**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;

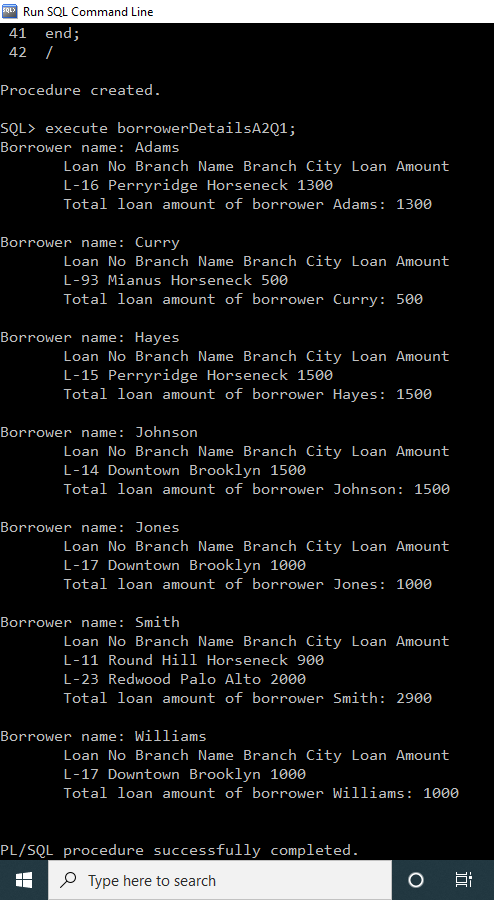
end;

/

**Execution**

Execute borrowerDetailsA2Q1;

**IMAGE**



# **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));

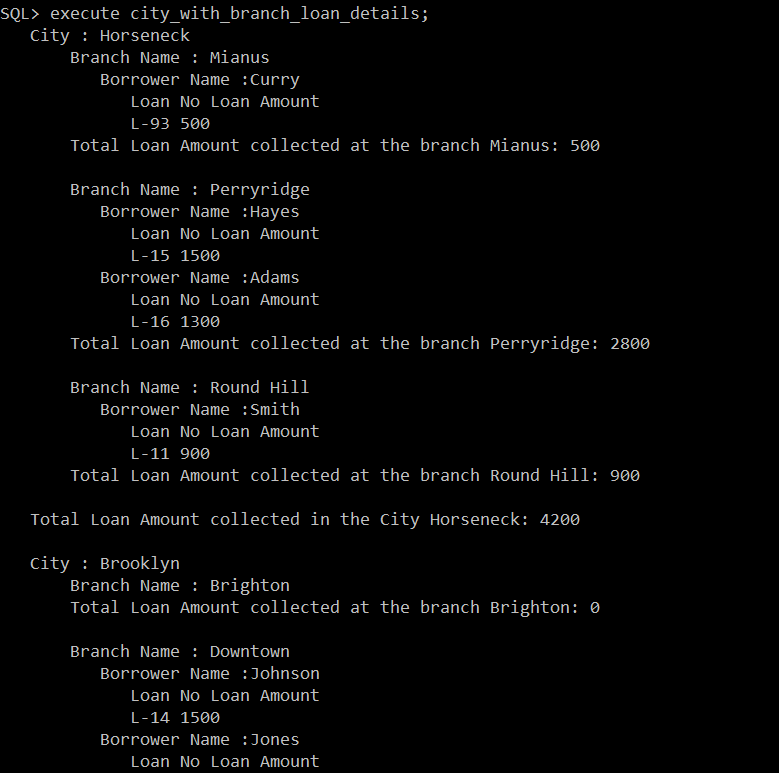
end;

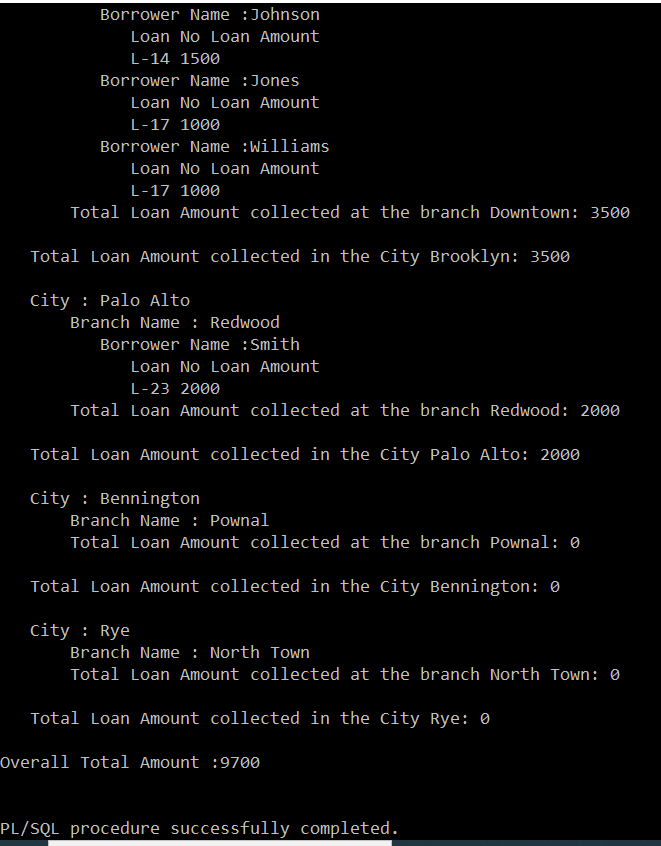
/

**Execution**

execute city\_with\_branch\_loan\_details;

**Image**





# **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));

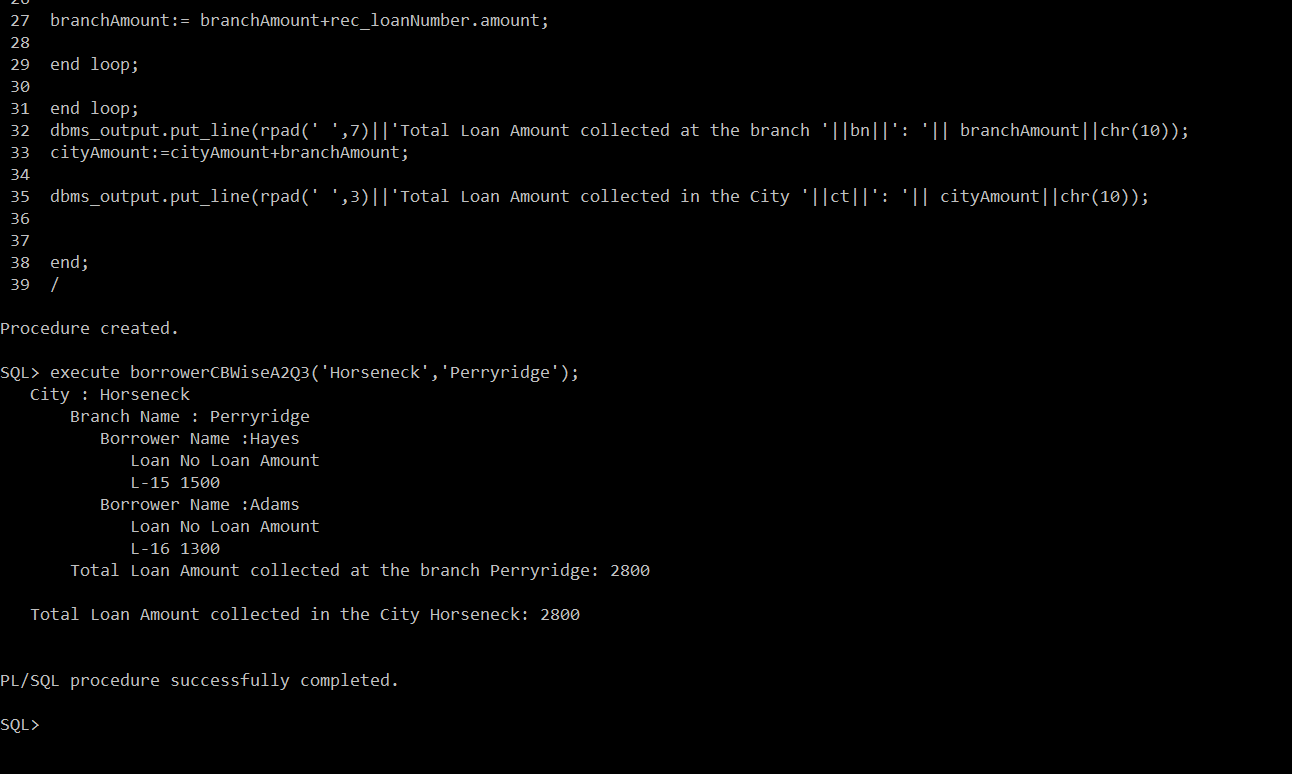
end;

/

**Execution**

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

**Image**



# **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;

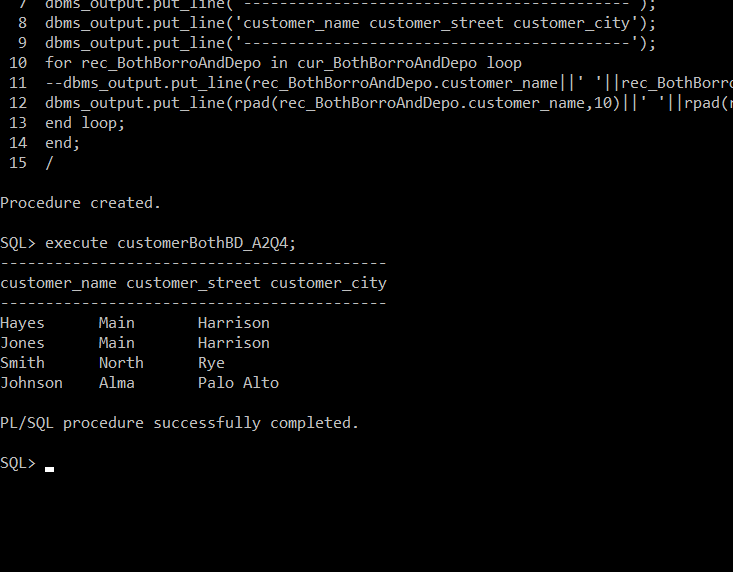
end;

/

**Execution**

rpad(rec\_BothBorroAndDepo.customer\_name,10)||' '||rpad(rec\_BothBorroAndDepo.customer\_street,10)||' '||rpad(rec\_BothBorroAndDepo.customer\_city,10);

**Image**



# **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;

/

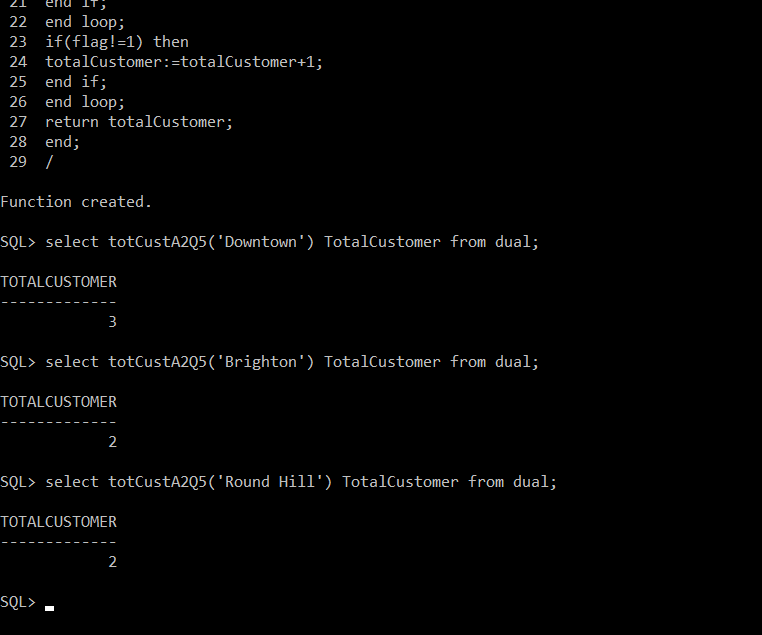
**Execution**

select totCustA2Q5('Downtown') TotalCustomer from dual;

select totCustA2Q5('Brighton') TotalCustomer from dual;

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

**Image**



# **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;

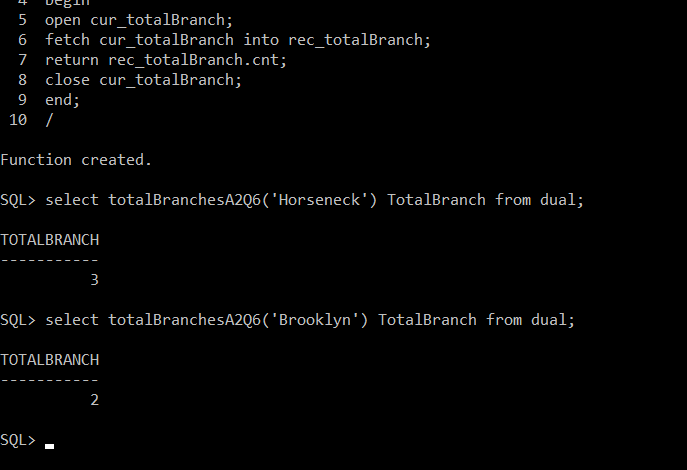
/

**Execution**

select totalBranchesA2Q6('Horseneck') TotalBranch from dual;

select totalBranchesA2Q6('Brooklyn') TotalBranch from dual;

**Image**



# 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

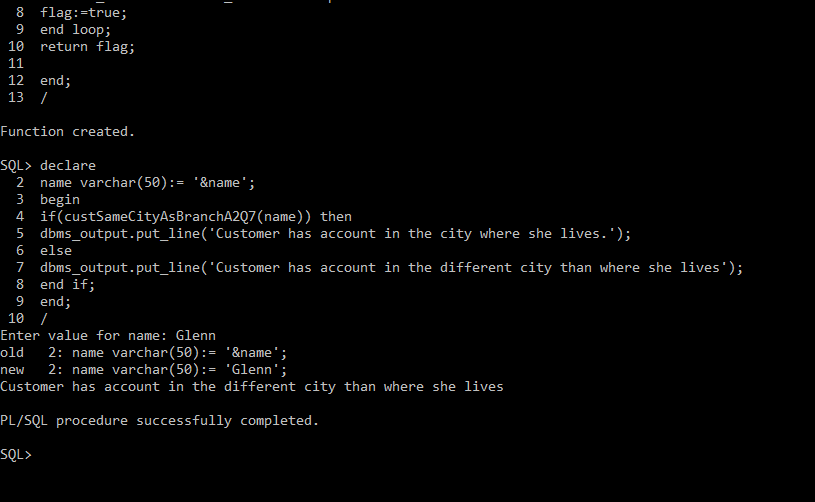
dbms\_output.put\_line('Customer has account in the different city than where she lives');

end if;

end;

/

**Image**



# 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;

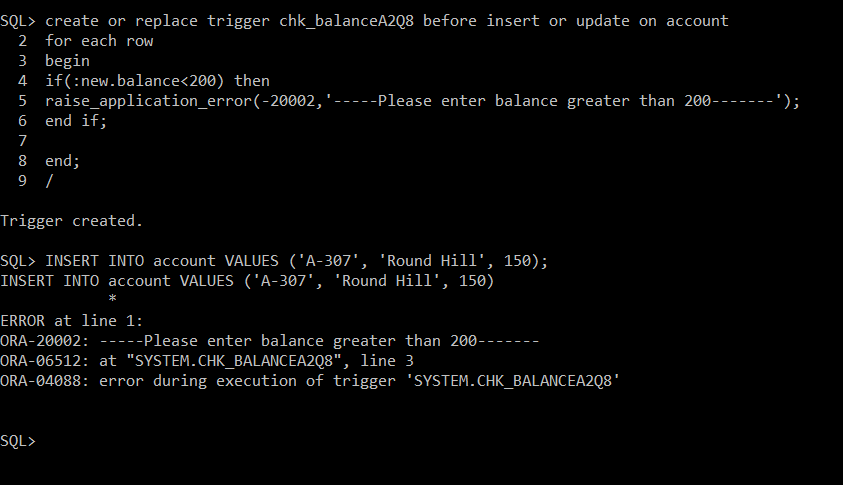
end;

/

**Execution**

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

**Image**



# 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;

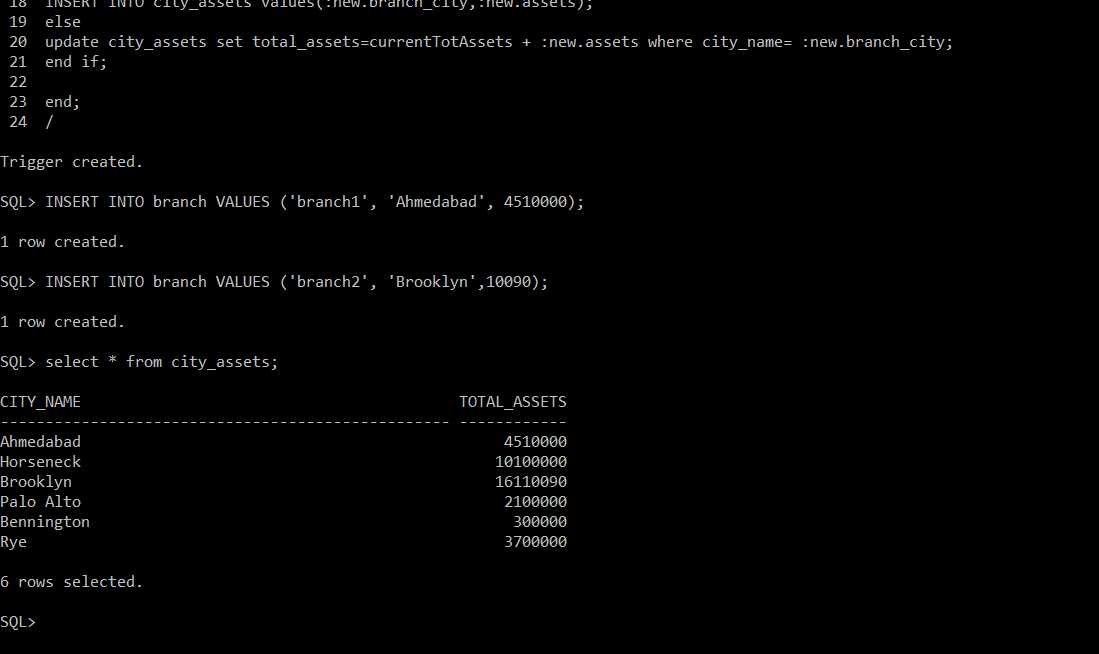
/

**Execution**

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

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

**Image**



# 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**

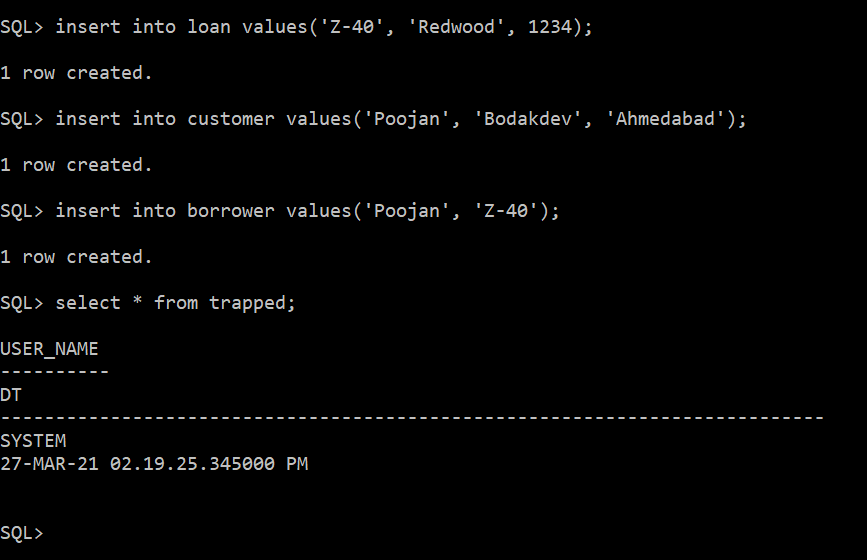
insert into loan values('Z-40', 'Redwood', 1234);

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

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

select \* from trapped;

**Image**



# 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.

**Procedure**

**Make table redolog\_values**

create table redolog\_values(c\_date date, field\_name varchar(20), before\_value varchar(50),after\_value varchar(50));

**Trigger**

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**

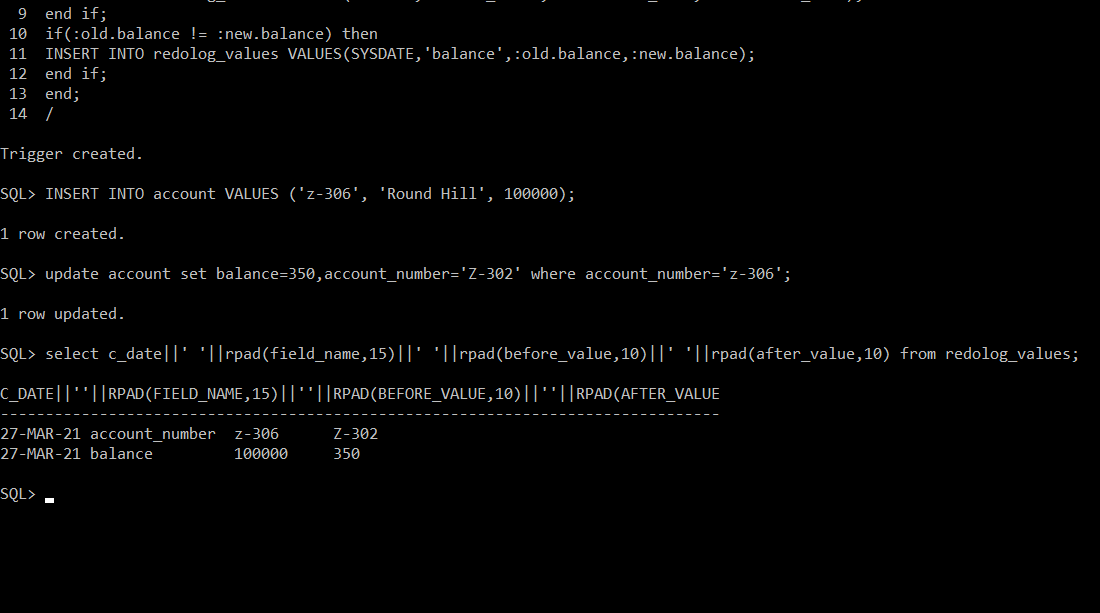
INSERT INTO account VALUES ('z-306', 'Round Hill', 100000);

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**



# 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**

