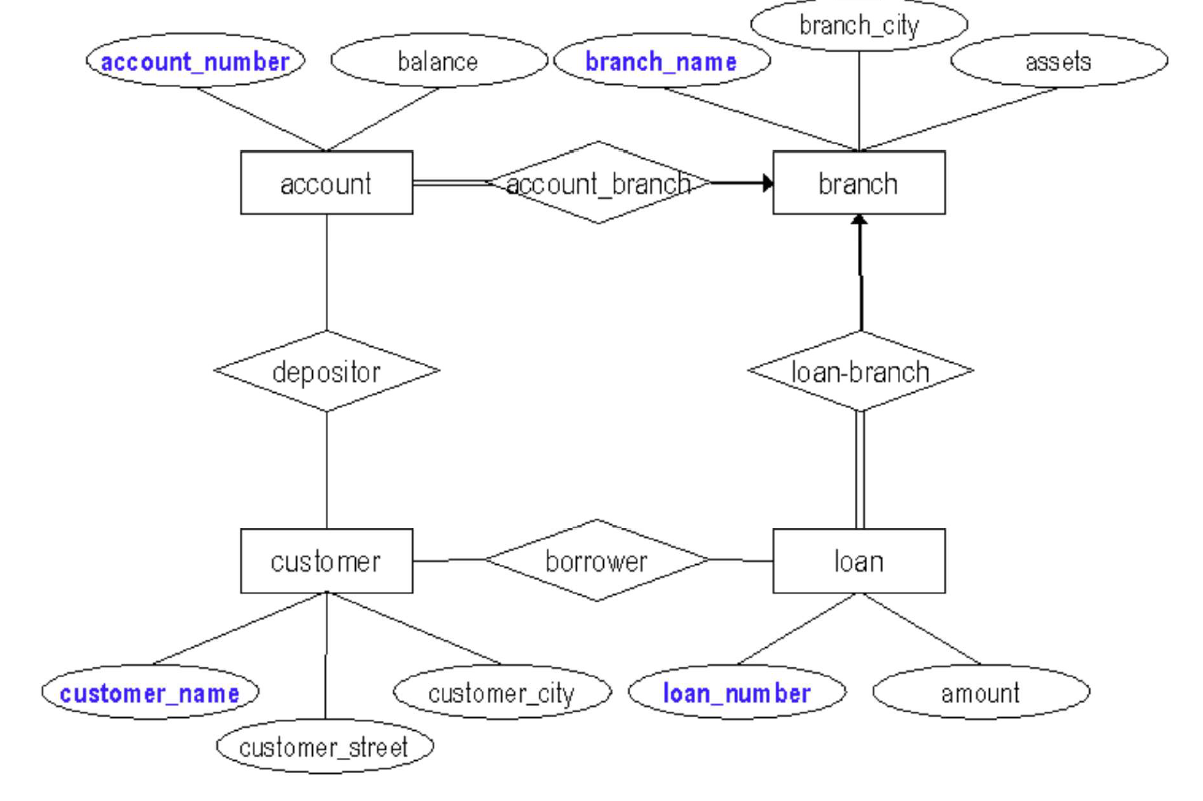
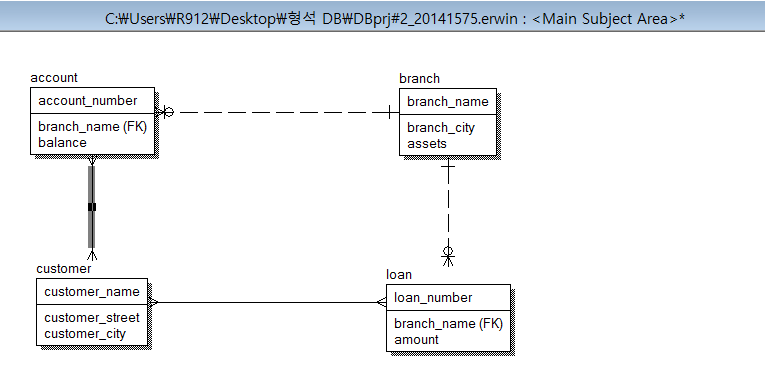
2017 Database System Project #2

20141575 임형석

**아래 ER 다이어그램에 맞는 데이터 베이스를 설계하고 입력 데이터를 바탕으로 속성별 타입과 제약사항을 설정한 후 , ER-win을 이용한 모드 별 설계 내용과 ER-win에서 생성된 데이터 베이스 스키마와 설계에 대한 세부 요구사항 지정 및 설명 등에 대한 내용을 작성해 재출해야 한다. 또한 실제 데이터베이스 서버와 연동한 후 스키마를 생성하고, 주어진 질의를 수행할 수 있는 SQL문을 작성해, 그에 따른 결과를 SQL문과 함께 보고서에 추가하여 제출한다.**



**Logical Mode**



-Entity & Attribute

1. Account  
   account\_number (Primary key) : 계좌 번호   
   balance : 잔액
2. Branch  
   branch\_name (Primary key) : 지점 이름  
   branch\_city : 지점이 위치한 도시  
   assets : 지점 자산
3. Customer  
   customer\_name(Primary key) : 고객 이름  
   customer\_city : 고객이 살고 있는 도시  
   customer\_street : 고객이 살고 있는 거리
4. loan  
   loan\_number (Primary key) : 대출 번호  
   amount : 대출 총액

-Entity 간의 Relation  
 1. account\_branch(Account & Branch)

ER 다이어 그램의 2중선은 total participation을 나타낸다. 어떤 account(계좌)도  
 branch에 속하지 않는다면 존재할 수 없고 또한 여러 계좌가 하나의 지점에 속하

기 때문에 account와 branch는 total participation인 동시에 many to one relation

관계를 가진다.

2. loan-branch (Loan & branch)

loan과 branch도 total participation이자 many to one relation 관계를 가진다.   
 대출을 하려면 어떤 지점에서 해야하는데 branch가 없다면 대출을 할 수

없다. 그리고 한 지점에서 여러 대출을 관리해야 하므로 many to one relation이다.

3. depositor (account & customer)

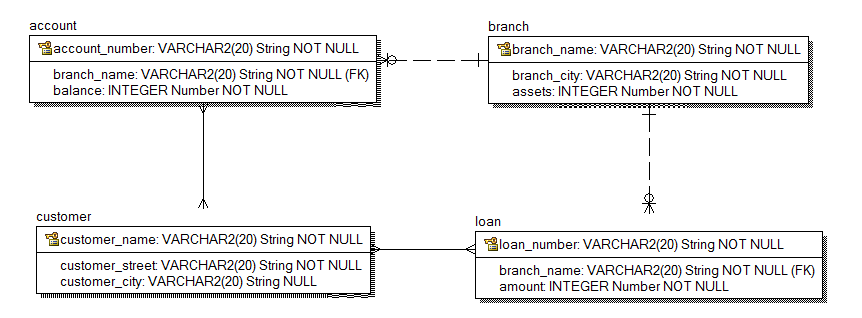
한명의 customer가 여러 account를 생성해 가질 수 있으므로 두 entity가 one-to-many   
 realtionship을 가진다.

4. borrower (customer & loan)

한명의 cunstomer가 여러 번 대출하고 그 때 마다 다른 대출 번호를 받으므로

두 entity가 one-to-many relationship을 가진다.

**Physical Mode**



**-Account**

**account\_number** : 계좌 번호이므로 primary key이고 계좌 번호가 숫자가 아닌  
 ‘A-101’ , ‘A-102’와 같이 char형이 섞여 있기 때문에 20글자까지 가능한 string형

으로 설정하였다.

**Balance** : 계좌 잔액이므로 데이터형을 integer 형으로 해주었고, 잔액은 없어도

0이므로 NOT NULL로 설정해주었다.

**-Branch  
 branch\_name :** 지점 이름이므로 primary key이고 지점 이름이 ‘Downtown’ 과 같은

지역 이름이기 때문에 20글자까지 가능한 string형으로 설정해주었다.

**branch\_city :** 지역 도시이고 ‘Brooklyn’ , ‘Rye’등 도시 이름이기 때문에

20글자까지 가능한 string형으로 설정해주었고 지점은 도시에 속해있으니

NOT NULL로 설정해주었다.

**assets :** 각 지점별 자산을 나타내므로 데이터형을 Interger형으로 해주었고,

자산이 없어도 0이므로 NOT NULL로 설정해주었다.

**-Account**

**customer\_name :** 고객의 이름으로 account의 primary key이다.

‘Adams’, ‘Curry’와 같은 이름이 사용되므로 데이터형은 20글자까지 가능한

string형으로 설정해주었다.

**customer\_street :** 고객이 살고 있는 도시의 거리 이름으로 ‘Spring’, ‘Senator’

와 같이 입력되므로 데이터형은 20글자 까지 가능한 string형으로 설정하였고,

각 도시의 거리에는 이름이 있을 것이기 때문에 NOT NULL로 설정하였다.

**customer\_city :** 고객이 살고 있는 도시 이름으로 ‘Pittsfield’, ‘Rye’와 같이

도시이름이 입력되므로 데이터형은 20글자까지 가능한 string형으로 설정했고

주소에는 반드시 도시가 포함되기 때문에 NOT NULL로 설정해주었다.

**-Loan**

**loan\_number :** 대출 번호로 Loan entity의 primary key이고 숫자로 이루어진 것이

아니라 ‘L-11’, ‘L-14’로 숫자와 알파벳이 섞여있으므로 20글자까지 가능한 string형

으로 지정해주었다.

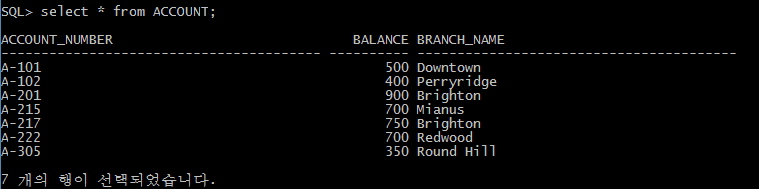
**amount :** 각 대출 번호별 대출 총액이므로 숫자로만 이루어져있으니 데이터형을

integer형으로 설정해주었고, 대출 하지 않으면 대출번호도 생성되지 않으니 NOT

NULL로 설정해주었다.

**데이터 무결성**

모든 entity table에는 primary key를 가지고 있고, 기본 키로 선택된 열은 전부 NOT NULL로 빈 값이 들어가지 않는다. 또한 foreign key의 경우에는 모든 테이블에서 primary key 또는 candidate key와 foreign key의 조합으로 적용되었다. 그리고 특정 attribute 값들이 모두 정의된 domain 범위내에 속해있는 것을 알 수 있고 지정된 데이터 형식을 만족하는 값만 포함한다.

1. **다음 데이터를 입력하시오 (insert)**  
   **<Account relation>**  
   insert into ACCOUNT (ACCOUNT\_NUMBER,BRANCH\_NAME,BALANCE)  
   values (‘A-101’,’Downtown’,500);  
   insert into ACCOUNT (ACCOUNT\_NUMBER,BRANCH\_NAME,BALANCE)  
   values (‘A-102’,’Perryridge’,400);  
   insert into ACCOUNT (ACCOUNT\_NUMBER,BRANCH\_NAME,BALANCE)  
   values (‘A-201’,’Brighton’,900);  
   insert into ACCOUNT (ACCOUNT\_NUMBER,BRANCH\_NAME,BALANCE)  
   values (‘A-215’,’Mianus’,700);  
   insert into ACCOUNT (ACCOUNT\_NUMBER,BRANCH\_NAME,BALANCE)  
   values (‘A-217’,’Brighton’,750);  
   insert into ACCOUNT (ACCOUNT\_NUMBER,BRANCH\_NAME,BALANCE)  
   values (‘A-222’,’Redwood’,700);  
   insert into ACCOUNT (ACCOUNT\_NUMBER,BRANCH\_NAME,BALANCE)  
   values (‘A-305’,’Round Hill’,350);  
   

**<DEPOSITOR relation>**create table DEPOSITOR (CUSTOMER\_NAME VARCHAR2(20), ACCOUNT\_NUMBER VARCHAR2(20) , Primary key(CUSTOMER\_NAME,ACCOUNT\_NUMBER));  
insert into DEPOSITOR (CUSTOMER\_NAME,ACCOUNT\_NUMBER)

values ('Hayes','A-102');

insert into DEPOSITOR (CUSTOMER\_NAME,ACCOUNT\_NUMBER)

values ('Johnson','A-101');

insert into DEPOSITOR (CUSTOMER\_NAME,ACCOUNT\_NUMBER)

values ('Johnson','A-201');

insert into DEPOSITOR (CUSTOMER\_NAME,ACCOUNT\_NUMBER)

values ('Jones','A-217');

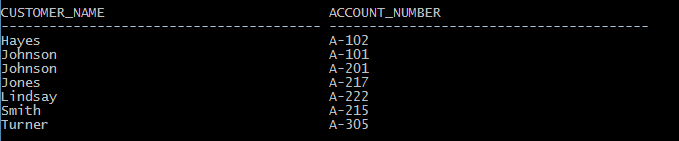
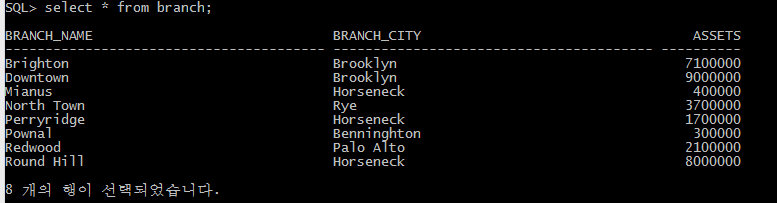
insert into DEPOSITOR (CUSTOMER\_NAME,ACCOUNT\_NUMBER)

values ('Lindsay','A-222');

insert into DEPOSITOR (CUSTOMER\_NAME,ACCOUNT\_NUMBER)

values ('Smith','A-215');

insert into DEPOSITOR (CUSTOMER\_NAME,ACCOUNT\_NUMBER)

values ('Turner','A-305');  
  
**<BRANCH relation>**  
 insert into BRANCH (BRANCH\_NAME,BRANCH\_CITY,ASSETS)  
 values (‘Brighton’,’Brooklyn’,7100000);  
 insert into BRANCH (BRANCH\_NAME,BRANCH\_CITY,ASSETS)  
 values (‘Downtown’,’Brooklyn’,9000000);   
 insert into BRANCH (BRANCH\_NAME,BRANCH\_CITY,ASSETS)  
 values (‘Mianus’,’Horseneck’,400000);   
 insert into BRANCH (BRANCH\_NAME,BRANCH\_CITY,ASSETS)  
 values (‘North town’,’Rye’,3700000);   
 insert into BRANCH (BRANCH\_NAME,BRANCH\_CITY,ASSETS)  
 values (‘Perryridge’,’Horseneck’,1700000);   
 insert into BRANCH (BRANCH\_NAME,BRANCH\_CITY,ASSETS)  
 values (‘Pownal’,’Benninghton’,300000);   
 insert into BRANCH (BRANCH\_NAME,BRANCH\_CITY,ASSETS)  
 values (‘Redwood’,’Palo Alto’,2100000);   
 insert into BRANCH (BRANCH\_NAME,BRANCH\_CITY,ASSETS)  
 values (‘Round Hill’,’Horseneck’,8000000); 

**<CUSTOMER relation>**  
insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

values (‘Adams’,’Spring’,’Pittsfield’);  
insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

values (‘Brooks’,’Senator’,’Brooklyn’);  
insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

values (‘Curry’,’North’,’Rye’);  
insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

values (‘Glenn’,’Sand Hill’,’Woodside’);  
insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

values (‘Green’,’Walnut’,’Stamford’);  
insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

values (‘Hayes’,’Main’,’Harrison’);  
insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

values (‘Johnson’,’Alma’,’Palo Alto’);  
insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

values (‘Jones’,’Main’,’Harrison’);  
insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

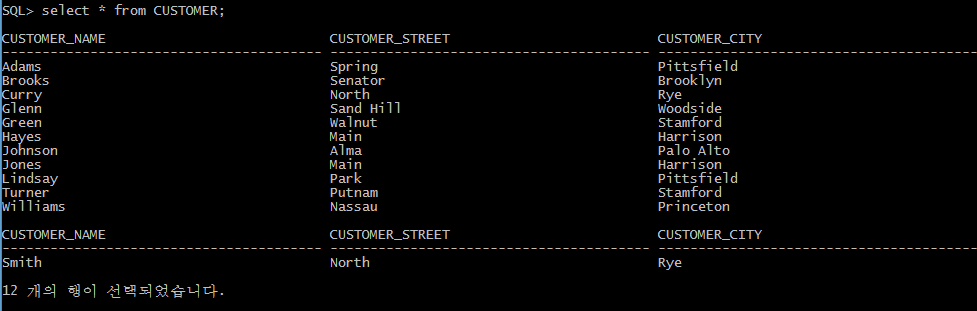
values (‘Lindsay’,’Park’,’Pittsfield’);

insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

values (‘Smith’,’North’,’Rye’);  
insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

values (‘Turner’,’Putnam’,’Stamford’);  
insert into CUSTOMER (CUSTOMER\_NAME,CUSTOMER\_STREET,CUSTOMER\_CITY)

values (‘Williams’,’Nassau’,’Princeton’);



**<LOAN relation>**  
insert into LOAN (LOAN\_NUMBER, BRANCH\_NAME,AMOUNT)

values (‘L-11’,’Round Hill’, 900);  
insert into LOAN (LOAN\_NUMBER, BRANCH\_NAME,AMOUNT)

values (‘L-14’,’Downtown’, 1500);  
insert into LOAN (LOAN\_NUMBER, BRANCH\_NAME,AMOUNT)

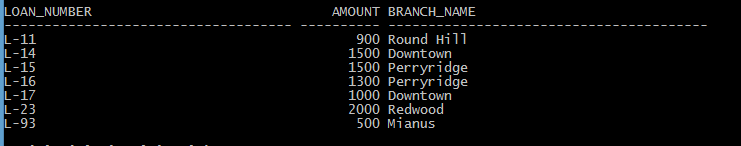
values (‘L-15’,’Perryridge’, 1500);  
insert into LOAN (LOAN\_NUMBER, BRANCH\_NAME,AMOUNT)

values (‘L-16’,’Perryridge’, 1300);  
insert into LOAN (LOAN\_NUMBER, BRANCH\_NAME,AMOUNT)

values (‘L-17’,’Downtown’, 1000);  
insert into LOAN (LOAN\_NUMBER, BRANCH\_NAME,AMOUNT)

values (‘L-23’,’Redwood’, 2000);  
insert into LOAN (LOAN\_NUMBER, BRANCH\_NAME,AMOUNT)

values (‘L-93’,’Mianus’, 500);



**<BORROWER relation>**

create table BORROWER (CUSTOMER\_NAME VARCHAR2(20), LOAN\_NUMBER VARCHAR2(20), primary key(CUSTOMER\_NAME,LOAN\_NUMBER));  
insert into BORROWER (CUSTOMER\_NAME,LOAN\_NUMBER)

values ('Adams','L-16');

insert into BORROWER (CUSTOMER\_NAME,LOAN\_NUMBER)

values ('Curry','L-93');

insert into BORROWER (CUSTOMER\_NAME,LOAN\_NUMBER)

values ('Hayes','L-15');

insert into BORROWER (CUSTOMER\_NAME,LOAN\_NUMBER)

values ('Johnson','L-14');

insert into BORROWER (CUSTOMER\_NAME,LOAN\_NUMBER)

values ('Jones','L-17');

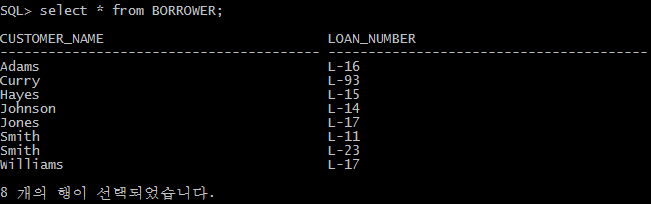
insert into BORROWER (CUSTOMER\_NAME,LOAN\_NUMBER)

values ('Smith','L-11');

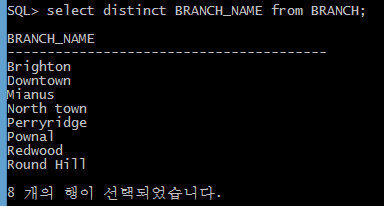
insert into BORROWER (CUSTOMER\_NAME,LOAN\_NUMBER)

values ('Smith','L-23');

insert into BORROWER (CUSTOMER\_NAME,LOAN\_NUMBER)

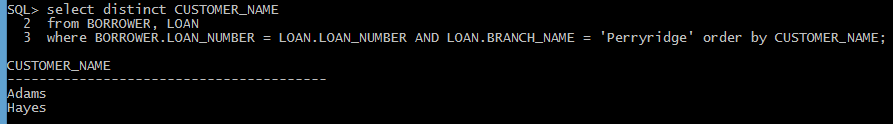
values ('Williams','L-17');  


**2) 중복되지 않은 모든 지점들의 이름을 구하라.**



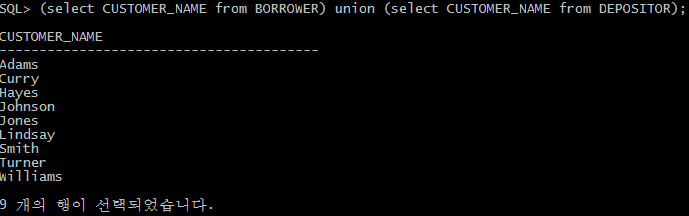
중복되지 않는 지점들을 구하기 위해 distinct를 이용하였다.

**3) Perryridge 지점의 대출을 가진 모든 고객들을 알파벳 순서로 나열하라.**

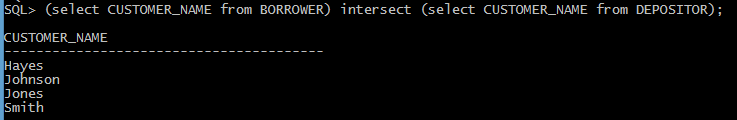


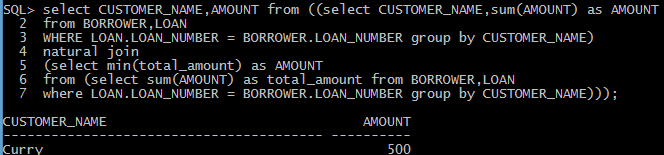
Perryridge지점에 대출을 가진 고객을 구하기 위해 BORROWER와 LOAN 테이블에서 같은 LOAN\_NUMBER를 가지고 Perryridge지점에서 빌렸을 때의 고객의 이름을 구하였고, 고객의 이름을 알파벳 순서대로 order by를 이용해 나열했다.

**4) 은행에서 대출, 계좌 혹은 둘 다를 가진 모든 고객을 나열하라.**

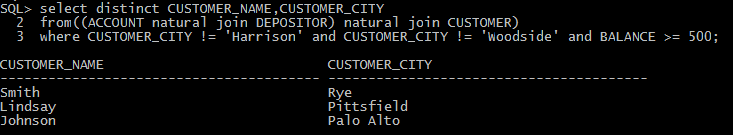


은행에서 계좌,대출 혹은 둘 다 가진 고객이니 두 테이블의 합집합(union)을 한 것의 고객의 이름을 구하면 된다.

**5) 은행에 대출과 계좌 모두를 가진 모든 고객을 나열하라.**은행에서 계좌,대출 둘 다 가진 고객이어야만 하므로 두 테이블의 교집합(intersect) 한 것의 고객의 이름을 구하면 된다.

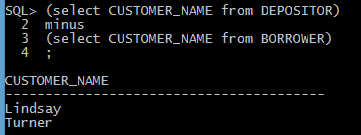
**6) 대출 총액이 가장 작은 고객의 이름과 대출 총액을 구하여라.**대출 총액이 가장 작은 고객과 총액을 구하기 위해 각 고객 별로 대출 총액을 구한 후 대출 총액 중 가장 작은 값을 가지는 대출액을 구하고 두 table을 natural join시킨 후 대출 총액이 가 장 작은 고객의 이름과 대출 총액을 구한다.

**7) Harrison과 Woodside에 살지 않으면서 계좌에 잔고가 500이상 있는 고객의 이름과 고객이 사는 도시를 구하라.**



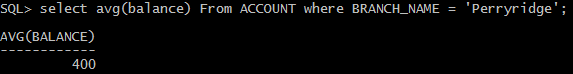
고객의 이름, 도시와 계좌를 구하기 위해 ACCOUNT,DEPOSITOR,CUSTOMER 테이블을 natrual join해주고 조건으로 customer의 city가 ‘Harrison’과 ‘woodside’가 아니고, 잔액이 500이상 남은 고객의 이름과 고객의 도시를 출력해주었다.

**8) 은행에 계좌는 가지고 있지만 대출은 가지고 있지 않은 모든 고객들을 나열하라.**



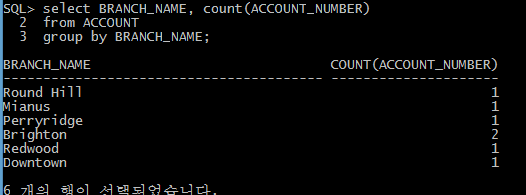
계좌를 가진 고객에서 대출이 있는 고객을 빼주었다.

**9) Perryridge 지점에서 계좌의 평균 잔고를 구하여라.**

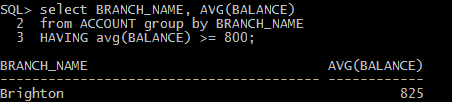


where절로 지점의 이름은 ‘Perryridge’로 고르고 ACCOUNT의 balance를 평균 낸 것을 select해주었다.

**10) 각 지점의 예금자들의 수를 구하라.**

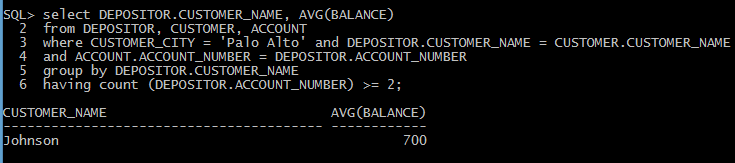


일단 ACCOUNT 테이블에서 각 지점마다의 account\_number를 count해준다.

**11) 평균 잔고가 $800 이상인 지점 이름과 평균 잔고를 나열하라.**

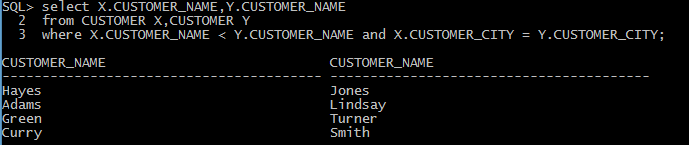
Account 테이블에서 branch name으로 group하여 잔액의 평균이 800이상인 것만 출력해준다.

**12) Palo Alto에 살고 최소한 두 개의 계좌를 가진 각각의 고객들의 이름과 평균 잔고를 구하라.**

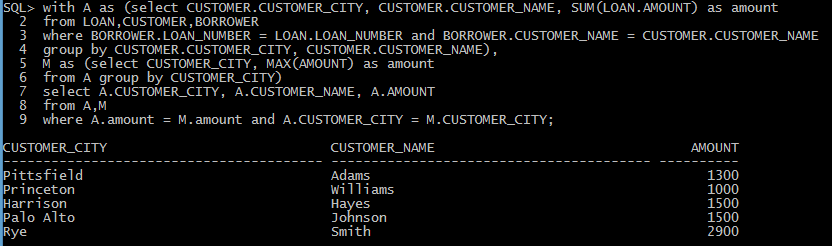


일단 고객의 account 정보와 사는 위치,이름 그리고 잔액을 알아야 하기 때문에 DEPOSITOR, CUSTOMER,ACCOUNT 총 3가지의 table을 사용했다. 일단 Palo Alto에 살고, 그 중 2개 이상의 계좌가 있는 사람들을 골라 출력해주어야 하므로, CUSTOMER\_NAME으로 group해주고, ACCOUNT\_NUMBER가 2개이상인 것으로 조회하였다.

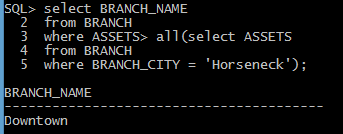
**13) 같은 도시에 사는 고객의 이름의 쌍을 구하여라.**



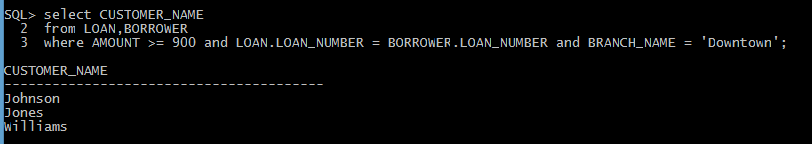
두개의 customer table을 만들고, 각각의 테이블에서 같은 도시에 살고, 이름이 중복하여 출력되지 않도록 하게 조건을 걸어 조회하였다.

**14) 각 도시 별로 가장 높은 대출 총액을 가지고 있는 고객의 이름과 대출 총액을 구하여라. 단, 대출을 가진 고객이 살지 않는 도시는 표시하지 않는다.**

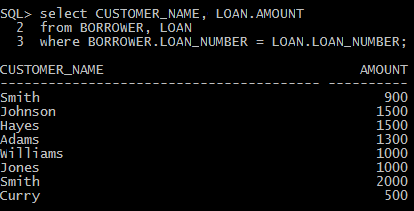
각 도시별로 고객 별 대출 총액과 이름을 가지는 table을 A라고 하고, 각 도시마다 제일 높은 총액을 가지고 있는 table을 M이라고 하자. 두 테이블의 도시와 총액이 같을 때 그 때의 도시와 대출 총액과 그 때의 고객이름을 조회한다.

**15) Horseneck에 있는 각 지점보다 큰 자산 값을 갖는 모든 지점들의 이름을 나열하라.**

horseneck에 있는 각 지점 보다 큰 자산을 가지는 지점을 나열하기 위해 all을 사용하였고 큰 자산을 가지는 지점의 이름을 조회하였다.

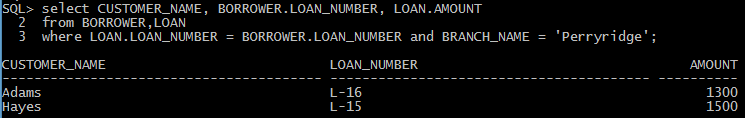
**16) Downtown 지점에서 $900 이상의 대출 총액을 지닌 고객들을 전부 구하라.**

고객의 대출 정보와 대출 총액을 이용하기 위해 LOAN과 BORROWER 테이블을 가져왔고, Downtown 지점이면서 대출 총액이 $900이상인 고객의 이름만 조회하였다.

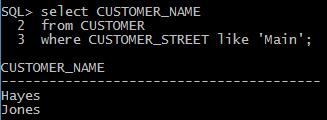
**17) 은행에 대출을 가지고 있는 모든 고객들에 대해 그들의 이름과 대출 총액을 구하라.**

대출을 가지고 있는 모든 고객에 대해 정보를 가져와야하니 BORROWER와 LOAN table을 이용하였고, 두 테이블에서 빌린 사람의 대출 번호를 비교해 이름과 그 사람의 대출 총액을 조회해주었다.

**18) Perryridge 지점의 모든 대출에 대해여 고객의 이름과 대출 번호, 대출액을 구하라.**

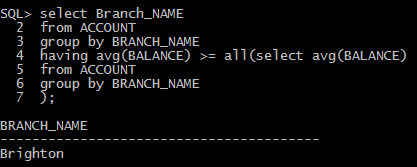


Perryridge의 지점에 대출을 받은 고객에 대해 이름, loan\_number, amount를 구하니 BORROWER와 LOAN 테이블을 이용하여 찾아주었다.

**19) 이름에 'Main'이라는 부분 문자열이 포함된 거리에 살고 있는 모든 고객들의 이름을 구하여라.**

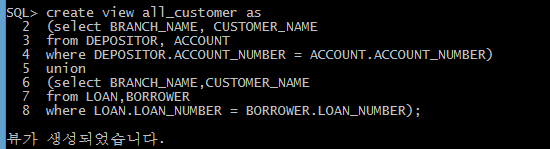
여기선 like를 이용하여 Main이라는 문자열이 포함된 고객의 거리를 찾아 조회해주었다.

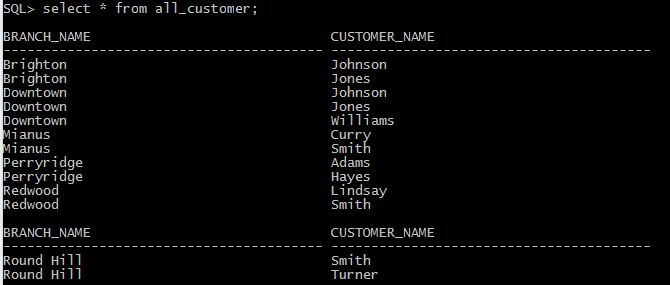
**20) 가장 높은 평균 잔고를 가진 지점을 구하라.**



각 지점의 평균 잔고를 구해주고 그것을 테이블로 만든 후 all을 이용해 모든 지점의 평균 잔고랑 비교하여 그 중 가장 큰 것을 구하여 평균 잔고가 가장 높은 지점을 조회하였다.

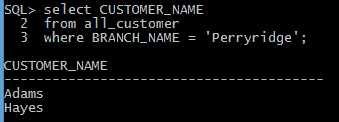
**21) 지점 이름과 그 지점에 계좌나 대출 둘 중 하나를 가진 고객 이름으로 구성된 View를 작성하라. (단 View의 이름은 all\_customer이다.)**





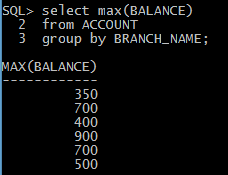
union을 이용해 계좌나 대출이 있는 사람을 조회하였고 create view로 뷰를 생성하였다.

**22) 21에서 생성된 View를 이용하여 Perryridge 지점의 모든 고객 이름을 나열하라.**



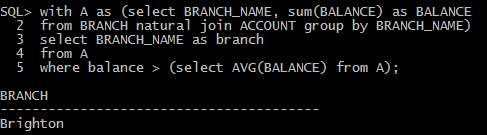
21에서 생성한 view에서 ‘Perryridge’지점의 고객을 찾아 조회하였다

**23) 각 지점에서 총 잔고의 최대값을 나열하라.**



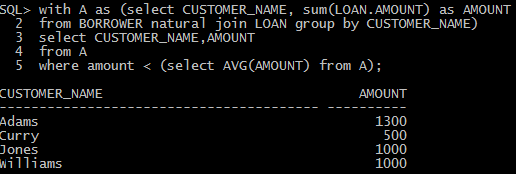
max를 이용하여 ACCOUNT테이블에서 각 지점에서 잔고의 최대 값을 조회하였다.

**24) 모든 지점의 총 계좌 예금의 평균보다 많은 총 계좌 예금을 갖는 모든 지점을 나열하라.**



모든 지점의 총 계좌 예금을 구하기 위해 branch 와 account를 natural join해주었고 그것을 지점 별로 group해준 것을 A라고 해주었고, A의 balance와 그것들의 평균을 비교해 더 높은 지점을 조회해주었다.

**25) 평균 대출 총액보다 적은 대출 총액을 가지고 있는 고객의 이름과 대출 총액을 구하라.**



대출 총액을 가지고 있는 고객 이름과 대출 총액 table을 만들어주기 위해 BORROWER와 LOAN을 natural join시키고 고객이름 별로 group해준 것을 A라고 하고 대출 총액과 고객들의 평균 대출 총액을 비교하여 더 작은 고객의 이름, 대출 총액을 조회해주었다.