<프로젝트 기술 문서>

**2018 Database System**

**Project #2**

**Constructing SQL**

데이터베이스 시스템

박 석 교수님

서강대학교 공과대학

컴퓨터공학과 3학년

20141544 안시현

**목차**

1. 프로젝트 목표 및 문제정의

2. ER Diagram

3. Physical Mode

3-1. 사용된 도메인

3-2. 사용된 Validation Constraint

3-3. ACCOUNT

3-4. CUSTOMER

3-5. BRANCH

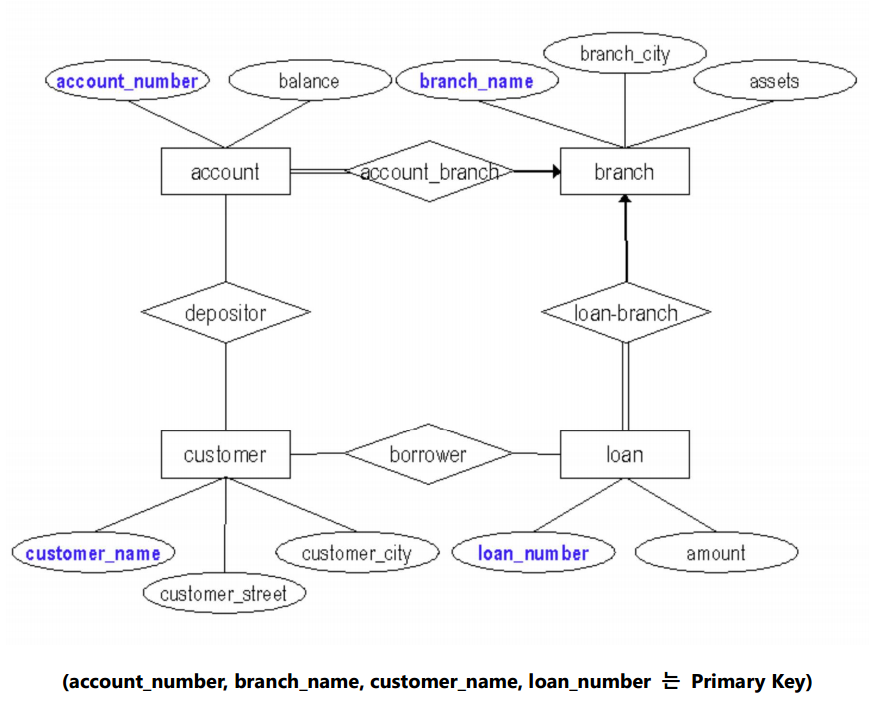
3-6. LOAN

4. SQL 질의문 및 결과

**1. 프로젝트 목표 및 문제 정의**

\* 프로젝트 목표 : 주어진 ER 다이어그램에 맞는 데이터베이스를 설계하고 입력 데이터를 바탕으로 속성별 타입과 제약을 설정한다. 즉 ER-win을 이용하여 모드 별로 설계한다. 이를 바탕으로 실제 데이터 베이스 서버와 연동한 후 스키마를 생성하고 주어진 문제에 대한 질의를 SQL문으로 작성하여 그에 따른 결과를 확인한다.

**2. ER 다이어그램**



**<전체 ER 다이어그램>**

주어진 데이터베이스 설계의 ER 다이어그램은 위와 같다. ER 다이어그램의 Entity로 account, branch , loan , customer로 구성되며 이들은 attribute를 가지며 Relationship으로 연결되어 있다.

account Entity는 계좌를 나타낸다. 이는 account\_number(계좌번호)를 primary key로 가지며 이에

더불어 balance(잔액)를 attribute로 가지고 있다. 그리고 branch Entity와는 account\_branch라는

relationship을 가지고 customer와는 depositor의 relationship을 가진다. customer Entity는 고객을

나타내며 customer\_name(고객이름)을 primary key로 가지며 이에 더불어 customer\_street(고객이

거주하는 거리) , customer\_city(고객이 거주하는 도시)를 attribute로 가지고 있다. 그리고 account

Entity와는 depositor라는 relationship을 loan Entity와는 borrower라는 relationship을 가진다.

branch Entity는 지점을 나타내며 branch\_name(지점이름)을 primary key로 가지며 이에 더불어

assets(지점이 보유한 자산) , branch\_city(지점이 위치하는 도시)를 attribute로 가지고 있다. 그리고

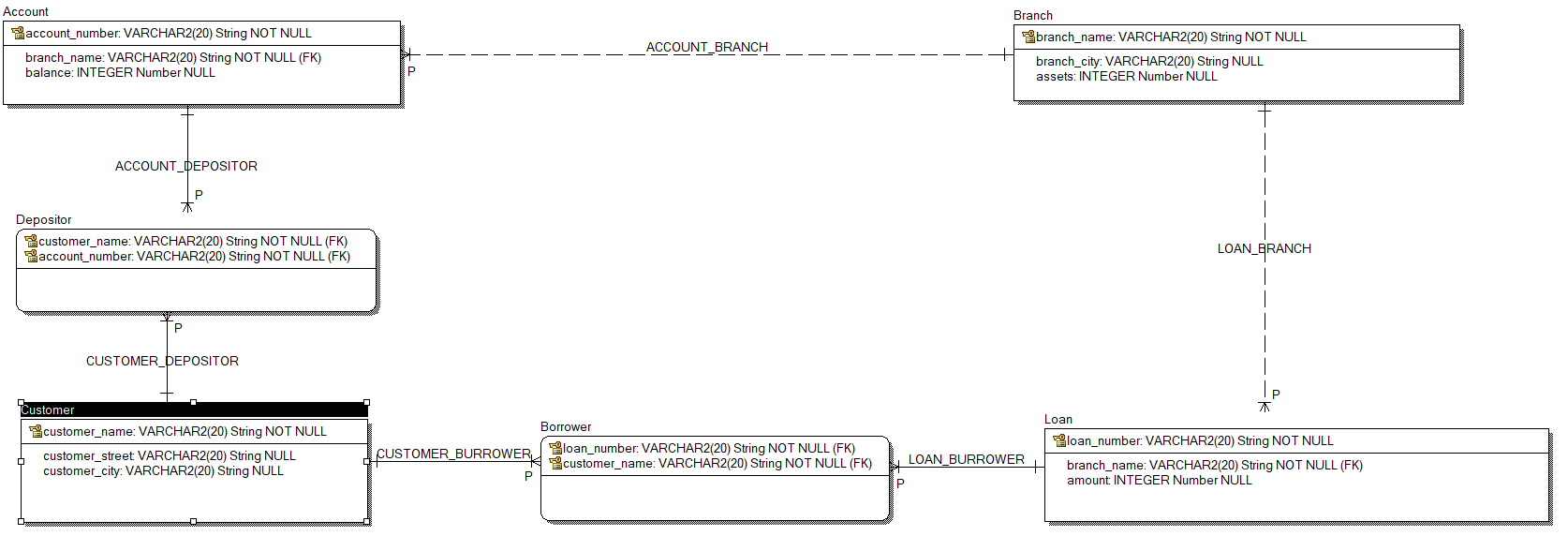
account Entity와는 account\_branch라는 relationship을 loan Entity와는 loan\_branch라는

relationship을 가진다. loan Entity는 대출을 나타내며 loan\_number(대출번호)을 primary key로 가

지며 이에 더불어 amount(대출한 금액)를 attribute로 가지고 있다. 그리고 customer Entity와는

borrower라는 relationship을 branch Entity와는 loan\_branch라는relationship을 가진다.

**3. Physical Mode**



**<Physical Mode>**

**3-1. 사용된 Domain**

1. string : 문자열을 저장하고 이때 datatype은 varchar를 이용하였다.

ACCOUNT Entity(계좌)의 ACCOUNT\_NUMBER(계좌번호) , BRANCH\_NAME(지점이름)과

CUSTOMER Entity(고객)의 CUSTOMER\_NAME(고객이름) , CUSTOMER\_STREET(고객 거주거리) ,

CUSTOMER\_CITY(고객 거주 도시)와 BRANCH Entity의 BRANCH\_NAME(지점이름) , BRANCH\_CITY(지점 입주 도시)와 LOAN Entity의 BRANCH\_NAME(지점이름) , LOAN\_NUMBER(대출번호)를 나타내는데 사용한다.

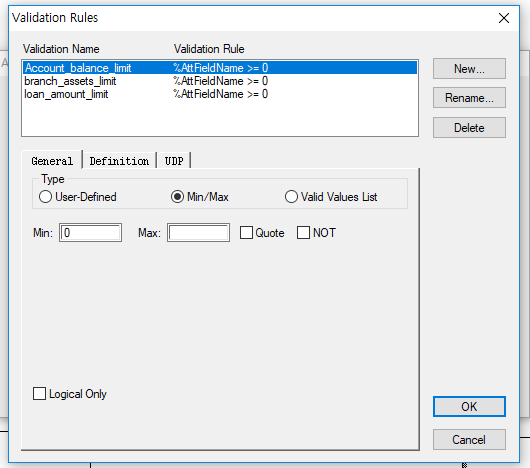
1. number : 돈 액수를 저장하고 이때 datatype은 INTEGER이다. 이 datatype은

Logical Mode에서 datatype인 MONEY에 해당하므로 0이상이다.

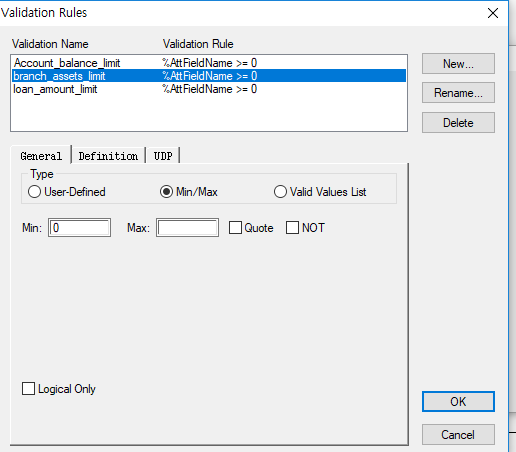
ACCOUNT Entity(계좌)의 BALANCE(잔액)과 BRANCH Entity의 ASSETS(자산)과 LOAN Entity의 AMOUNT(대출액)을 나타내는데 사용한다.

**3-2. 사용된 Validation Constraint** :

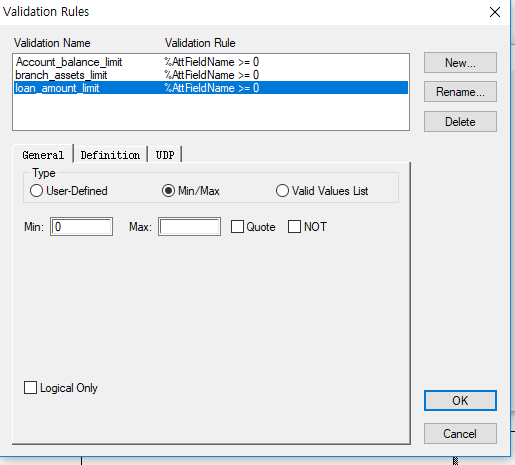
(1) ACCOUNT\_balance\_limit : 이는 ACCOUNT Entity Set의 BALANCE의 Validation Rule로서, 계좌의 잔액은 0원 이상이어야 한다.



(2)branch\_asset\_limit : 이는 Branch Entity Set의 asset의 Validation Rule로서 기업의 자산은 0원 이상이어야 한다.



(3) loan\_amount\_limit : 이는 loan Entity Set의 amount의 Validation Rule로서 대출 총액은 0원이 넘어야 한다.



**3-3. ACCOUNT**

계좌를 나타내는 Entity Set이다.

\*ACCOUNT Entity Set의 Attribute :

(1) ACCOUNT\_NUMBER : 계좌의 계좌번호를 나타내는 Attribute이다. 데이터베이스에 입력되는 각각의 계좌마다 다르게 주어지기 때문에 ACCOUNT\_NUMBER를 Primary Key로 사용하였다. Domain은 String이고 NOT NULL이며 datatype은 VARCHAR이다.

(2) BALANCE : 계좌의 잔액을 저장하는 Attribute이다. Domain은 NUMBER이고 datatype은 INTEGER이다. 그리고 계좌만 소유하고 잔고가 0원 일수도 있기 때문에 NULL이 가능하다. 잔고는 0원 이상 이어야 한다.

\*ACCOUNT의 Relationship :

ACCOUNT는 CUSTOMER와 DEPOSOTOR Relationship을 이루고 이때 Cardinality는many-to-many이다. 그리고 ACCOUNT는 BRANCH와 ACCOUNT\_BRANCH Relationship을 이루고 이때 Cardinality는 many-to-one이다. 그리고 ACCOUNT는 이 Relationship에 total participation이다. 따라서 ACCOUNT는 BRANCH의 Primary Key인 BRANCH\_NAME을 Foreign Key로 가진다.

**3-4. CUSTOMER**

고객을 나타내는 Entity Set이다.

\*CUSTOMER Entity Set의 Attribute :

(1) CUSTOMER\_NAME : 고객의 이름을 나타내는 Attribute이다. 데이터베이스에 입력되는 각각의 고객마다 다르게 주어지기 때문에 ACCOUNT\_NUMBER를 Primary Key로 사용하였다. Domain은 String이고 NOT NULL이며 datatype은 VARCHAR이다.

(2) CUSTOMER\_STREET : 고객의 거주 거리를 나타내는 Attribute이다. Domain은 String이고datatype은 VARCHAR이다.

(3) CUSTOMER\_CITY: 고객의 거주 도시를 나타내는 Attribute이다. Domain은 String이고 datatype은 VARCHAR이다.

\*CUSTOMER의 Relationship :

CUSTOMER는 ACCOUNT와 DEPOSOTOR Relationship을 이루고 이때 Cardinality는many-to-many이다. 그리고 CUSTOMER는 LOAN와 BORROWER Relationship을 이루고 이때 Cardinality는 many-to-many이다.

**3-5. BRANCH**

지점을 나타내는 Entity Set이다.

\*BRANCH Entity Set의 Attribute :

(1) BRANCH\_NAME : 지점의 이름을 나타내는 Attribute이다. 데이터베이스에 입력되는 각각의 지점마다 다르게 주어지기 때문에 ACCOUNT\_NUMBER를 Primary Key로 사용하였다. Domain은 String이고 datatype은 VARCHAR이다..

(2) ASSETS : 지점의 자산을 저장하는 Attribute이다. Domain은 NUMBER이고 datatype은 INTEGER이다. 그리고 0이상의 숫자이다.

(3) BRANCH\_CITY : 지점의 입주 도시를 나타내는 Attribute이다. Domain은 String datatype은 VARCHAR이다.

\* BRANCH의 Relationship :

BRANCH는 ACCOUNT와 ACCOUNT\_BRANCH Relationship을 이루고 이때 Cardinality는 one-to-many이다. 그리고 BRANCH는 LOAN와 LOAN\_BRANCH Relationship을 이루고 이때 Cardinality는 one-to-many이다. 이때 LOAN은 이 Relationship에 total participation이다. LOAN은 BRANCH의 Primary Key인 BRANCH\_NAME을 Foreign Key로 가진다.

**3-6. LOAN**

대출을 나타내는 Entity Set이다.

\*LOAN Entity Set의 Attribute :

(1) LOAN\_NUMBER : 대출의 번호를 나타내는 Attribute이다. 데이터베이스에 입력되는 각각의 대출마다 다르게 주어지기 때문에 ACCOUNT\_NUMBER를 Primary Key로 사용하였다. Domain은 String이고 datatype은 VARCHAR이다.

(2) AMOUNT : 대출액수를 저장하는 Attribute이다. Domain은 NUMBER이고 datatype은 INTEGER이다. 그리고 대출금 총 합 AMOUNT는 0 이상이어야 한다.

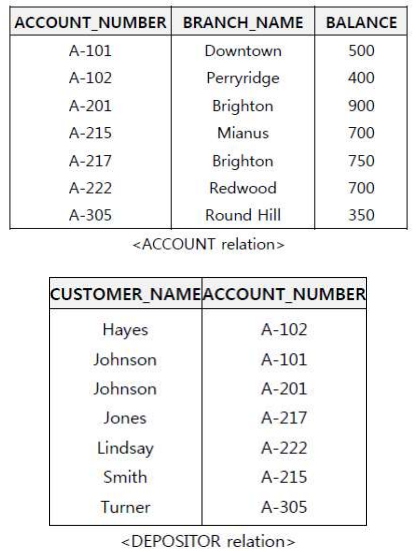
\* LOAN의 Relationship :

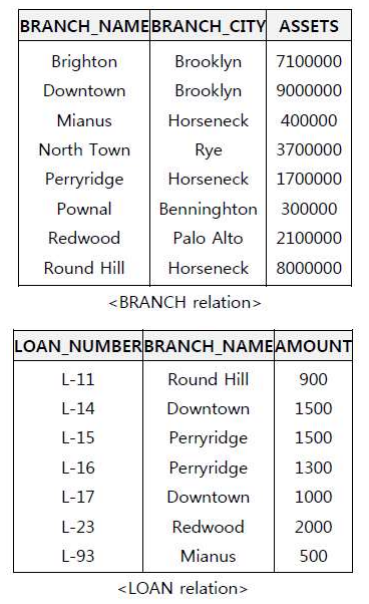
LOAN은 CUSTOMER와 BORROWER Relationship을 이루고 이때 Cardinality는 many-to-many이다. 그리고 LOAN는 BRANCH와 LOAN\_BRANCH Relationship을 이루고 이때 Cardinality는 many-to-one이다. 이때 LOAN은 이 Relationship에 total participation이다.

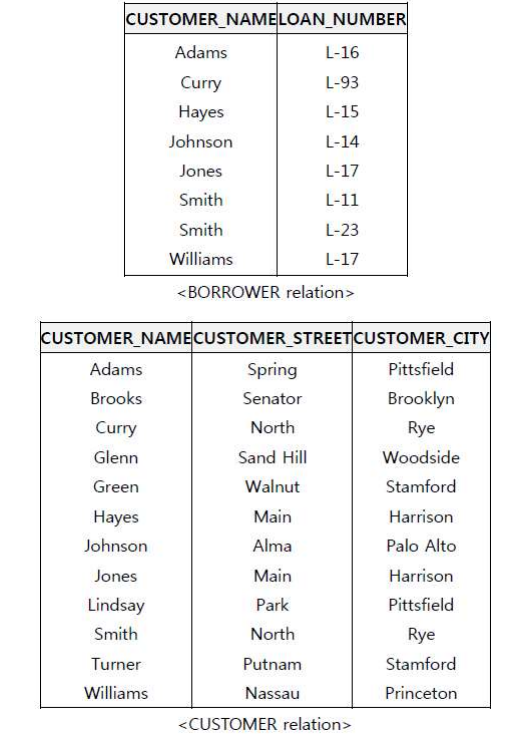
**4. SQL 질의문 및 결과**

**(1) 다음 데이터를 입력하시오 (insert)**

입력할 데이터는 다음과 같다.

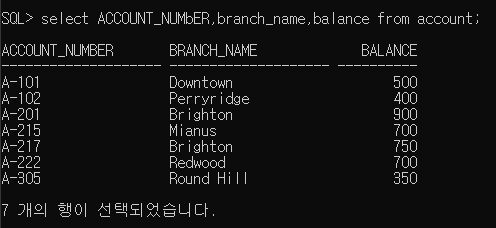






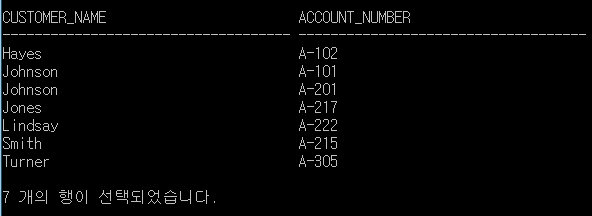
우선 주어진 ER다이어그램으로부터 맞는 ER-win 데이터베이스 스키마를 생성하였다. 이는 앞에 3.Physical Mode를 참고하면 된다. 이 ER-win파일을 실제 dblab.sogang.ac.kr데이터베이스 서버에 연동하여 기본적으로 ACCOUNT Table , CUSTOMER Table , BRANCH Table , LOAN Table, DEPOSITOR Table, BORROWER Table을 생성하였다. 따라서 이에 대한 CREATE TABLE관련된 SQL 질의문을 입력하지 않아도 된다.





Account table 출력은 위와 같다.

DEPOSITOR relation출력은 다음과 같다.



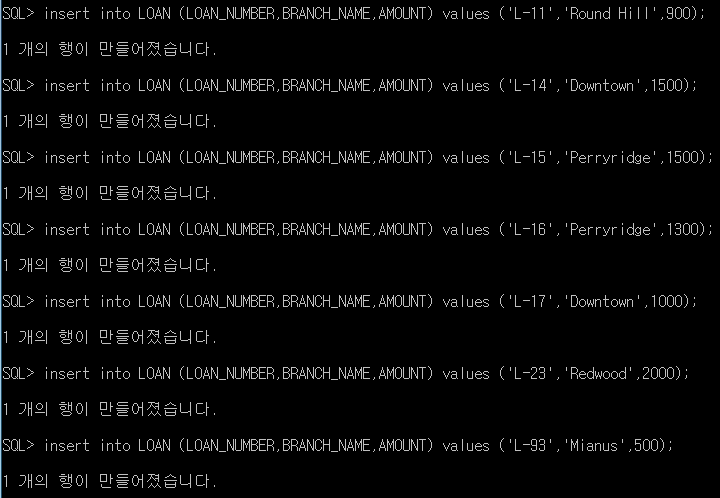
SQL 입력 BRANCH relation은 다음과 같다.



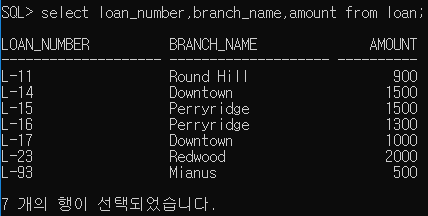
BRANCH relation출력은 다음과 같다.



SQL 입력 LOAN relation은 다음과 같다.

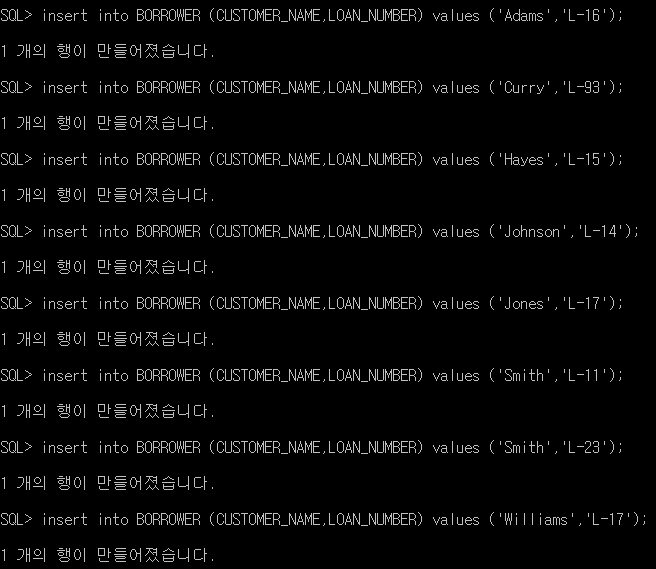


LOAN relation출력은 다음과 같다.

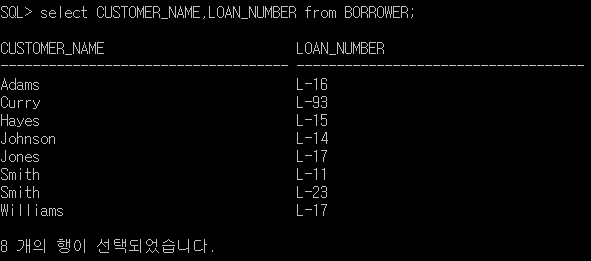


SQL 입력 BORROWER relation은 다음과 같다.





BORROWER relation출력은 다음과 같다.



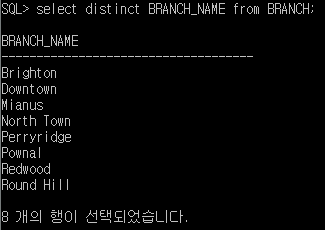
SQL 입력 CUSTOMER relation은 다음과 같다.



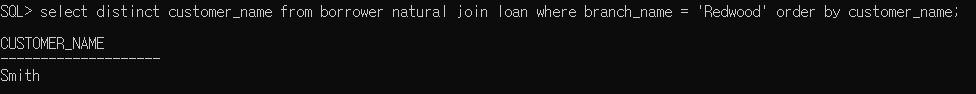
CUSTOMER relation출력은 다음과 같다.



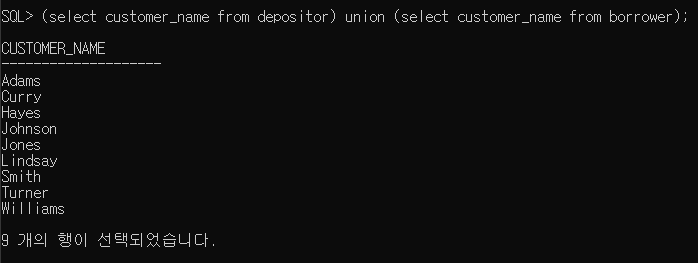
**(2) 중복되지 않은 모든 지점들의 이름을 구하라.**

****

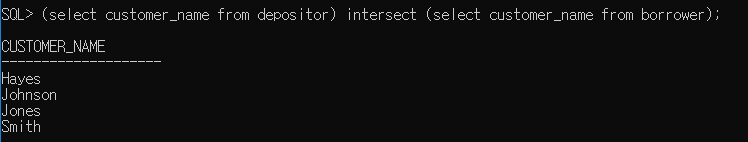
**(3)** Redwood 지점의 대출을 가진 모든 고객들을 알파벳 순서로 나열하라



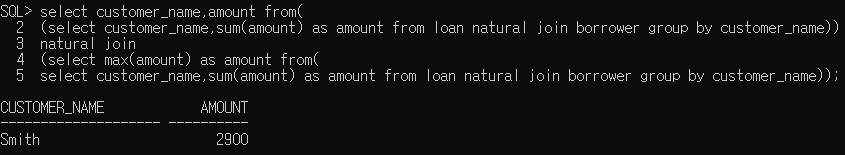
**(4)** 은행에서 대출, 계좌 혹은 둘 다를 가진 모든 고객을 나열하라.



**(5)** 은행에 대출과 계좌 모두를 가진 모든 고객을 나열하라.



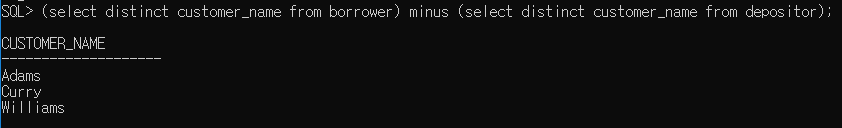
**(6)** 대출 총액이 가장 큰 고객의 이름과 대출 총액을 구하여라.



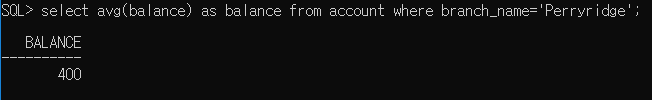
**(7)** Harrison과 Stamford에 살지 않으면서 계좌에 잔고가 1000이하 있는 고객의 이름과 고객이 사는 도시를 구하라.



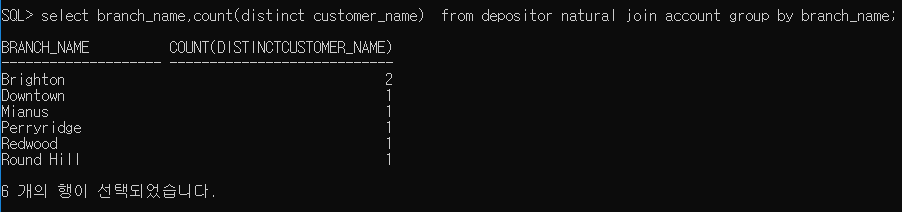
**(8)** 은행에 계좌는 가지고 있지 않지만 대출은 가지고 있는 모든 고객들을 나열하라.

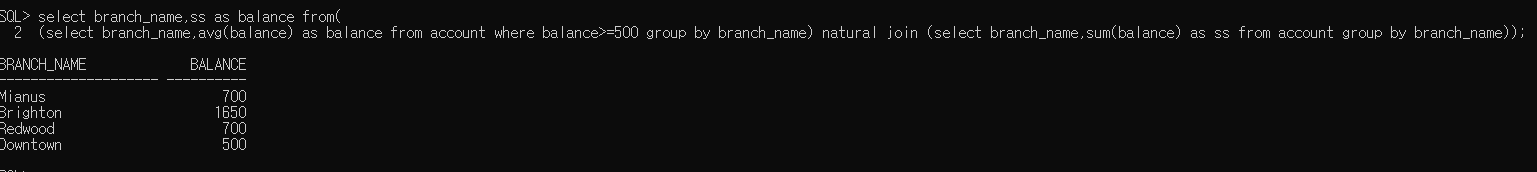


**(9)** Perryridge 지점에서 계좌의 평균 잔고를 구하여라.**.**

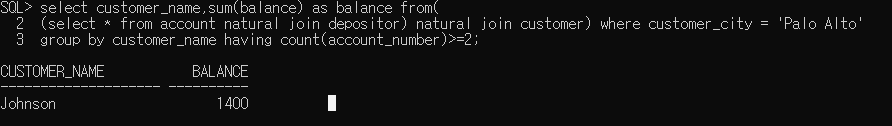


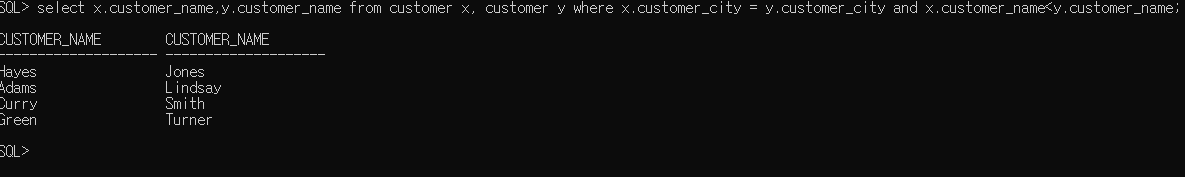
**(10)** 각 지점의 예금자들의 수를 구하라.**.**

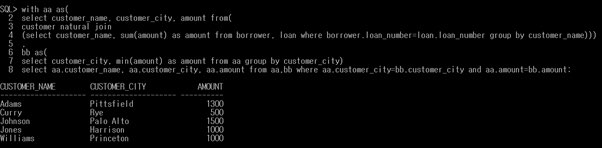


**(11)** 평균 잔고가 $500 이상인 지점 이름과 총 잔고를 나열하라.

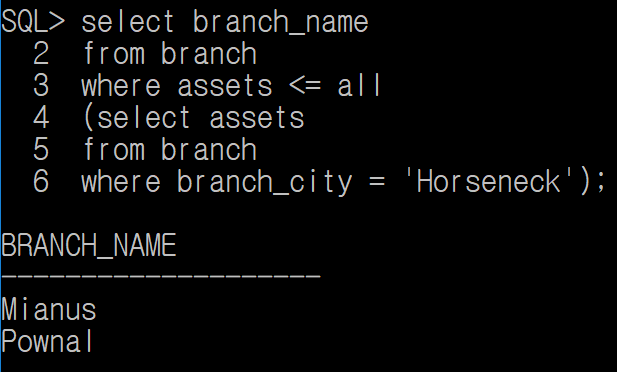
**(12)** Palo Alto에 살고 최소한 두 개의 계좌를 가진 각각의 고객들의 이름과 잔고의 합을 구하라.

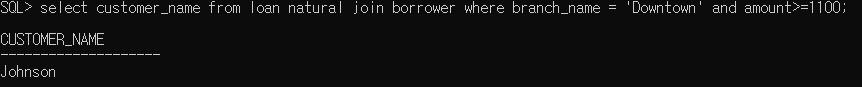


**(13)** 같은 도시에 사는 고객의 이름의 쌍을 구하여라

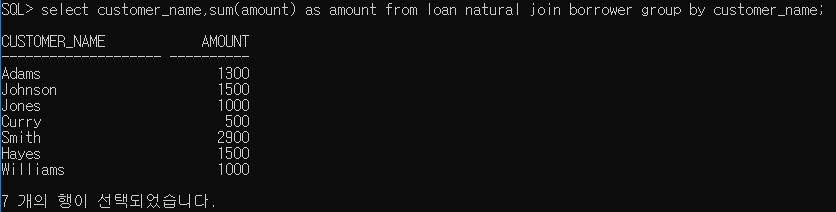
**(14)** 각 도시 별로 가장 낮은 대출 총액을 가지고 있는 고객의 이름과 대출 총액을 구하여라. 단, 대출을 가 진 고객이 살지 않는 도시는 표시하지 않는다.

**(15)** Horseneck에 있는 각 지점보다 작거나 같은 자산 값을 갖는 모든 지점들의 이름을 나열하라.

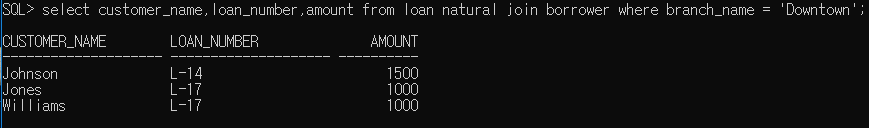


**(16)** Downtown 지점에서 $1100 이상의 대출 총액을 지닌 고객들을 전부 구하라

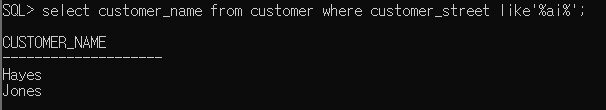
**(17)** 은행에 대출을 가지고 있는 모든 고객들에 대해 그들의 이름과 대출 총액을 구하라.



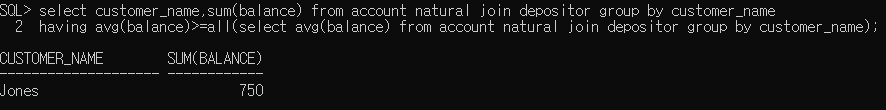
**(18)** Downtown 지점의 모든 대출에 대해여 고객의 이름과 대출 번호, 대출액을 구하라.



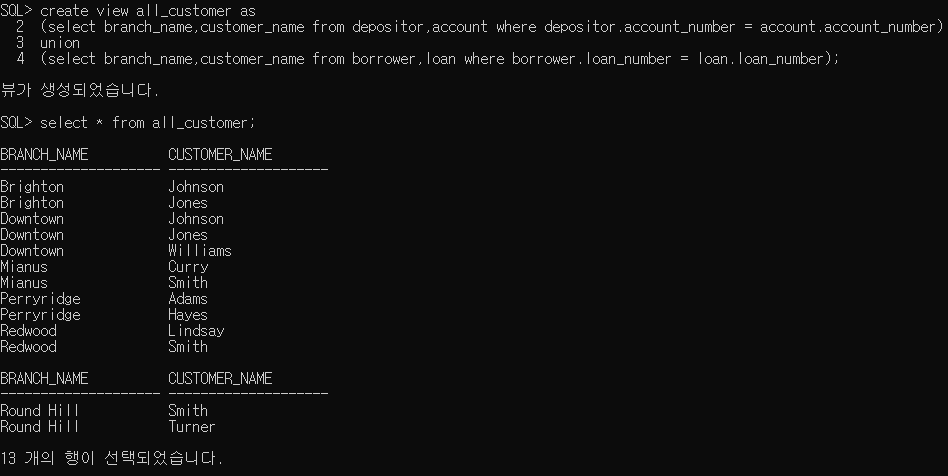
**(19)** 이름에 'ai'이라는 부분 문자열이 포함된 거리에 살고 있는 모든 고객들의 이름을 구하여라.



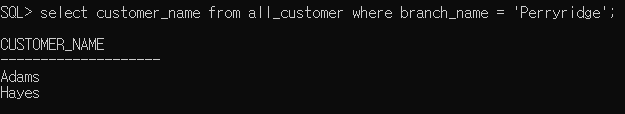
**(20)** 가장 높은 평균 잔고를 가진 고객의 이름과 총잔고를 구하라.



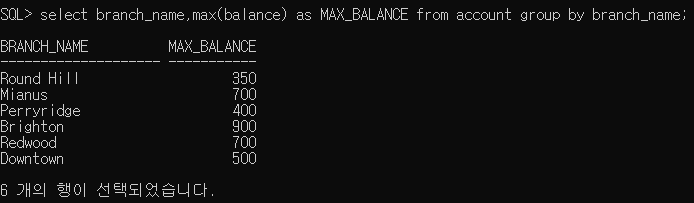
**(21)** 지점 이름과 그 지점에 계좌나 대출 둘 중 하나를 가진 고객 이름으로 구성된 View를 작성하라. (단 View의 이름은 all\_customer이다.)

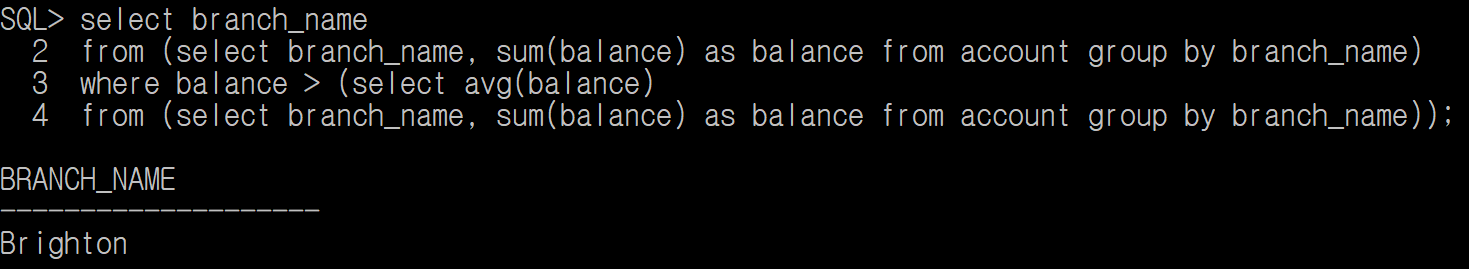


**(22)** 21에서 생성된 View를 이용하여 Perryridge 지점의 모든 고객 이름을 나열하라.



**(23)** 각 지점에서 총 잔고의 최대값을 나열하라.



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

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

