depositor d where c.customer_name=b.customer_name and c.customer_name=d.customer_name;

        rec_BothBorroAndDepo cur_BothBorroAndDepo%rowtype;

begin

        dbms_output.put_line('-------------------------------------------');

        dbms_output.put_line('customer_name customer_street customer_city');

        dbms_output.put_line('-------------------------------------------');

        for rec_BothBorroAndDepo in cur_BothBorroAndDepo loop
```

```
                    --dbms_output.put_line(rec_BothBorroAndDepo.customer_name||' '||rec_BothBorroAndDepo.customer_street||'
'||rec_BothBorroAndDepo.customer_city);

                    dbms_output.put_line(rpad(rec_BothBorroAndDepo.customer_name,10)||'
'||rpad(rec_BothBorroAndDepo.customer_street,10)||' '||rpad(rec_BothBorroAndDepo.customer_city,10));

          end loop;

end;

/
```

Execution

rpad(rec_BothBorroAndDepo.customer_name,10)||' '||rpad(rec_BothBorroAndDepo.customer_street,10)||'
'||rpad(rec_BothBorroAndDepo.customer_city,10);

Image



```
dbms_output.put_line('------------------------------------------');
dbms_output.put_line('customer_name customer_street customer_city');
dbms_output.put_line('------------------------------------------');
for rec_BothBorroAndDepo in cur_BothBorroAndDepo loop
--dbms_output.put_line(rec_BothBorroAndDepo.customer_name||' '||rec_BothBorro
dbms_output.put_line(rpad(rec_BothBorroAndDepo.customer_name,10)||' '||rpad(r
end loop;
end;
/

Procedure created.

SQL> execute customerBothBD_A2Q4;
------------------------------------------
customer_name customer_street customer_city
------------------------------------------
Hayes      Main       Harrison
Jones      Main       Harrison
Smith      North      Rye
Johnson    Alma       Palo Alto

PL/SQL procedure successfully completed.

SQL>
```

5. Write a function with parameter branch name. Return total no. of customers of that branch.

Procedure

```
create or replace function totCustA2Q5(bn branch.branch_name%type) return int as
                totalCustomer int:=0;
                flag int:=0;
                cursor cur_borrower is select distinct customer_name from borrower where borrower.loan_number in  (select
loan_number from loan where branch_name=bn );
                rec_borrower cur_borrower%rowtype;
                cursor cur_depositer is select distinct customer_name from depositor where depositor.account_number in  (select
account_number from account where branch_name=bn );
                rec_depositor cur_depositer%rowtype;
begin
        for rec_borrower in cur_borrower loop
                totalCustomer:=totalCustomer+1;
        end loop;
        for rec_depositor in cur_depositer loop
                flag:=0;
                for rec_borrower in cur_borrower loop
                        if(rec_borrower.customer_name=rec_depositor.customer_name) then
                                flag:=1;
                                exit;
                        end if;
                end loop;
                if(flag!=1) then
                        totalCustomer:=totalCustomer+1;
                end if;
        end loop;
        return totalCustomer;
end;
/
```

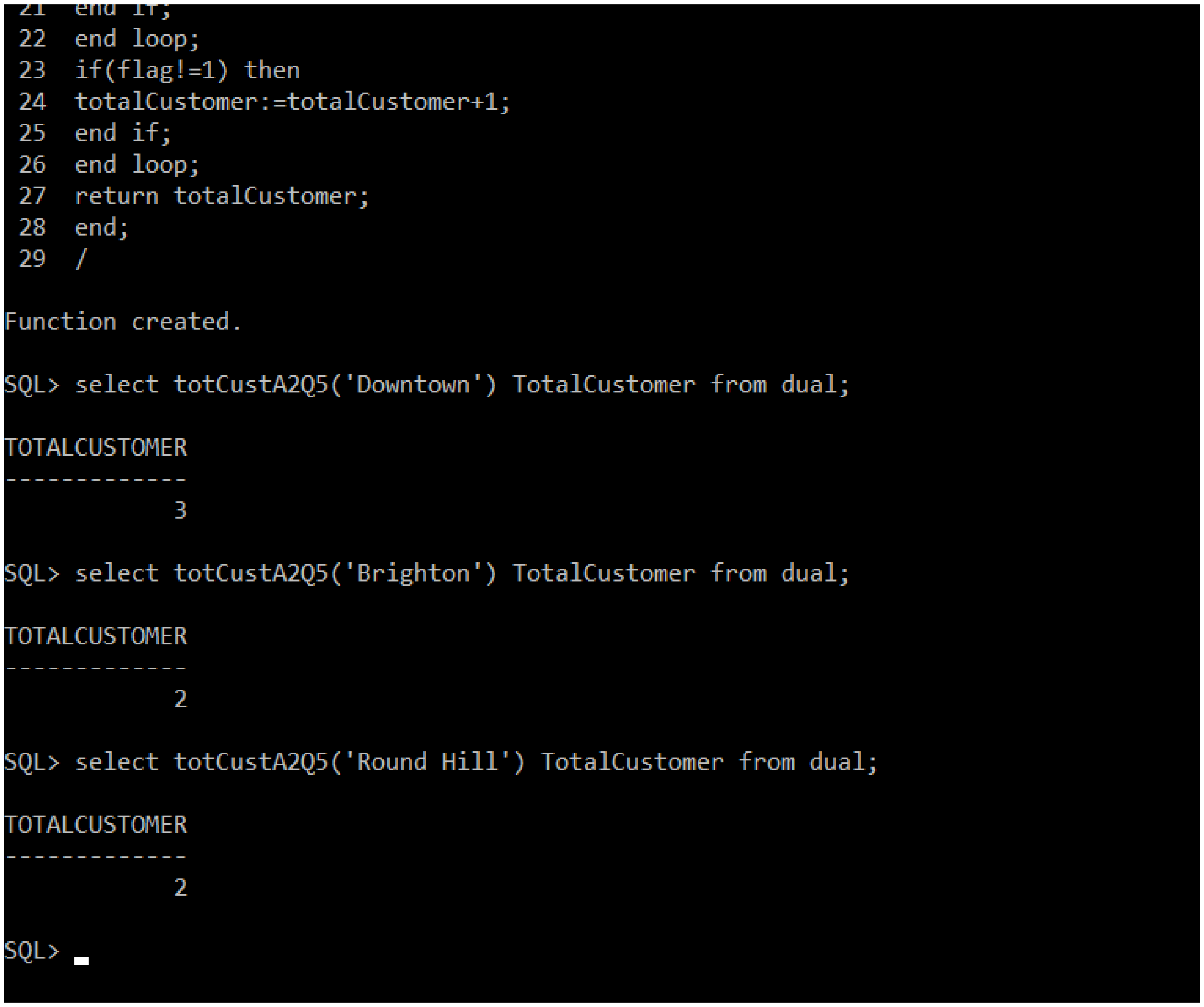
Execution

```
select totCustA2Q5('Downtown') TotalCustomer from dual;
```

select totCustA2Q5('Brighton') TotalCustomer from dual;

select totCustA2Q5('Round Hill') TotalCustomer from dual;

Image



```
end if;
end loop;
if(flag!=1) then
totalCustomer:=totalCustomer+1;
end if;
end loop;
return totalCustomer;
end;
/

Function created.

SQL> select totCustA2Q5('Downtown') TotalCustomer from dual;

TOTALCUSTOMER
-------------
            3

SQL> select totCustA2Q5('Brighton') TotalCustomer from dual;

TOTALCUSTOMER
-------------
            2

SQL> select totCustA2Q5('Round Hill') TotalCustomer from dual;

TOTALCUSTOMER
-------------
            2

SQL>
```

6. Write a function with parameter city name. Return total no. of branches of that city.

Procedure

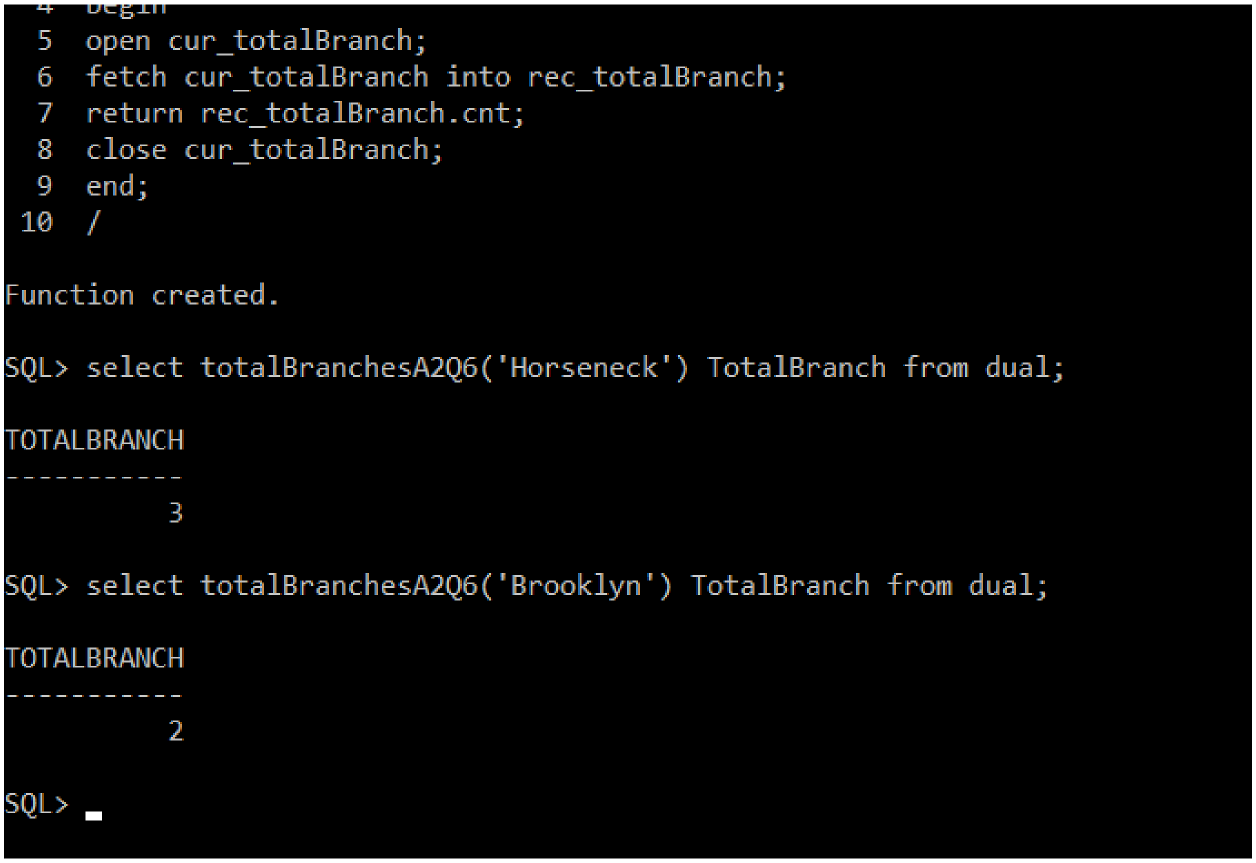
create or replace function totalBranchesA2Q6 (ct branch.branch_city%type) return int as

```
        cursor cur_totalBranch is select count(branch_name) cnt from branch where branch_city=ct group by branch_city;
        rec_totalBranch cur_totalBranch%rowtype;
begin
        open cur_totalBranch;
        fetch cur_totalBranch into rec_totalBranch;
        return rec_totalBranch.cnt;
        close cur_totalBranch;
end;
/
```

Execution

```
select totalBranchesA2Q6('Horseneck') TotalBranch from dual;
select totalBranchesA2Q6('Brooklyn') TotalBranch from dual;
```

Image



```
begin
open cur_totalBranch;
fetch cur_totalBranch into rec_totalBranch;
return rec_totalBranch.cnt;
close cur_totalBranch;
end;
/

Function created.

SQL> select totalBranchesA2Q6('Horseneck') TotalBranch from dual;

TOTALBRANCH
-----------
          3

SQL> select totalBranchesA2Q6('Brooklyn') TotalBranch from dual;

TOTALBRANCH
-----------
          2

SQL>
```

7. Write a function with parameter customer name. Return True if the customer lives in the city where he has account, else return false. Show message too.

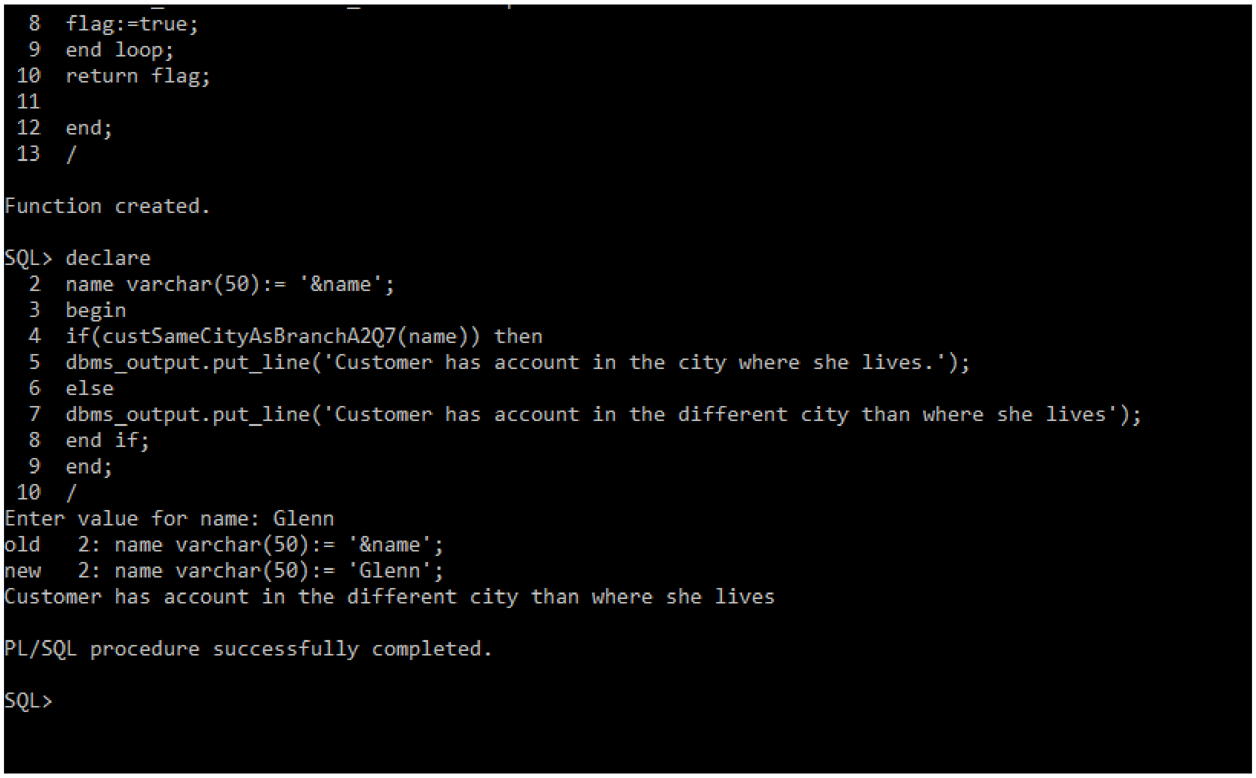
Procedure

```
create or replace function custSameCityAsBranchA2Q7  (cn customer.customer_name%type) return boolean as

        cursor cur_customer is select c.customer_name from customer c,depositor d,account a,branch b where c.customer_name=cn and c.customer_name=d.customer_name and d.account_number=a.account_number and a.branch_name=b.branch_name and c.customer_city=b.branch_city;

        rec_customer cur_customer%rowtype;

        flag boolean:=false;

begin

        for rec_customer in cur_customer loop

                flag:=true;

        end loop;

        return flag;

end;

/
```

Execution

```
--calling block

declare

        name varchar(50):= '&name';

begin

        if(custSameCityAsBranchA2Q7(name)) then

                dbms_output.put_line('Customer has account in the city where she lives.');

        else

                dbms_output.put_line('Customer has account in the different city than where she lives');

        end if;

end;

/
```

Image



```
flag:=true;
end loop;
return flag;

end;
/

Function created.

SQL> declare
name varchar(50):= '&name';
begin
if(custSameCityAsBranchA2Q7(name)) then
dbms_output.put_line('Customer has account in the city where she lives.');
else
dbms_output.put_line('Customer has account in the different city than where she lives');
end if;
end;
/
Enter value for name: Glenn
old   2: name varchar(50):= '&name';
new   2: name varchar(50):= 'Glenn';
Customer has account in the different city than where she lives

PL/SQL procedure successfully completed.

SQL>
```

8. Write a trigger to check balance amount when user inserts or updates balance in accounts table. If balance < 200, don't allow to insert/update the record and display appropriate error message.

Procedure

```
create or replace trigger chk_balanceA2Q8 before insert or update on account

for each row

begin

        if(:new.balance<200) then

                raise_application_error(-20002,'-----Please enter balance greater than 200-------');

        end if;
```

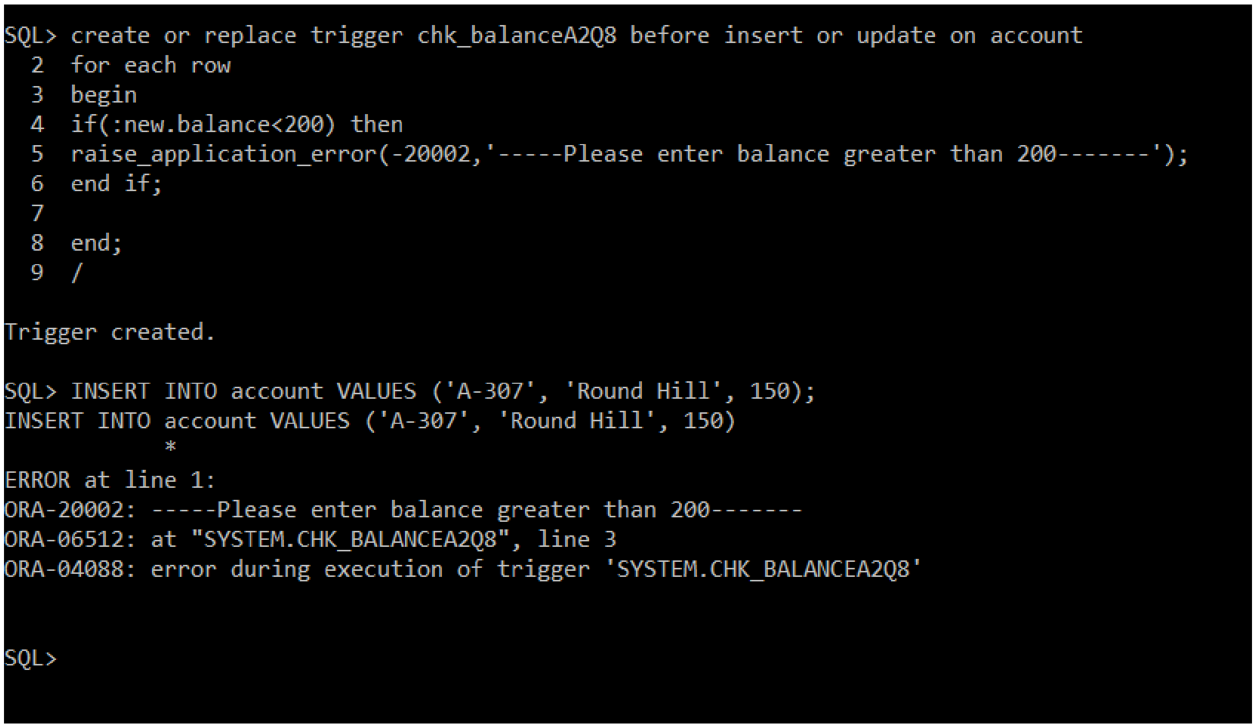
end;

/

Execution

INSERT INTO account VALUES ('A-307', 'Round Hill', 150);

Image



```
SQL> create or replace trigger chk_balanceA2Q8 before insert or update on account
  for each row
  begin
  if(:new.balance<200) then
  raise_application_error(-20002,'-----Please enter balance greater than 200-------');
  end if;

  end;
  /

Trigger created.

SQL> INSERT INTO account VALUES ('A-307', 'Round Hill', 150);
INSERT INTO account VALUES ('A-307', 'Round Hill', 150)
            *
ERROR at line 1:
ORA-20002: -----Please enter balance greater than 200-------
ORA-06512: at "SYSTEM.CHK_BALANCEA2Q8", line 3
ORA-04088: error during execution of trigger 'SYSTEM.CHK_BALANCEA2Q8'


SQL>
```

9. Create a table named city_assets with fields city_name and total_assets. Write a trigger which will insert/update a record in city_assets table when user inserts a new record in the branch table. city_assets table should contain total assets of each city. If the city is inserted for the first time in branch table, insert a new record for that city in the city_assets table. If the city which user is inserting in the branch table already exists in the city_assets table, update the amount in the city_assets table.

Procedure

Make city_assets table

```
create table city_assets(city_name varchar(50) primary key, total_assets int);
```

Insert data

Data can be found from select branch_city,sum(assets) from branch group by branch_city;

```
INSERT INTO city_assets values('Horseneck',10100000);

INSERT INTO city_assets values('Brooklyn',16100000);

INSERT INTO city_assets values('Palo Alto',2100000);

INSERT INTO city_assets values('Bennington',300000);

INSERT INTO city_assets values('Rye',3700000);
```

Trigger

```
create or replace trigger tr_city_assetsA2Q9 before insert on branch

        for each row

        declare

        currentTotAssets int :=0;

        flag int:=0;

        cursor cur_city_assets is select * from city_assets;

        rec_city_assets cur_city_assets%rowtype;

begin

        for rec_city_assets in cur_city_assets loop

                if(:new.branch_city=rec_city_assets.city_name) then
```

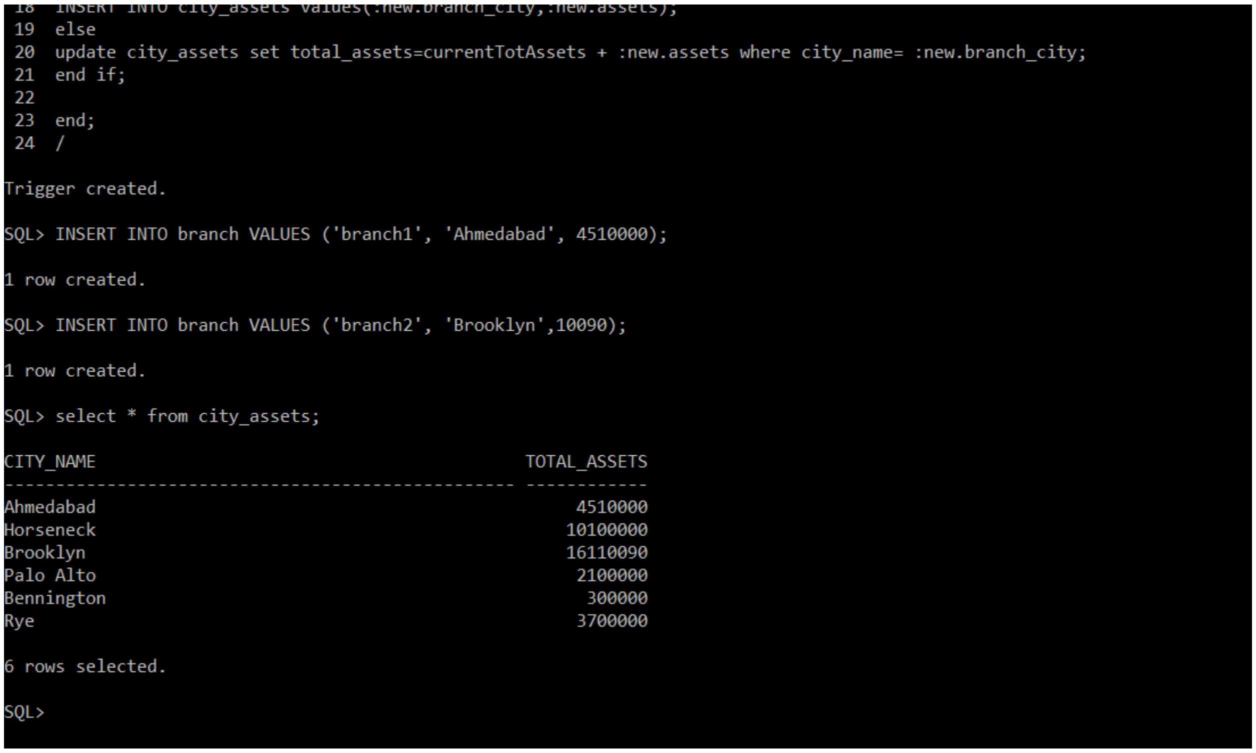
```
                        flag:=1;

                        currentTotAssets:=rec_city_assets.total_assets;

                        exit;

                end if;

        end loop;

        if(flag=0) then

                INSERT INTO city_assets values(:new.branch_city,:new.assets);

        else

                update city_assets set total_assets=currentTotAssets + :new.assets where city_name= :new.branch_city;

        end if;

end;

/
```

Execution

INSERT INTO branch VALUES ('branch1', 'Ahmedabad', 4510000);

INSERT INTO branch VALUES ('branch2', 'Brooklyn',10090);

Image



```
INSERT INTO city_assets values(:new.branch_city,:new.assets);
else
update city_assets set total_assets=currentTotAssets + :new.assets where city_name= :new.branch_city;
end if;

end;
/

Trigger created.

SQL> INSERT INTO branch VALUES ('branch1', 'Ahmedabad', 4510000);

1 row created.

SQL> INSERT INTO branch VALUES ('branch2', 'Brooklyn',10090);

1 row created.

SQL> select * from city_assets;

CITY_NAME                                          TOTAL_ASSETS
-------------------------------------------------- ------------
Ahmedabad                                               4510000
Horseneck                                              10100000
Brooklyn                                               16110090
Palo Alto                                               2100000
Bennington                                               300000
Rye                                                     3700000

6 rows selected.

SQL>
```

10. Write a trigger which will insert details of user, current date and time in a table named “trapped” after user made any changes(insert/delete/update) in the borrower table on -weekends and on weekdays between 10 pm to 6 am. The table trapped contains the fields user_name and date_time.

Procedure

Make table trapped

```
create table trapped(user_name varchar(10),dt timestamp);
```

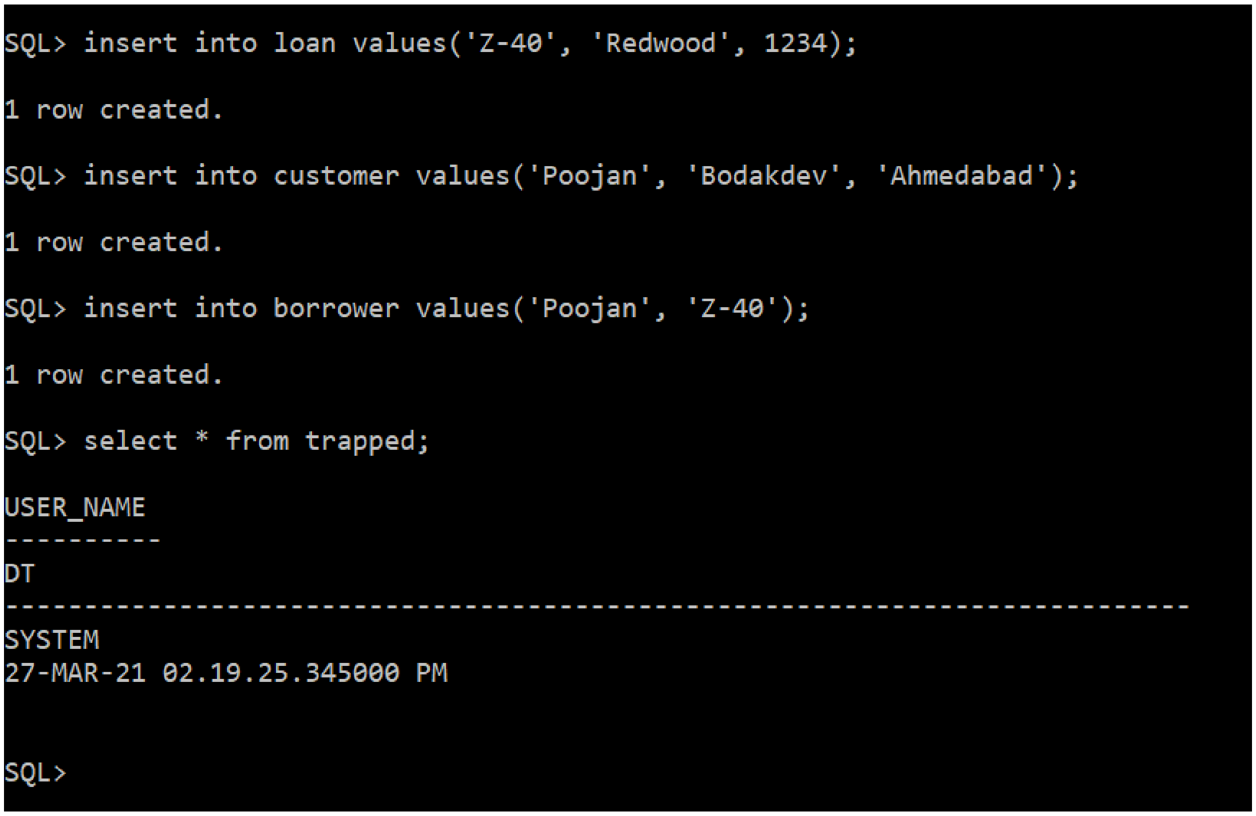
Trigger

```
create or replace trigger tr_trappedA2Q10 after insert or delete or update on borrower
for each row
begin
   if (to_char(sysdate,'dy')='sat' or to_char(sysdate,'dy')='sun' or to_number(to_char(sysdate,'HH24'))<6 or
to_number(to_char(sysdate,'HH24'))>22) then
      insert into trapped values(user, SYSTIMESTAMP);
   end if;
end;
/
```

Execution

```
insert into loan values('Z-40', 'Redwood', 1234);
insert into customer values('Poojan', 'Bodakdev', 'Ahmedabad');
insert into borrower values('Poojan', 'Z-40');
select * from trapped;
```

Image



```
SQL> insert into loan values('Z-40', 'Redwood', 1234);

1 row created.

SQL> insert into customer values('Poojan', 'Bodakdev', 'Ahmedabad');

1 row created.

SQL> insert into borrower values('Poojan', 'Z-40');

1 row created.

SQL> select * from trapped;

USER_NAME
----------
DT
---------------------------------------------------------------------------
SYSTEM
27-MAR-21 02.19.25.345000 PM


SQL>
```

11. Write a trigger when any record is updated in the account table. When value of any field is updated, keep track of before and after values in the table “redolog_values” for each field of the account table. The redolog_values table contains the fields c_date, field_name, before_valueand after_value.

<u>Procedure</u>

<u>Make table redolog_values</u>

create table redolog_values(c_date date, field_name varchar(20), before_value varchar(50),after_value varchar(50));

<u>Trigger</u>

```
create or replace trigger tr_accntTrckA2Q11 after update on account

for each row

begin

        if(:old.account_number != :new.account_number) then

                INSERT INTO redolog_values VALUES(SYSDATE,'account_number',:old.account_number,:new.account_number);
        end if;

        if(:old.branch_name != :new.branch_name) then

                INSERT INTO redolog_values VALUES(SYSDATE,'branch_name',:old.branch_name,:new.branch_name);

        end if;

        if(:old.balance != :new.balance) then

                INSERT INTO redolog_values VALUES(SYSDATE,'balance',:old.balance,:new.balance);

        end if;

end;

/
```

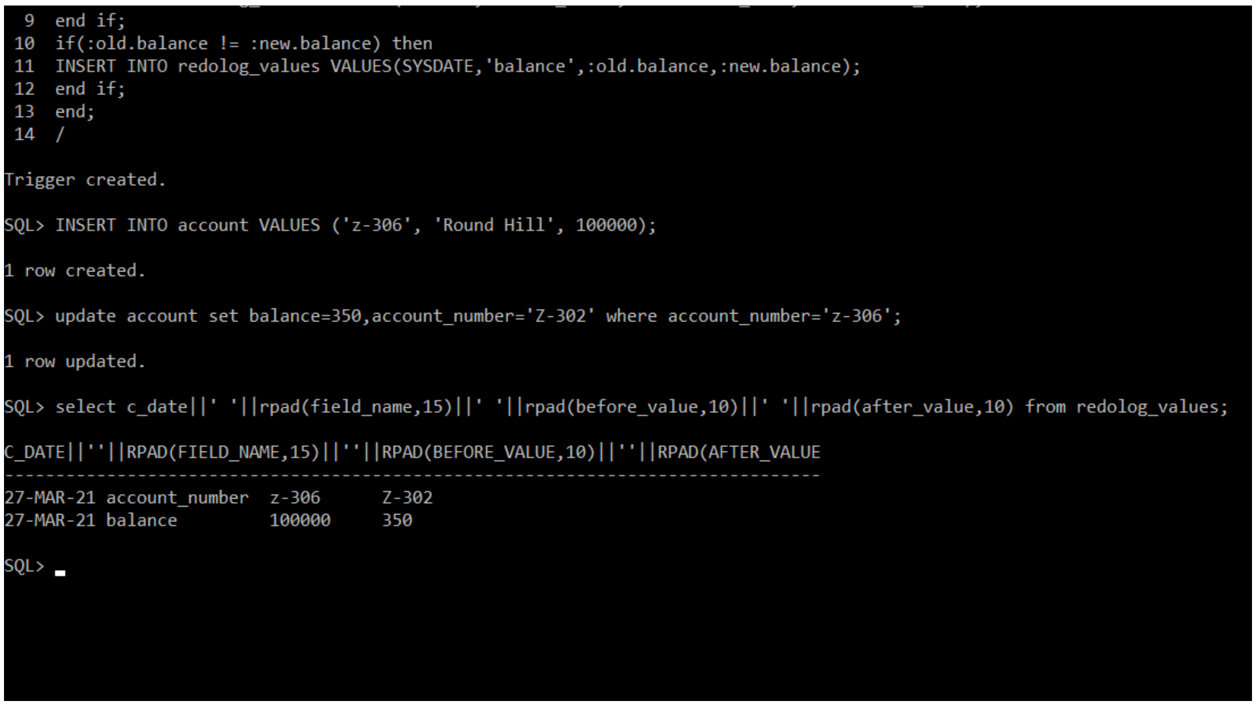
Execution

```
INSERT INTO account VALUES ('z-306', 'Round Hill', 100000);

update account set balance=350,account_number='Z-302' where account_number='z-306';
```

(Here 2 values are updated balance and account_number)

```
select c_date||' '||rpad(field_name,15)||' '||rpad(before_value,10)||' '||rpad(after_value,10) from redolog_values;
```

Image



```
end if;
if(:old.balance != :new.balance) then
INSERT INTO redolog_values VALUES(SYSDATE,'balance',:old.balance,:new.balance);
end if;
end;
/

Trigger created.

SQL> INSERT INTO account VALUES ('z-306', 'Round Hill', 100000);

1 row created.

SQL> update account set balance=350,account_number='Z-302' where account_number='z-306';

1 row updated.

SQL> select c_date||' '||rpad(field_name,15)||' '||rpad(before_value,10)||' '||rpad(after_value,10) from redolog_values;

C_DATE||''||RPAD(FIELD_NAME,15)||''||RPAD(BEFORE_VALUE,10)||''||RPAD(AFTER_VALUE
--------------------------------------------------------------------------------
27-MAR-21 account_number  z-306      Z-302
27-MAR-21 balance         100000     350

SQL>
```

12. Write a trigger which will delete all child records from the borrower and depositor tables when customer record is deleted from the customer table.

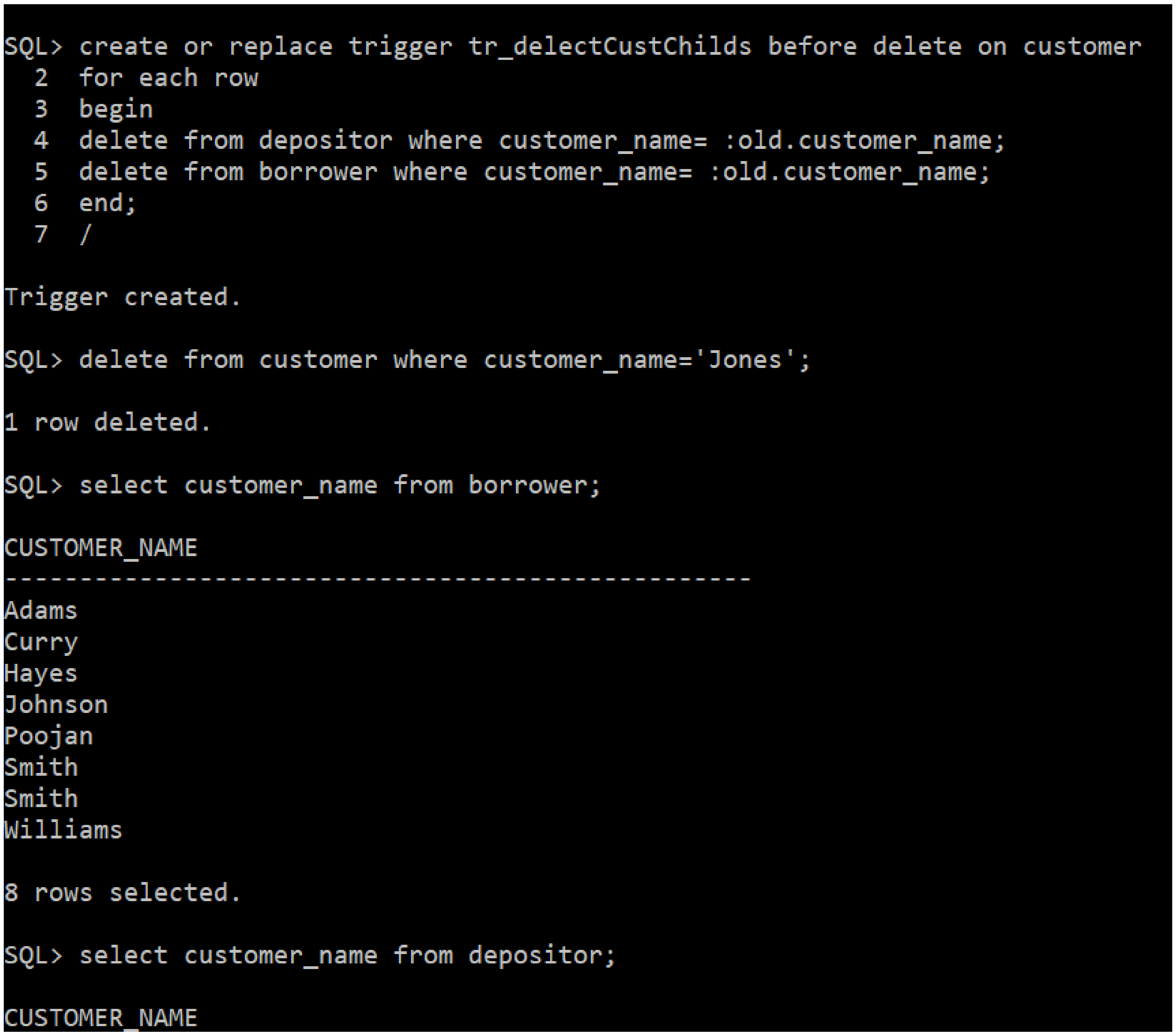
Procedure

```
create or replace trigger tr_delectCustChilds before delete on customer
for each row
begin
	delete from depositor where customer_name= :old.customer_name;
	delete from borrower where customer_name= :old.customer_name;
end;
/
```

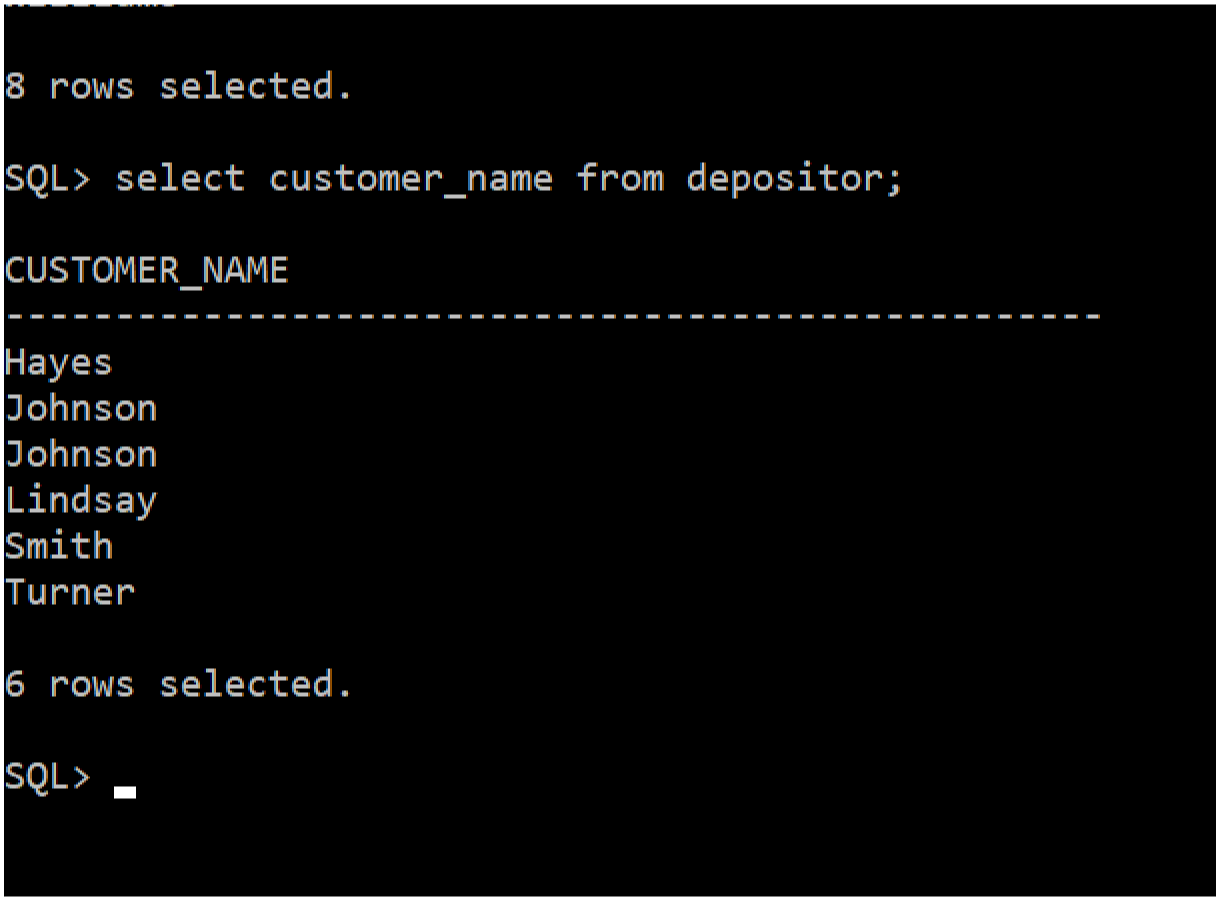
Execution

```
delete from customer where customer_name='Jones';
select customer_name from borrower;
select customer_name from depositor;
```

Image



```
SQL> create or replace trigger tr_delectCustChilds before delete on customer
for each row
begin
delete from depositor where customer_name= :old.customer_name;
delete from borrower where customer_name= :old.customer_name;
end;
/

Trigger created.

SQL> delete from customer where customer_name='Jones';

1 row deleted.

SQL> select customer_name from borrower;

CUSTOMER_NAME
--------------------------------------------------
Adams
Curry
Hayes
Johnson
Poojan
Smith
Smith
Williams

8 rows selected.

SQL> select customer_name from depositor;

CUSTOMER_NAME
```



```
8 rows selected.

SQL> select customer_name from depositor;

CUSTOMER_NAME
--------------------------------------------------
Hayes
Johnson
Johnson
Lindsay
Smith
Turner

6 rows selected.

SQL> 
```