2019 Database System

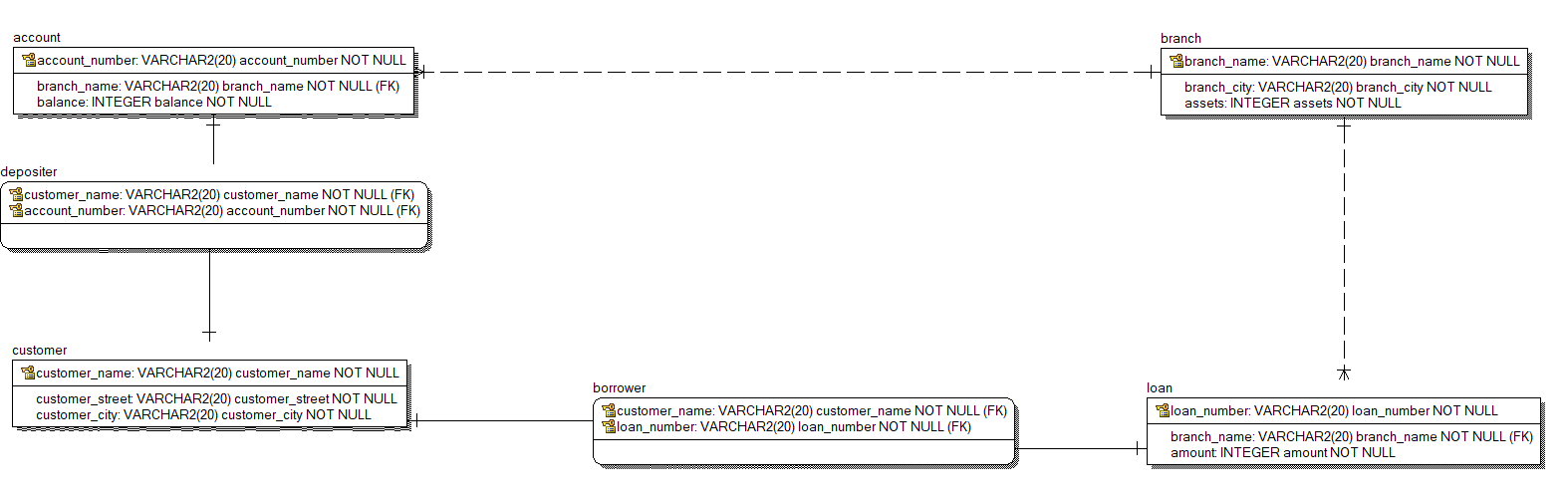
Project #2

이름 : 이승철

학과 : 컴퓨터공학과

학번 : 20141561

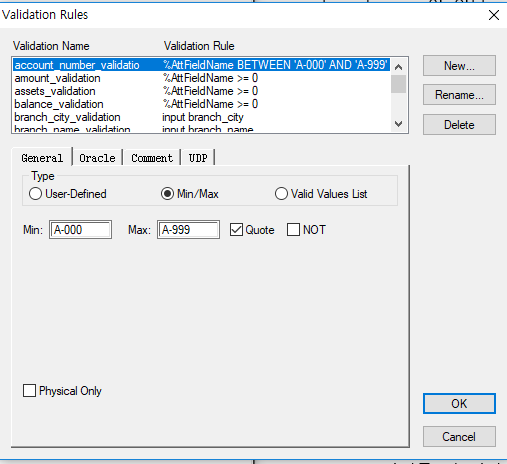
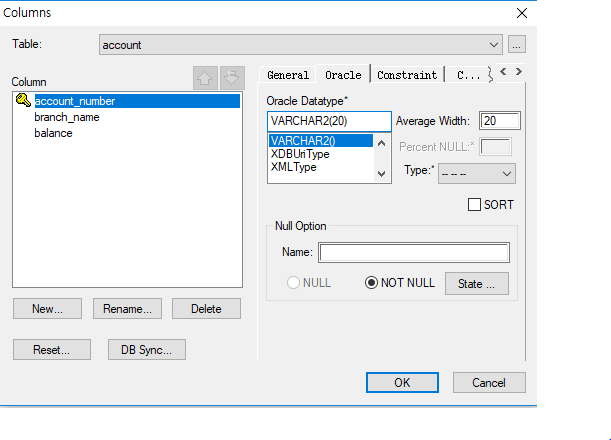
**1. ER Win Physical Mode**

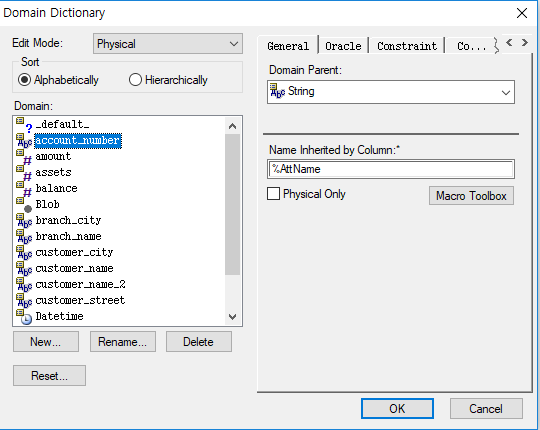


**<ACCOUNT>**

Account는 account\_number를 primary key로 가지고 속성에는 balance와 foreign key인 branch\_name이 있다.

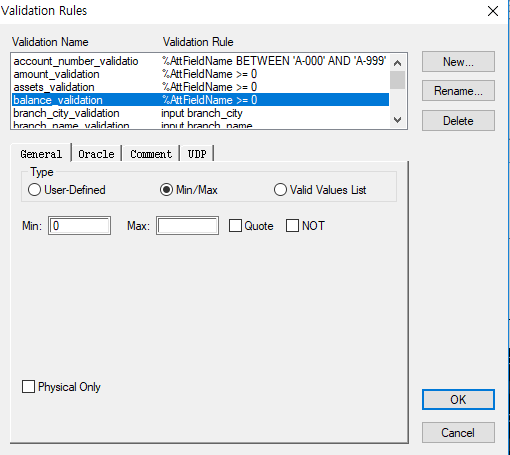
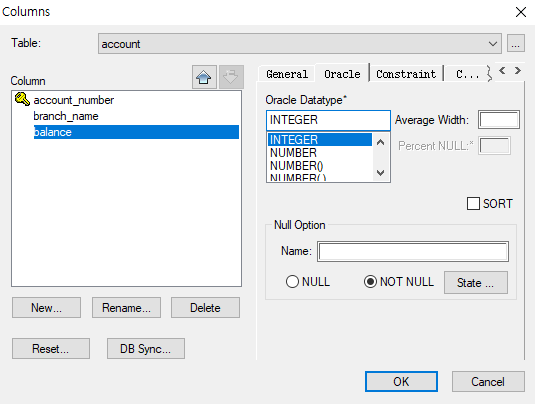
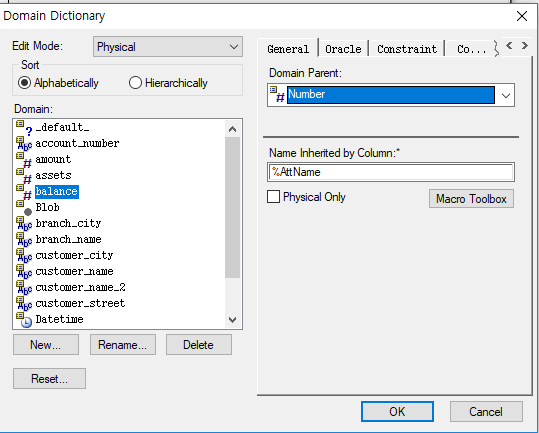
1) account\_number





Account\_number는 string의 type을 가지고 data type은 varchar()와 not null이다. 왜냐하면 account\_number는 사용자의 계정 고유 번호이며 length가 다양한 character string이므로 varchar()의 타입을 갖고 null값을 가지면 안되기 때문이다. 그리고 validation rule에는 최소/최대를 썼는데 이는 최소가 A-000이고 최대가 A-999로 가정을 했기 때문이다.

2) balance

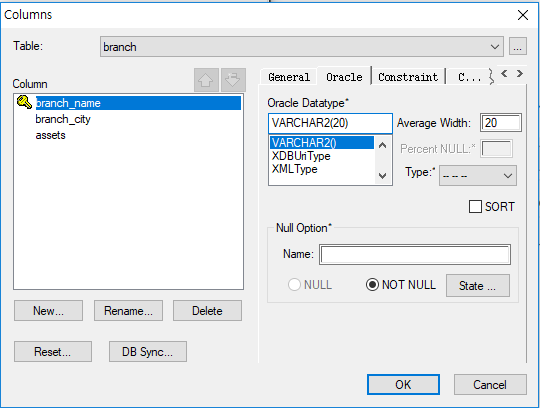
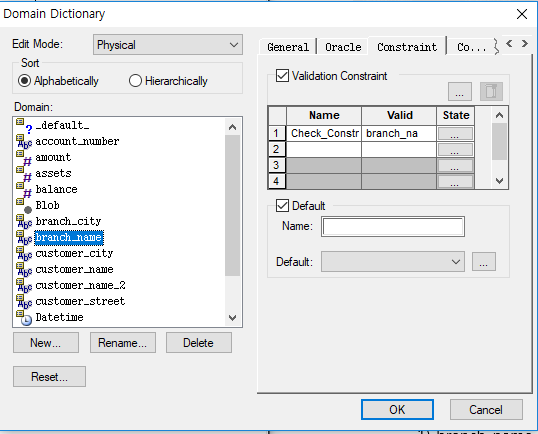
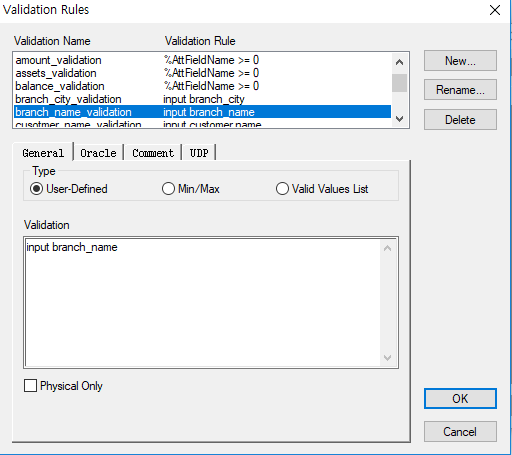


Balance는 number의 type을 가지고 data type은 integer와 not null이다. 왜냐하면 Balance는 잔액을 나타내야 하므로 integer의 data type을 갖고 null값을 가지면 안되기 때문이다. validation에는 최소/최대를 썼는데 이 때 최소가 0이고 최대값을 설정하지 않아서 양수인 정수의 값을 갖도록 했다.

**<BRANCH>**

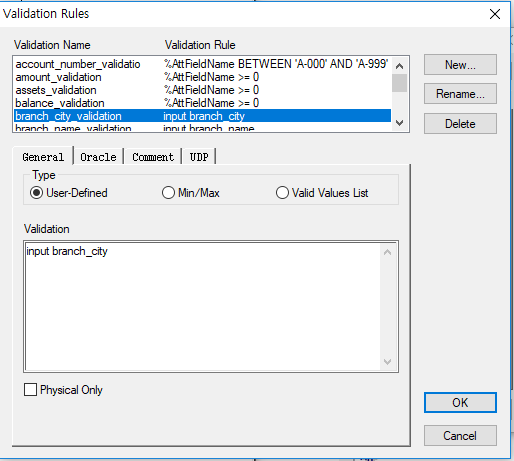
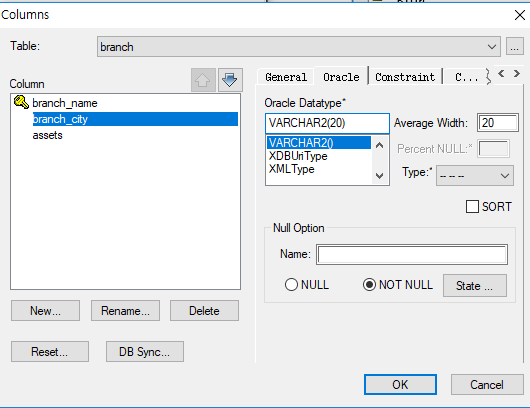
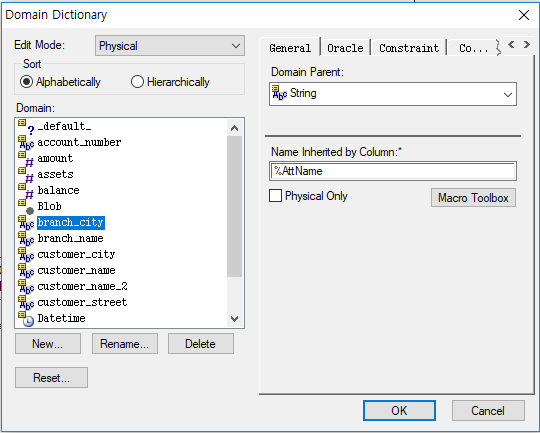
Branch는 branch\_name을 primary key로 가지고 속성에는 branch\_city와 assets이 있다.

1) branch\_name

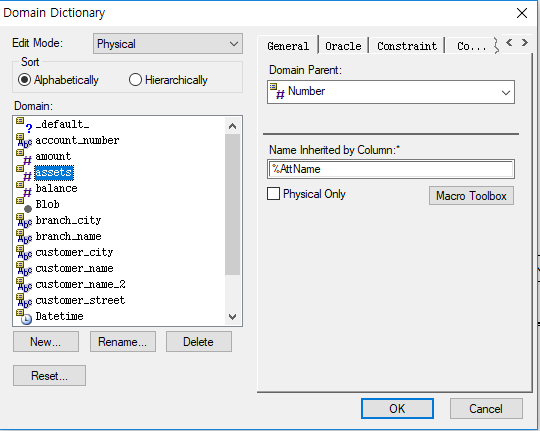
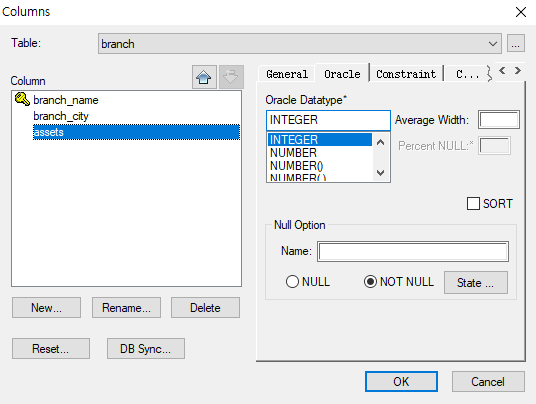
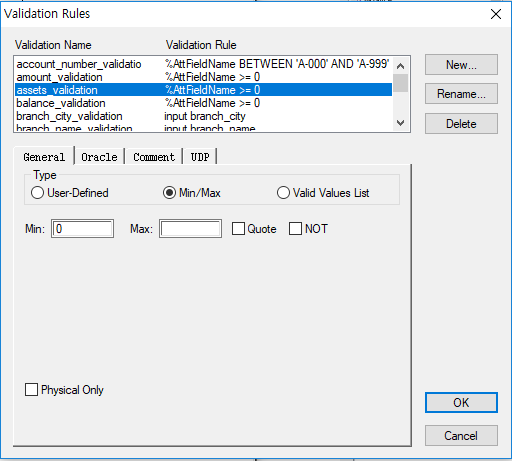
Branch\_name은 string의 type을 가지고 있고 data type은 varchar()와 not null이다. 왜냐하면 branch\_name은 은행 지점의 이름으로 length가 다양한 이름을 가질 수 있고 이름은 반드시 존재하기 때문이다. Validation은 user-defined로 다양한 branch\_name을 받도록 했다.

2) branch\_city



Branch\_city는 string의 type을 가지고 있고 data type은 varchar()와 not null이다. 왜냐하면 branch\_city은 은행 지점의 도시 이름으로 length가 다양한 이름을 가질 수 있고 이름은 반드시 존재하기 때문이다. Validation은 user-defined로 다양한 branch\_city를 받도록 했다.

3) assets

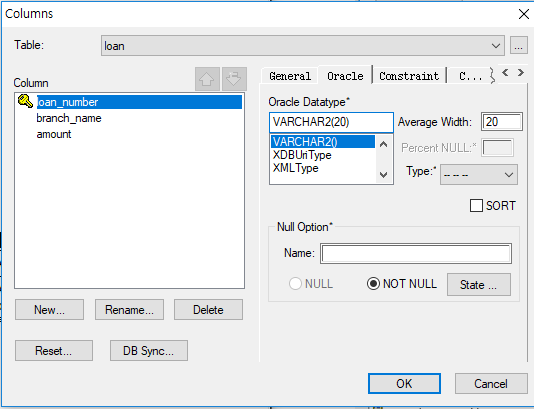
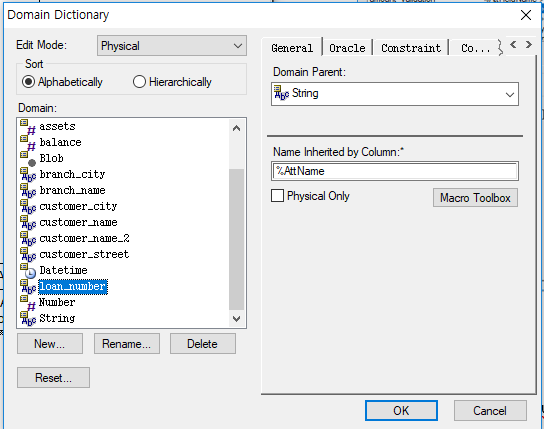
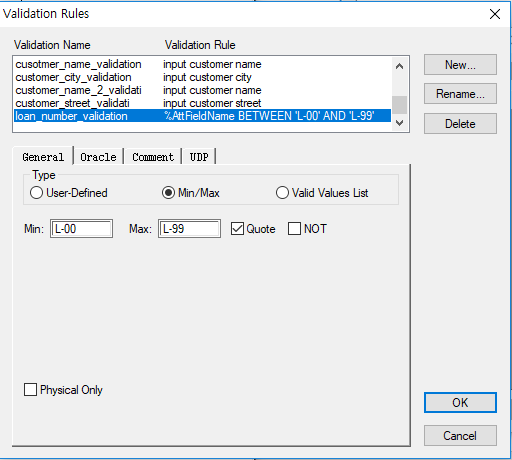
  

Assets은 number의 type을 가지고 있고 data type은 integer와 not null이다. 왜냐하면 assets은 보유 자산으로 정수의 값을 가질 수 있고 자산은 반드시 존재하기 때문이다. Validation은 최소/최대를 썼는데 이 때 최소가 0이고 최대값을 설정하지 않아서 양수인 정수의 값을 갖도록 했다.

**<LOAN>**

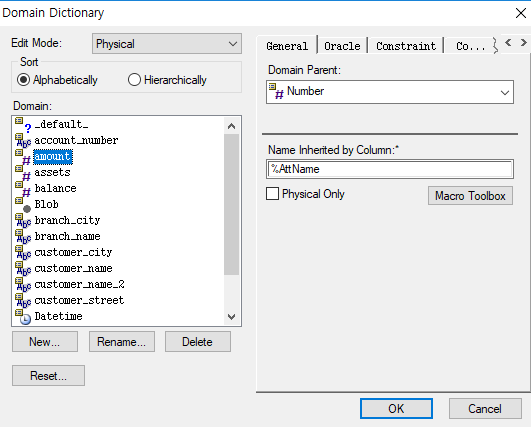
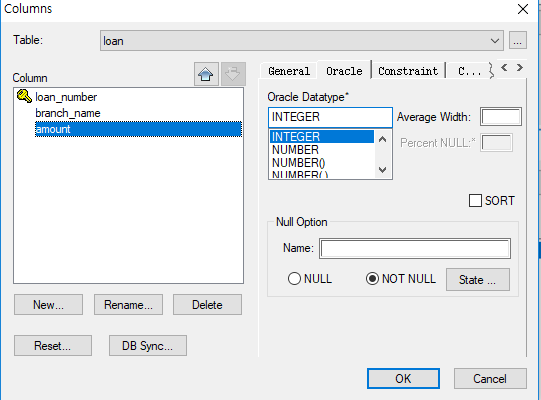
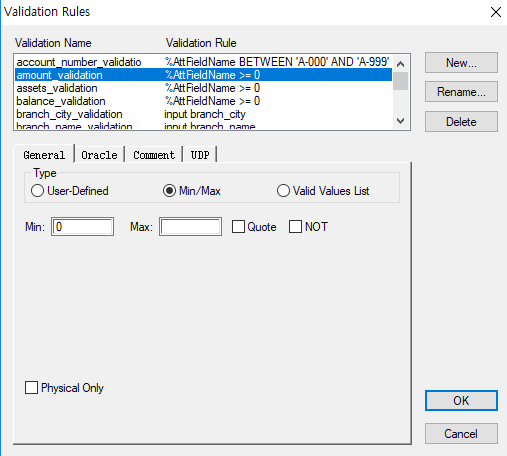
Loan은 loan\_number를 primary key로 가지고 속성에는 amount와 foreign\_key인 branch\_name이 있다.

1) loan\_number

Loan\_number는 string의 type을 가지고 data type은 varchar()와 not null이다. 왜냐하면 loan\_number는 사용자의 대출 고유 번호이며 length가 다양한 character string이므로 varchar()의 타입을 갖고 사용자 한명당 고유번호가 존재하기 때문이다. 그리고 validation rule에는 최소/최대를 썼는데 이는 최소가 L-00이고 최대가 L-99로 가정을 했기 때문이다.

2) amount

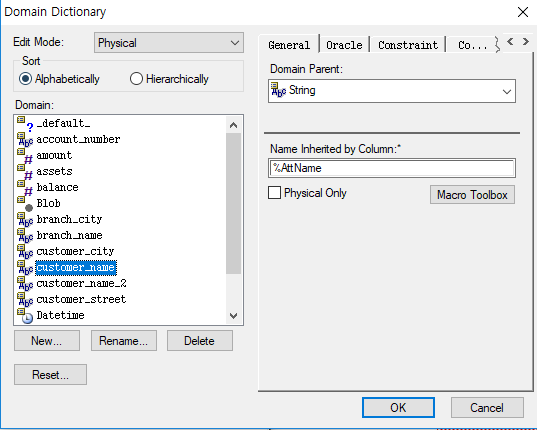
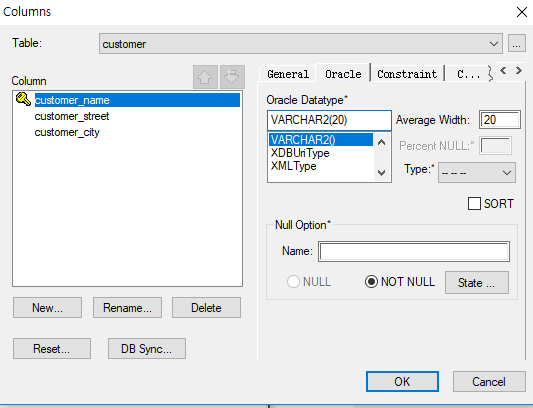
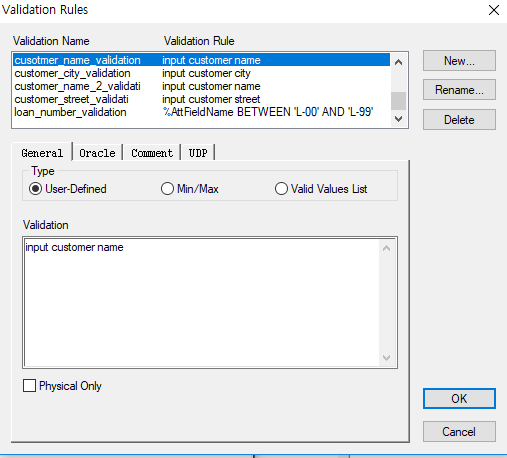
  

Amount은 number의 type을 가지고 있고 data type은 integer와 not null이다. 왜냐하면 Amount은 대출 금액으로 정수의 값을 가질 수 있고 대출이 있는 한 대출 금액은 반드시 존재하기 때문이다. Validation은 최소/최대를 썼는데 이 때 최소가 0이고 최대값을 설정하지 않아서 양수인 정수의 값을 갖도록 했다.

**<CUSTOMER>**

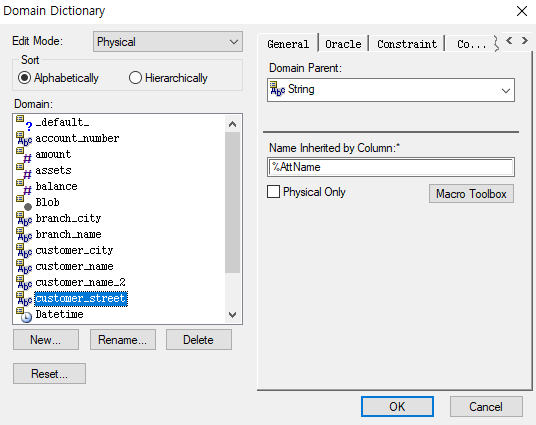
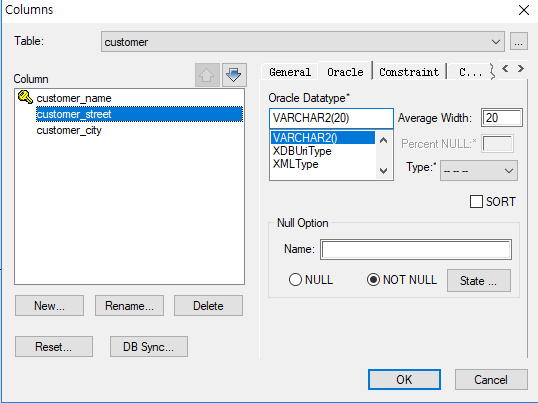
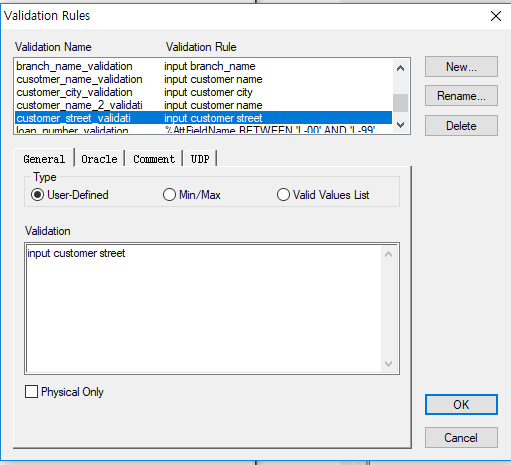
Customer는 customer\_name을 primary key로 가지고 속성에는 customer\_street와 customer\_city가 있다.

1) customer\_name

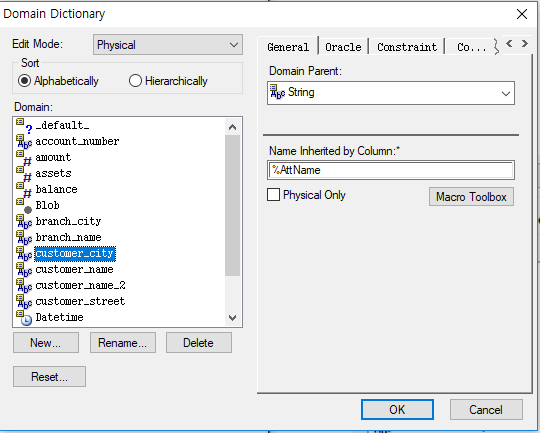
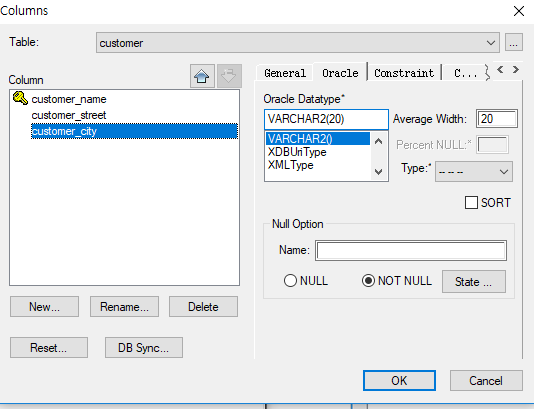
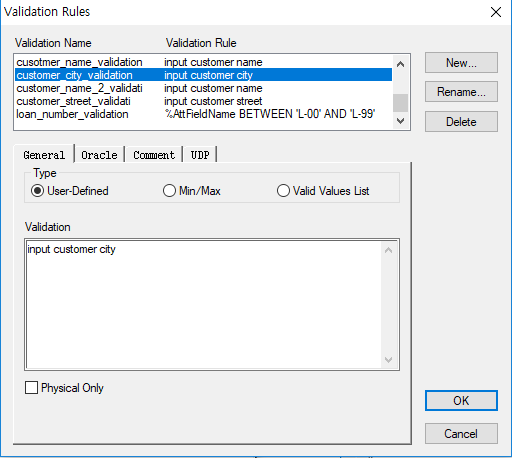
Customer\_name은 string의 type을 가지고 있고 data type은 varchar()와 not null이다. 왜냐하면 customer\_name은 고객 이름으로 length가 다양한 이름을 가질 수 있고 이름은 반드시 존재하기 때문이다. Validation은 user-defined로 다양한 customer\_name을 받도록 했다.

2) customer\_street

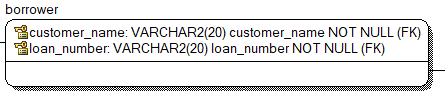
Customer\_street은 string의 type을 가지고 있고 data type은 varchar()와 not null이다. 왜냐하면 customer\_street은 고객 주소 도로 이름으로 length가 다양한 이름을 가질 수 있고 도로 이름은 반드시 존재하기 때문이다. Validation은 user-defined로 다양한 customer\_street을 받도록 했다.

3) customer\_city

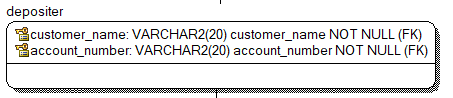
Customer\_city는 string의 type을 가지고 있고 data type은 varchar()와 not null이다. 왜냐하면 customer\_city은 고객 주소 도시 이름으로 length가 다양한 이름을 가질 수 있고 도시 이름은 반드시 존재하기 때문이다. Validation은 user-defined로 다양한 customer\_city를 받도록 했다.

**<BORROWER>**



Borrower는 loan과 customer 사이의 관계를 보여주는 관계로 고객의 대출현황을 나타내는 relation이다. loan의 loan\_number를 foreign key로, customer의 customer\_name을 foreign key로 가진다.

**<DEPOSITER>**

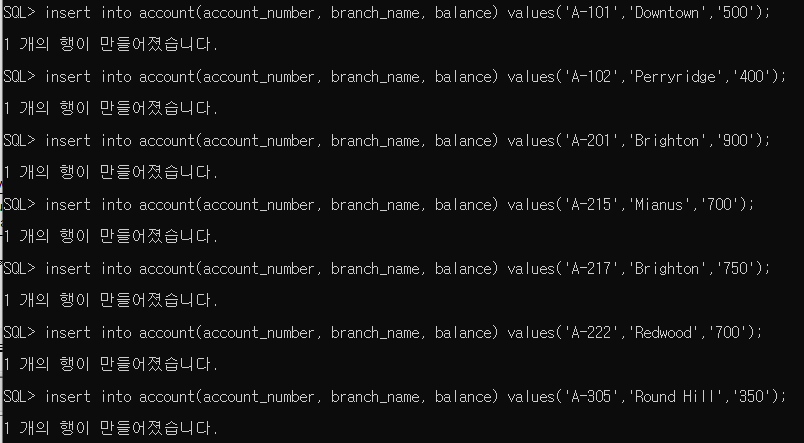


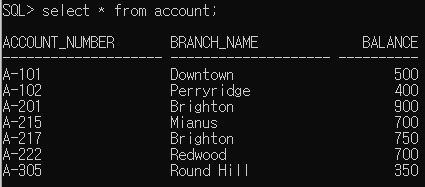
Depositer는 account와 customer사이의 관계를 보여주는 관게로 고객의 예금 현황을 나타내는 relation이다. account의 account\_number를 foreign key로, customer의 customer\_name을 foreign key로 가진다..

**2. SQL 질의문 및 결과**

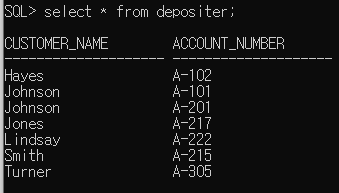
**1) 다음 데이터를 입력하시오 (insert)**

**<ACCOUNT relation>**

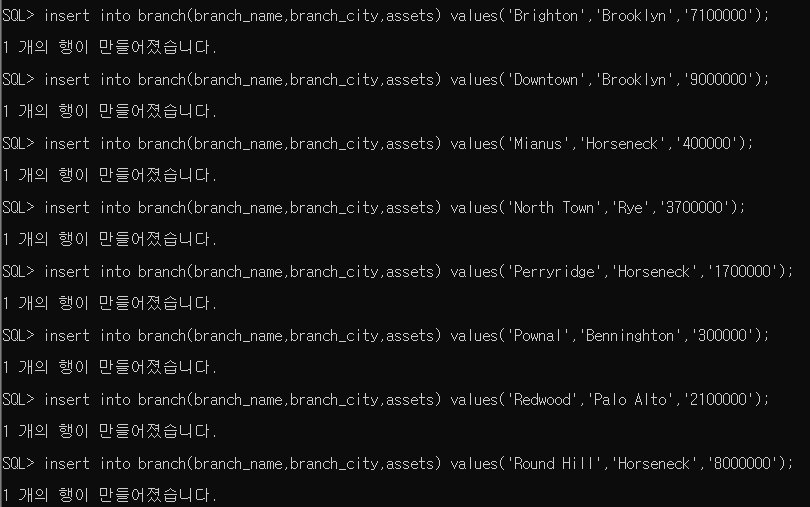




**<DEPOSITER relation>**

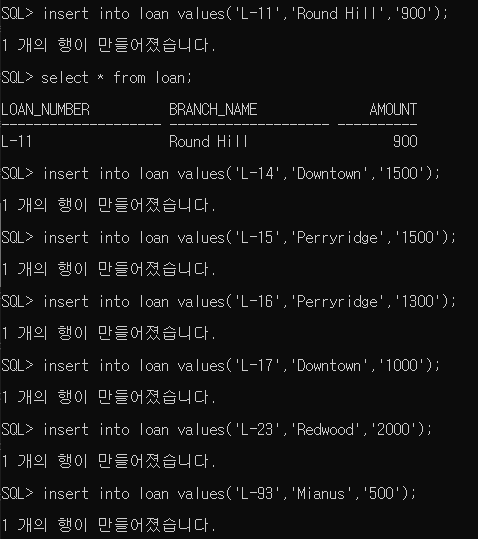


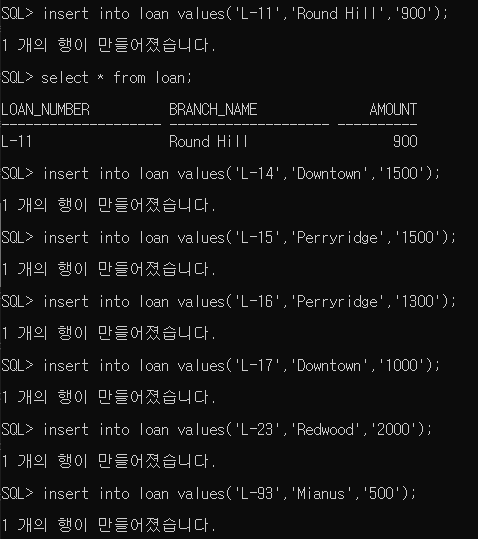
**<BRANCH relation>**

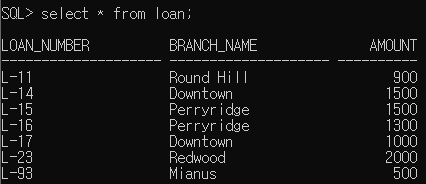




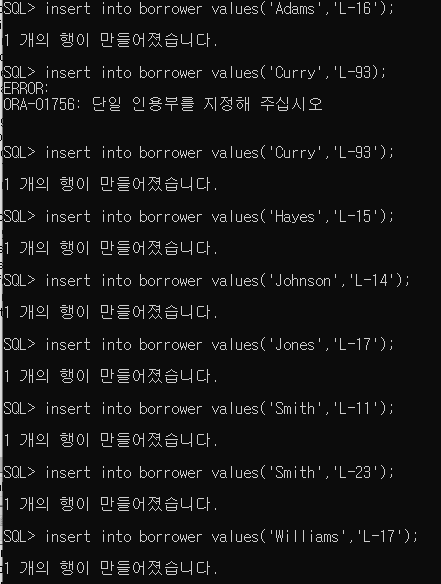
**<LOAN relation>**

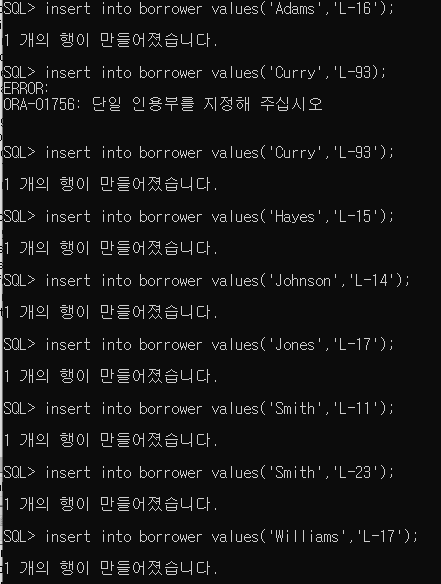


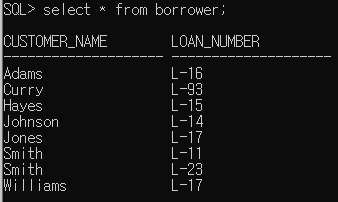




**<BORROWER relation>**





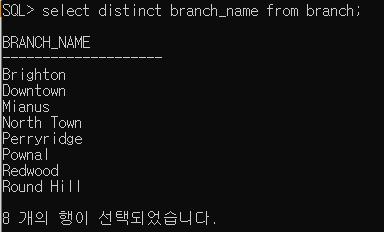


**<CUSTOMER relation>**

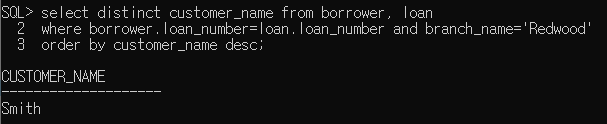




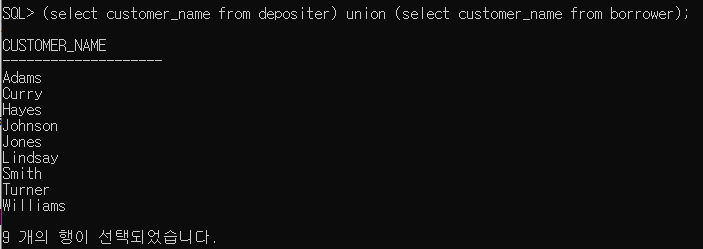
**2) 중복되지 않은 모든 지점들의 이름을 구하라.**



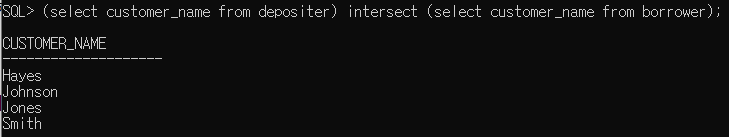
**3) Redwood 지점의 대출을 가진 모든 고객들을 알파벳 역순으로 나열하라.**



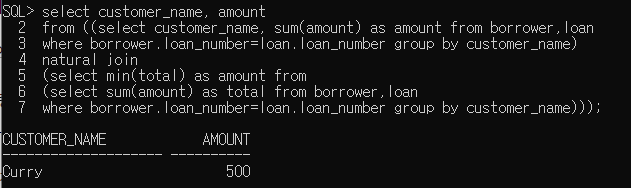
**4) 은행에서 대출, 계좌 혹은 둘 다를 가진 모든 고객을 나열하라.**



**5) 은행에 대출과 계좌 모두를 가진 모든 고객을 나열하라.**



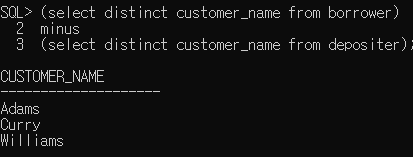
**6) 대출 총액이 가장 작은 고객의 이름과 대출 총액을 구하여라.**



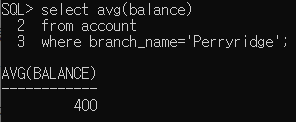
**7) Harrison과 Stamford에 살지 않으면서 계좌에 잔고의 합이 1000이하있는 고객의 이름과 고객이 사는 도시를 구하라.**



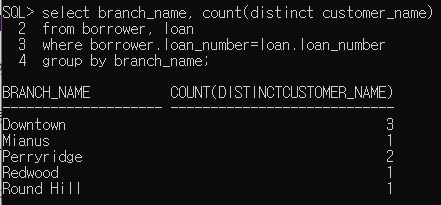
**8) 은행에 계좌는 없지만 대출은 가지고 있는 모든 고객들을 나열하라.**



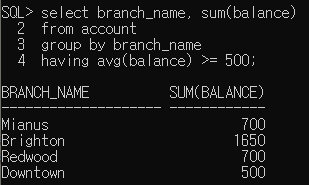
**9) Perryridge 지점에서 계좌의 평균잔고를 구하여라.**



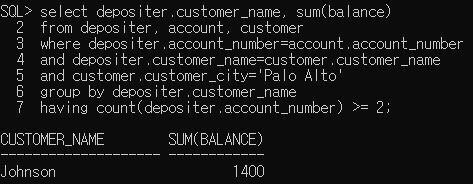
**10) 각 지점의 대출자들의 수를 구하라.**



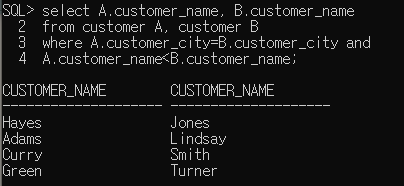
**11) 평균 잔고가 $500 이상인 지점 이름과 총 잔고를 나열하라.**



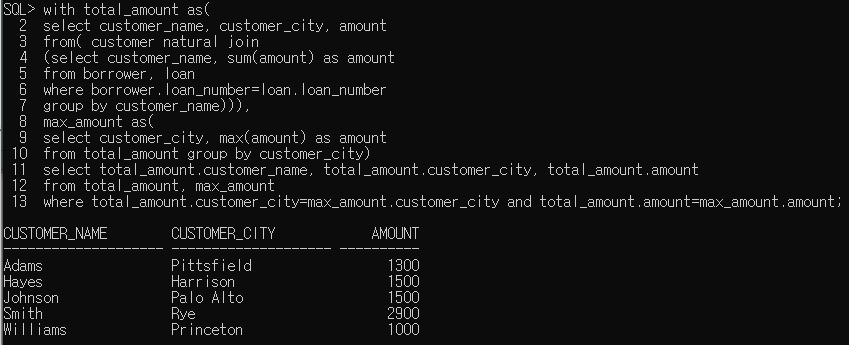
**12) Palo Alto에 살고 최소한 두 개의 계좌를 가진 각각의 고객들의 이름과 잔고의 합을 구하라.**



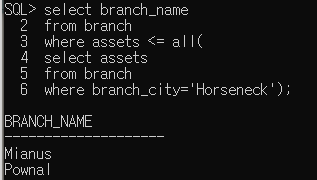
**13) 같은 도시에 사는 고객의 이름의 쌍을 구하여라.**



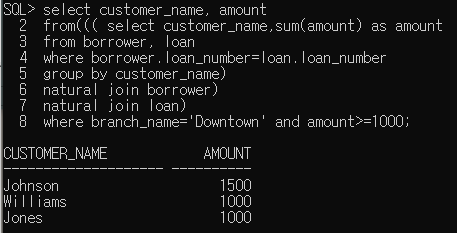
**14) 각 도시별로 가장 높은 대출총액을 가지고 있는 고객의 이름과 대출총액을 구하여라. 단, 대출을 가진 고객이 살지 않는 도시는 표시하지 않는다.**



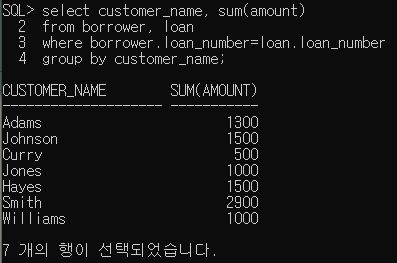
**15) Horseneck에 있는 각 지점보다 작거나 같은 자산값을 갖는 모든 지점들의 이름을 나열하라.**



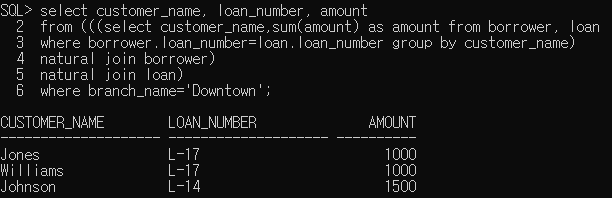
**16) Downtown지점에서 $1000 이상의 대출 총액을 지닌 고객들을 전부 구하라.**



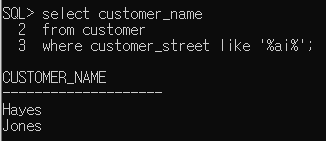
**17) 은행에 대출을 가지고 있는 모든 고객들에 대해 그들의 이름과 대출 총액을 구하라.**



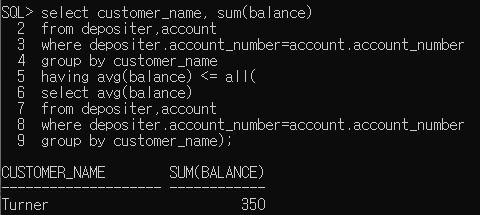
**18) Downtown지점의 모든 대출에 대하여 고객의 이름과 대출번호, 대출액을 구하라.**



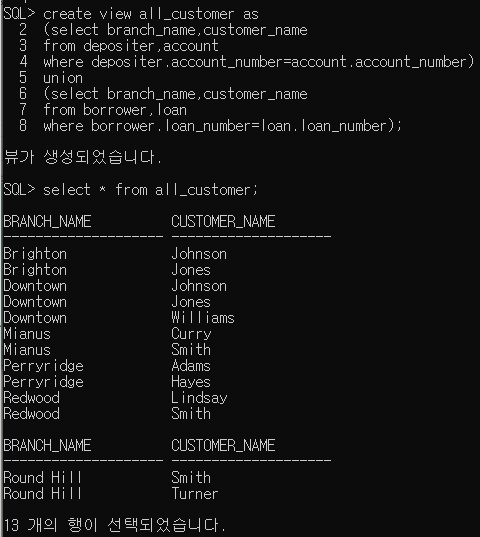
**19) 이름에 'ai'이라는 부분문자열이 포함된 거리에 살고 있는 모든 고객들의 이름을 구하여라.**



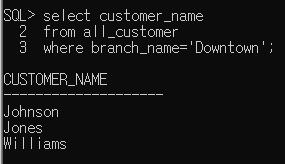
**20) 가장 낮은 평균잔고를 가진 고객의 이름과 총잔고를 구하라.**



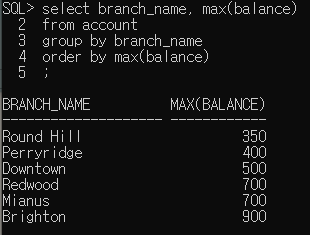
**21) 지점 이름과 그 지점에 계좌나 대출 둘 중 하나를 가진 고객이름으로 구성된 View를작성하라. (단, View의이름은 all\_customer이다.)**



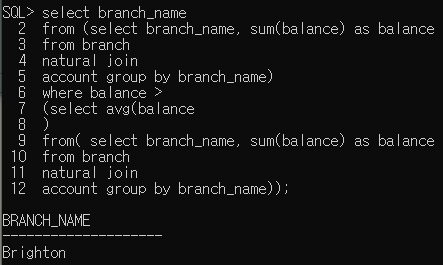
**22) 21에서생성된 View를이용하여 Downtown지점의 모든 고객이름을 나열하라.**



**23) 각 지점에서 총 잔고의 최대값을 나열하라.**



**24) 모든 지점의 총 계좌예금의 평균보다 많은 총계좌예금을 갖는 모든 지점을 나열하라.**



**25) 고객들의 평균 대출총액보다 많은 대출 총액을 가지고 있는 고객의 이름과 대출 총액을 구하라.**

