데이터베이스시스템 프로젝트2

Constructing SQL

2016\*\*\*\*

1. Er-Diagram 데이터 정의
2. Entity

|  |  |  |  |
| --- | --- | --- | --- |
| account | branch | Customer | Loan |
| \*account\_number | \*branch\_name | customer\_name | \*loan\_number |
| Branch\_name | branch\_city | customer\_street | branch\_name |
| balance | Asset | customer\_city | amount |

* 첫 행은 Entity의 이름이고, 각 Entity의 열은 Attribute 이름을 나타낸다. Attribute 앞에 ‘\*’ 표시는 Private Key를 의미한다.

1. Relation
2. Branch to account :: one to many (zero or more)

각 은행 지점에서 관리하는 계좌 정보를 표현한다. 한 은행에서 여러 개의 계좌를 가지고 있을 수 있으므로, entity 간 관계는 one to many로 설정한다.

1. branch to loan :: one to many (zero or more)

각 은행에서 관리하는 대출 상품을 표현한다. 한 은행 당 서로 다른 여러 개의 대출 상품이 있을 수 있기 때문에 entity 간 관계는 one to many 로 설정한다.

1. customer to loan (borrower) :: many to many (zero or more)

대출 상품을 이용하는 고객 정보를 표현한다. 한 명의 고객이 여러 개의 대출 상품에 가입하거나, 하나의 대출 상품에 여러 명의 고객이 가입할 수 있으므로 many to many로 entity의 관계를 설정한다.

1. account to customer (depositor) :: many to many (zero or more)

계좌를 관리하는 고객의 정보를 의미한다. 한 사람이 여러 개의 통장을 가질 수 있고, 법인 통장이나 모임 통장, 데이트 통장 등의 경우 하나의 계좌를 여러 명의 사람이 관리하는 것도 가능하기 때문에 entity 간의 관계는 many to many로 설정한다.

2. SQL 질의문 및 결과

1) 다음 데이터를 입력하시오 (insert)

insert into account values ('A-101', 500, 'Downtown');

insert into account values ('A-102', 400, 'Perryridge');

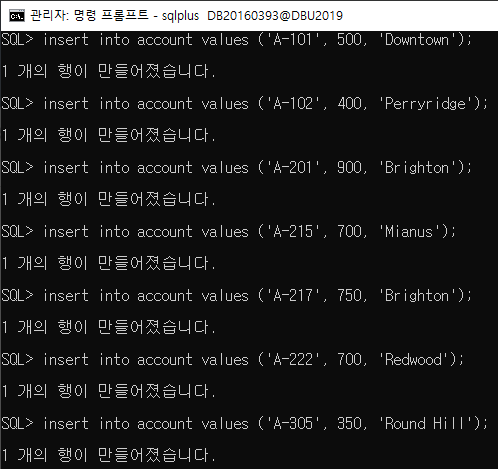
insert into account values ('A-201', 900, 'Brighton');

insert into account values ('A-215', 700, 'Mianus');

insert into account values ('A-217', 750, 'Brighton');

insert into account values ('A-222', 700, 'Redwood');

insert into account values ('A-305', 350, 'Round Hill');



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');



insert into loan values ('L-11', 900, 'Round Hill');

insert into loan values ('L-14', 1500, 'Downtown');

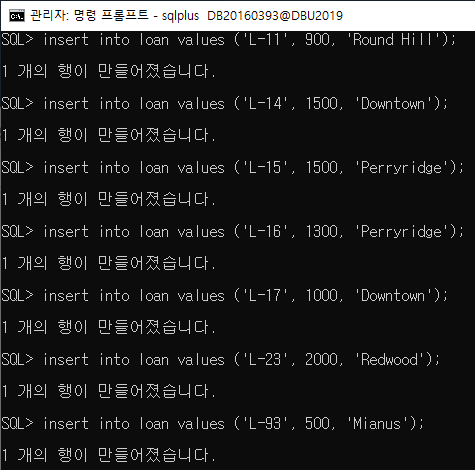
insert into loan values ('L-15', 1500, 'Perryridge');

insert into loan values ('L-16', 1300, 'Perryridge');

insert into loan values ('L-17', 1000, 'Downtown');

insert into loan values ('L-23', 2000, 'Redwood');

insert into loan values ('L-93', 500, 'Mianus');



insert into branch values ('Brighton', 'Brooklyn', 7100000);

insert into branch values ('Downtown', 'Brooklyn', 9000000);

insert into branch values ('Mianus', 'Horseneck', 400000);

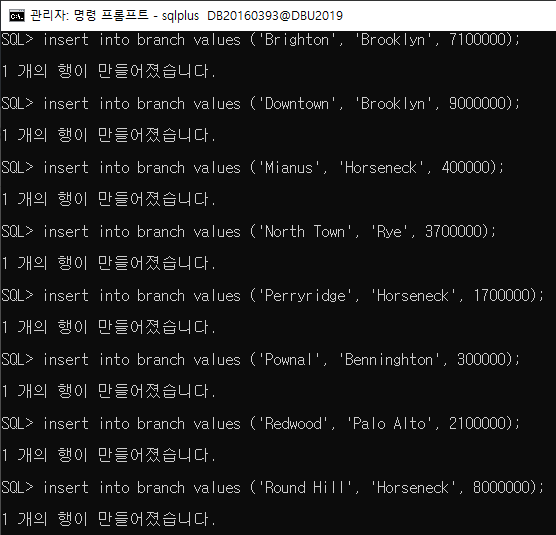
insert into branch values ('North Town', 'Rye', 3700000);

insert into branch values ('Perryridge', 'Horseneck', 1700000);

insert into branch values ('Pownal', 'Benninghton', 300000);

insert into branch values ('Redwood', 'Palo Alto', 2100000);

insert into branch values ('Round Hill', 'Horseneck', 8000000);



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');



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');

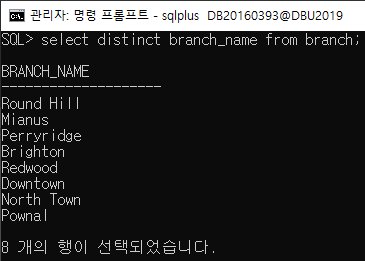
insert into customer values('Smith', 'North', 'Rye');

insert into customer values('Turner', 'Putnam', 'Stamford');



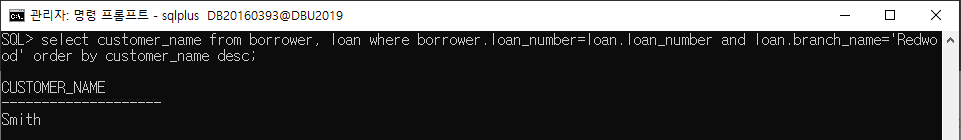
2) 중복되지 않은 모든 지점들의 이름을 구하라.

select distinct branch\_name from branch;



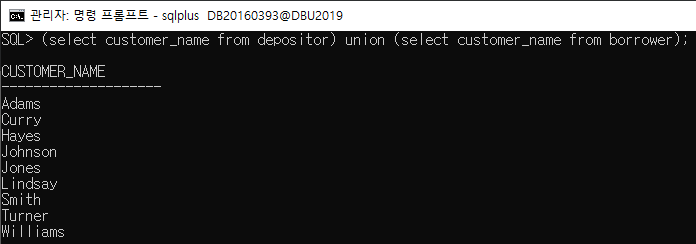
3) Redwood 지점의 대출을 가진 모든 고객들을 알파벳 역순으로 나열하라.

select customer\_name from borrower, loan where borrower.loan\_number=loan.loan\_number and loan.branch\_name='Redwood' order by customer\_name desc;



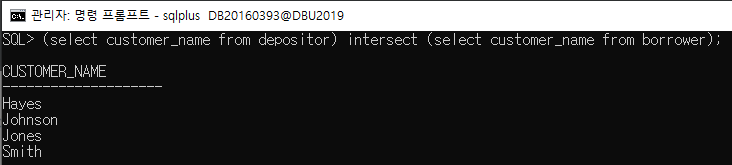
4) 은행에서 대출, 계좌 혹은 둘 다를 가진 모든 고객을 나열하라.

(select customer\_name from depositor) union (select customer\_name from borrower);



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

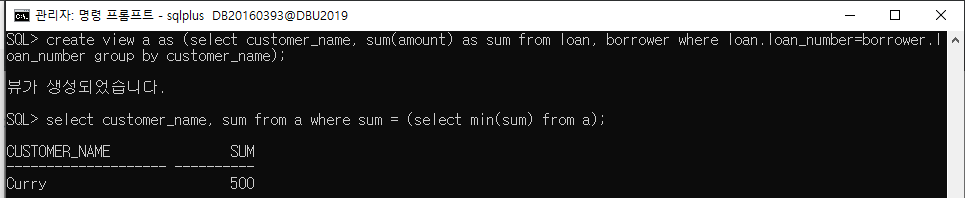
(select customer\_name from depositor) intersect (select customer\_name from borrower);



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

create view a as (select customer\_name, sum(amount) as sum from loan, borrower where loan.loan\_number=borrower.loan\_number group by customer\_name);

select customer\_name, sum from a where sum = (select min(sum) from a);

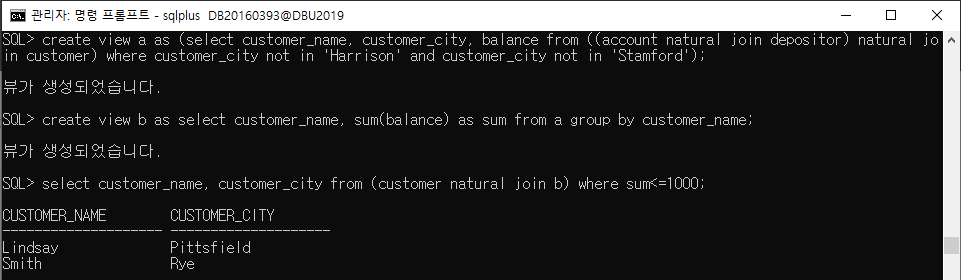


7) Harrison과 Stamford에 살지 않으면서 계좌에 잔고의 합이 1000이하 있는 고객의 이름과 고객이 사는 도시를 구하라.

create view a as (select customer\_name, customer\_city, balance from ((account natural join depositor) natural join customer) where customer\_city not in 'Harrison' and customer\_city not in 'Stamford');

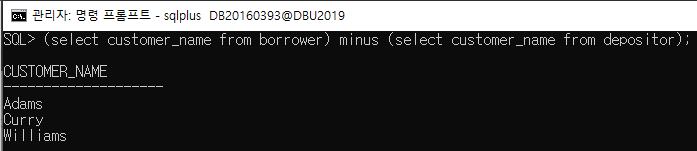
create view b as select customer\_name, sum(balance) as sum from a group by customer\_name;

select customer\_name, customer\_city from (customer natural join b) where sum<=1000;



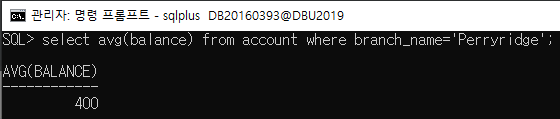
8) 은행에 계좌는 없지만 대출은 가지고 잇는 모든 고객들을 나열하라.

(select customer\_name from borrower) minus (select customer\_name from depositor);



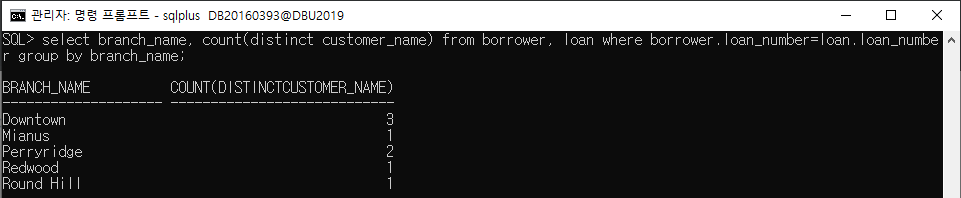
9) Perryridge 지점에서 계좌의 평균 잔고를 구하여라.

select avg(balance) from account where branch\_name='Perryridge';



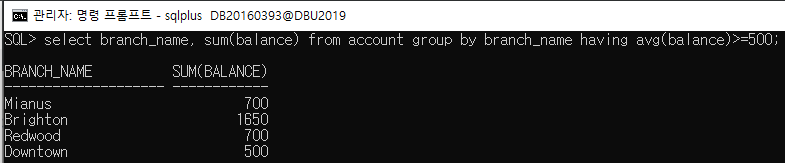
10) 각 지점의 대출자들의 수를 구하라.

select branch\_name, count(distinct customer\_name) from borrower, loan where borrower.loan\_number=loan.loan\_number group by branch\_name;



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

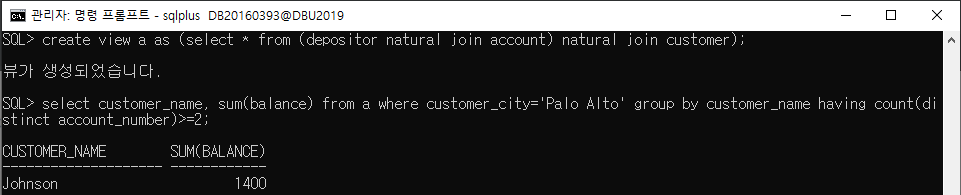
select branch\_name, sum(balance) from account group by branch\_name having avg(balance)>=500;



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

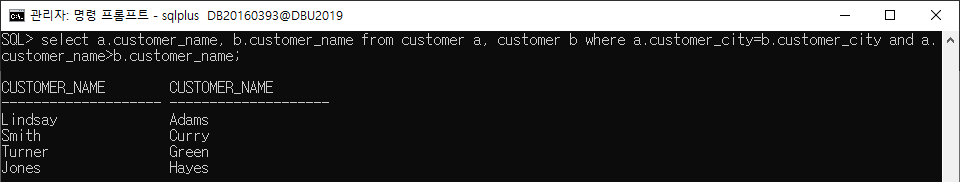
create view a as (select \* from (depositor natural join account) natural join customer);

select customer\_name, sum(balance) from a where customer\_city='Palo Alto' group by customer\_name having count(distinct account\_number)>=2;



13 같은 도시에 사는 고객의 이름의 쌍을 구하여라.

select a.customer\_name, b.customer\_name from customer a, customer b where a.customer\_city=b.customer\_city and a.customer\_name>b.customer\_name;

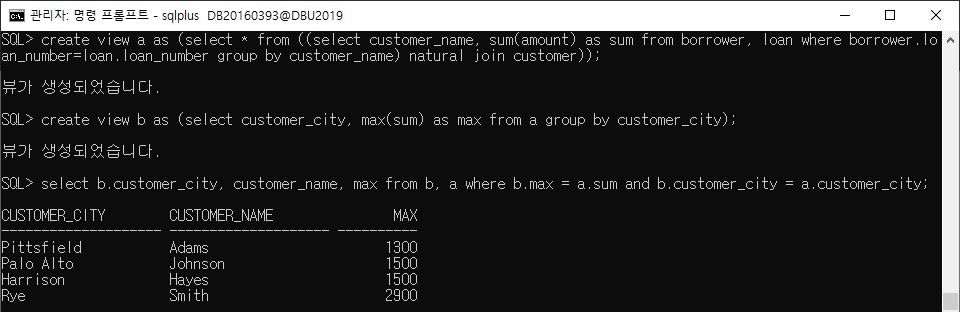


14) 각 도시 별로 가장 높은 대출 총액을 가지고 있는 고객의 이름과 대출 총액을 구하여라. 단 대출을 가진 고객이 살지 않는 도시는 표시하지 않는다.

create view a as (select \* from ((select customer\_name, sum(amount) as sum from borrower, loan where borrower.loan\_number=loan.loan\_number group by customer\_name) natural join customer));

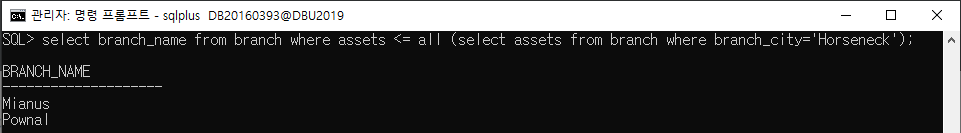
create view b as (select customer\_city, max(sum) as max from a group by customer\_city);

select b.customer\_city, customer\_name, max from b, a where b.max = a.sum and b.customer\_city = a.customer\_city;



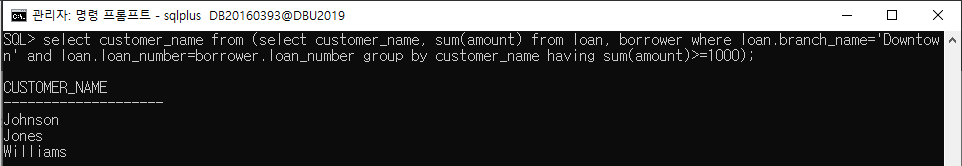
15) Horseneck에 있는 지점들의 자산 중 최소값 보다 작거나 같은 자산 값을 갖는 모든 지점들의 이름을 나열하라

select branch\_name from branch where assets <= min (select assets from branch where branch\_city='Horseneck');



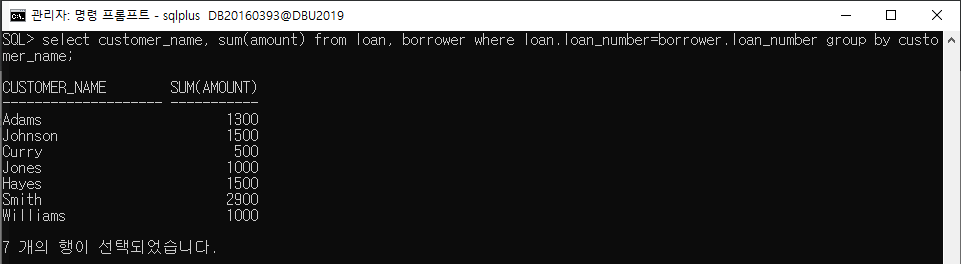
16) Downtown 지점에서 $1000 이상의 대출 총액을 지닌 고객들을 전부 구하라.

select customer\_name from (select customer\_name, sum(amount) from loan, borrower where loan.branch\_name='Downtown' and loan.loan\_number=borrower.loan\_number group by customer\_name having sum(amount)>=1000);



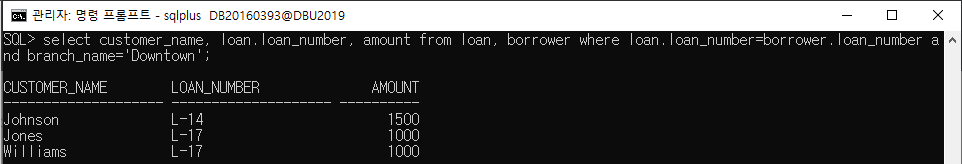
17) 은행에 대출을 가지고 있는 모든 고객들에 대해 그들의 이름과 대출 총액을 구하라.

select customer\_name, sum(amount) from loan, borrower where loan.loan\_number=borrower.loan\_number group by customer\_name;



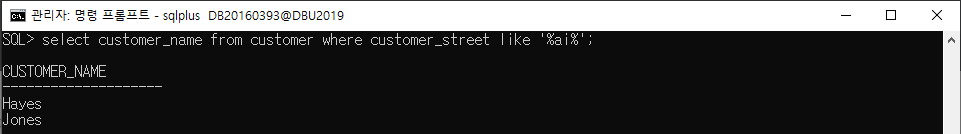
18) Downtown 지점의 모든 대출에 대하여 고객의 이름과 대출 번호, 대출액을 구하라.

select customer\_name, loan.loan\_number, amount from loan, borrower where loan.loan\_number=borrower.loan\_number and branch\_name='Downtown';



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

select customer\_name from customer where customer\_street like '%ai%';

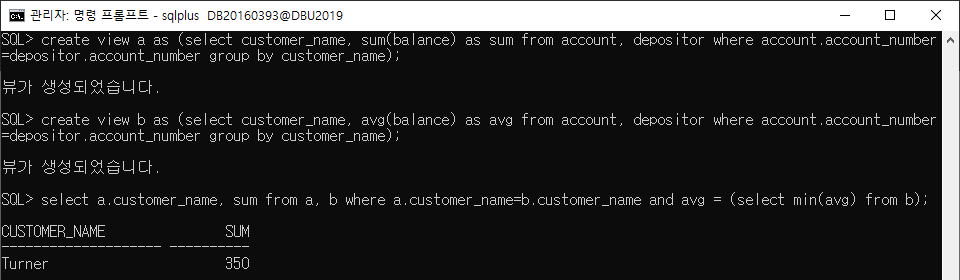


20) 가장 낮은 평균 잔고를 가진 고객의 이름과 총잔고를 구하라.

create view a as (select customer\_name, sum(balance) as sum from account, depositor where account.account\_number=depositor.account\_number group by customer\_name);

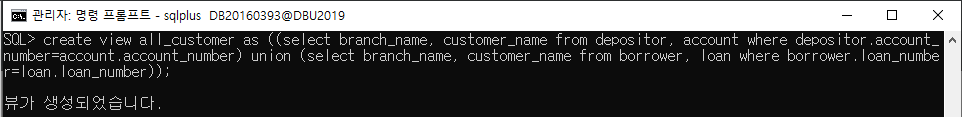
create view b as (select customer\_name, avg(balance) as avg from account, depositor where account.account\_number=depositor.account\_number group by customer\_name);

select a.customer\_name, sum from a, b where a.customer\_name=b.customer\_name and avg = (select min(avg) from b);



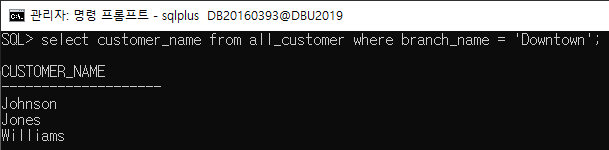
21) 지점 이름과 그 지점에 계좌나 대출 둘 중 하나를 가진 고객 이름으로 구성된 View를 작성하라. (단 View의 이름은 all\_customer이다.)

create view all\_customer as ((select branch\_name, customer\_name from depositor, account where depositor.account\_number=account.account\_number) union (select branch\_name, customer\_name from borrower, loan where borrower.loan\_number=loan.loan\_number));



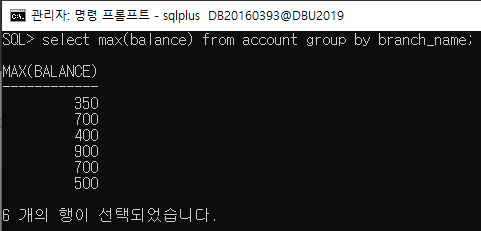
22) 21에서 생성된 View를 이용하여 Downtown 지점의 모든 고객 이름을 나열하라.

select customer\_name from all\_customer where branch\_name = 'Downtown';



23) 각 지점마다 잔고의 최대값을 나열하라.

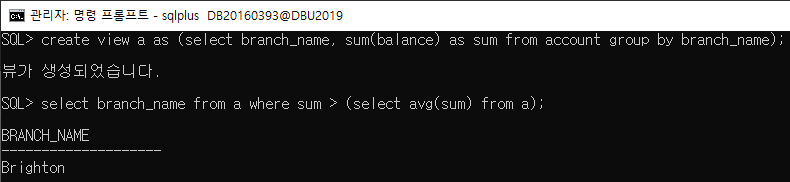
select max(balance) from account group by branch\_name;



24) 모든 지점의 총 계좌 예금의 평균보다 많은 총 계좌 예금을 갖는 모든 지점을 나열하라 (지점에 계좌가 없으면 해당 지점은 평균 계산에서 제외됩니다).

create view a as (select branch\_name, sum(balance) as sum from account group by branch\_name);

select branch\_name from a where sum > (select avg(sum) from a);



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

create view a as select customer\_name, sum(amount) as sum from borrower, loan where borrower.loan\_number=loan.loan\_number group by customer\_name;

select customer\_name, sum from a where sum > (select avg(sum) from a);

