DB assignment1 20201696 안선영

<1> 데이터베이스의 백업파일 : 첨부완료 (20201696_안선영_backup.sql)

<2>캡처 파일에 대한 설명

1. 데이터베이스 생성 (20201696_안선영_1.png)

2. 데이터베이스 스키마 생성 (20201696_안선영_2-1.png ~ 20201696_안선영_2-13.png)

- 직접 설계하는 과정을 보이기 위해 스키마를 생성하며 캡쳐를 하였으나, 조건을 만족시키기 위해 추후 변경한 사항들은 일일히 캡쳐하지 못하였습니다. 따라서 최종 결과물과는 조금 다른 부분이 있습니다.

3. 테이블 레코드(튜플) 생성 (20201696_안선영_3-1.png ~ 20201696_안선영_3-16.png)

- 직접 설계하는 과정을 보이기 위해 스키마를 생성하며 캡쳐를 하였으나, 조건을 만족시키기 위해 추후 변경한 사항들은 일일히 캡쳐하지 못하였습니다. 따라서 최종 결과물과는 조금 다른 부분이 있습니다.

4. 테이블 출력 (20201696_안선영_4-1.png ~ 20201696_안선영_4-8.png)

1). 생성한 테이블 조회한 명령어 (20201696_안선영_4-1.png ~ 20201696_안선영_4-4.png)

```
select * from account;

select * from borrower;

select * from branch;

select * from customer;

select * from customer_banker;

select * from depositor;

select * from employee;

select * from employee_dependent;

select * from loan;

select * from loan_branch;

select * from manager_worker;
```

```
select * from payment;
```

2. 조건에 따른 질의문 입력 테스트 (20201696_안선영_4-5.png ~ 20201696_안선영_4-8.png)

- 생성된 각 테이블에 각각 7개 이상의 적절한 튜플을 삽입한다. 지점 이름(branch_name), 고객 이름(customer_name), 고객 주소(customer_street), 고객 도시(customer_city), 부양 가족 이름(dependent_name), 전화 번호(telephone_number), 직원 이름(employee_name) 칼럼에 해당하는 값은 우리나라 실정에 맞는 실제 자료를 삽입한다. 이 중 고객 주소 칼럼에는 도시 제외한 주소(구, 동, 번지)를 입력하면 된다. //입력완료

- 화폐의 단위는 백 만원이며, 자료 입력 시 백 만원 이하의 금액 단위는 생략한다. 즉, 한 고객이 1,000만원을 예금한 정보는 10으로 입력된다. //입력완료

- 이씨 성을 가진 고객은 세 명만 존재한다. //이주연, 이재현, 이상연

```
select count(customer_id)
from customer
where customer_name Like "이%";
```

- 지점이 위치하고 있는 도시가 아닌 도시에 살고 있는 고객이 두 사람 이상 존재한다. //지창민, 손영재

```
select *
from customer
where customer_city not in (select branch_city from branch);
```

- 한 명의 직원이 두 명 이상의 고객을 동시에 관리하는 자료가 존재한다. //김민지 – 이주연, 이재현

```
select e.employee_id, e.employee_name
from employee e join customer_banker b on e.employee_id = b.employee_id
```

```
group by e.employee_id
having count(b.customer_id)>=2;
```

- 관리하는 고객이 없는 직원이 존재한다. //김민정, 장원영, 다니엘

```
select e.employee_id, e.employee_name
from employee e
where not exists(
   select 1
   from customer_banker b
   where e.employee_id = b.employee_id );
```

- 오직 한 고객만을 관리하는 직원이 존재한다. // 강해린-김영훈

```
select e.employee_id, e.employee_name
from employee e join customer_banker b on e.employee_id = b.employee_id
group by e.employee_id
having count(b.customer_id)=1;
```

- 자신의 담당 직원(자신을 관리하는 직원)이 없는 고객이 존재한다. // 손영재, 지창민

```
select c.customer_id, c.customer_name
from customer c
where not exists(
   select 1
   from customer_banker b
   where c.customer_id = b.customer_id );
```

- 지점 도시(branch_city)의 컬럼 값은 반드시 다섯 개 이상이 존재하여야 한다.

```
select count(*)
from branch;
```

- 지점 도시 '대전'이 반드시 존재하며, 이 도시에는 하나 이상의 지점 이름(branch_name)이 존재하여야 한다.

```
select *
from branch
where branch_city="대전";
```

- 지점 도시 '서울'에는 두 개 이상의 지점 이름이 존재하여야 한다.

```
select *
from branch
where branch_city="서울";
```

- 지점 이름(branch_name)은 반드시 '지점' 접미사로 끝난다. 예를 들어 '명동지점', '숭실대지점'과 같이 입력되어야 한다.

```
select count(branch_name)
from branch
where branch_name not Like "%지점";
```

- 자산(asset)의 값은 천 만원 단위이며 최대값이 50 억원을 넘지 않도록 한다. 또한, 반드시 '서울'에 있는 지점들 중 하나가 전체 지점들 중 최대 자산을 갖도록 한다.

```
select branch_city, sum(assets)
from branch
```

```
group by branch_city;
```

- '서울'이 아닌 곳에 있는 지점들 중에 고객에게 대출해 주지 않은 지점이 하나가 존재하도록 한다.//당진지점

```
select b.branch_name
from branch b left join loan_branch c on b.branch_name = c.branch_name
where b.branch_city != '서울' and b.branch_name not in (select c.branch_name from loan_branch c);
```

- '서울'에 있는 지점들은 각 지점의 대출 금액의 합이 그 지점의 자산을 반드시 넘지 않도록 한다.

```
select b.branch_city, sum(l.amount) as total_loan_amount, sum(b.assets) as total_branch_amount
from loan l left join loan_branch c on(l.loan_number=c.loan_number) left join branch b on (b.branch_name=c.branch_name)
group by b.branch_city
HAVING branch_city='서울';
```

- '서울'이 아닌 곳에 있는 지점들 중에서 대출하여 준 금액의 합이 자산을 넘는 지점이 두 개가 생기도록 한다. //성남지점, 대전지점

```
select b.branch_city, sum(l.amount) as total_loan_amount, sum(b.assets) as total_branch_amount
from loan l left join loan_branch c on(l.loan_number=c.loan_number) left join branch b on (b.branch_name=c.branch_name)
group by b.branch_city
HAVING branch_city !='서울';
```

- 같은 지점에서 세 번 대출한 고객이 반드시 한 명 생기도록 자료를 입력한다. //분당지점 김선우

```
select b.branch_name, count(l.loan_number) as total_count
```

```
from loan l left join loan_branch c on(l.loan_number=c.loan_number) left join branch b on (b.branch_name=c.branch_name)
group by b.branch_name
having count(*) >=3;
```

- 세 군데 지점에서 모두 대출한 고객이 생기도록 자료를 입력한다. // 이주연
- 둘 이상의 고객이 함께 대출한(대출계좌 공유) 경우가 생기도록 자료를 입력한다. //이재현,김영훈
- '서울'에 있는 모든 지점에서 대출한 고객이 생기도록 자료를 입력한다. //이주연
- 자신을 관리하는 직원이 없는 고객은 대출을 하지 못한다. //손영재, 지창민
- 대출하지 않은 고객은 둘 이상 존재한다. //손영재, 지창민
- 상환(payment)이 한번도 이루어지지 않은 대출이 나타나도록 자료를 입력한다. //이주연
- 한 사람이 대출금을 세 번에 걸쳐서 상환한 내용이 나타나도록 자료를 입력한다. //김선우
- 상환이 완료된 대출이 두 건 이상 생기도록 자료를 입력한다. //이재현, 김영훈
- 상환 날짜는 2020년 1월 1일부터 2023년 12월 31일 사이의 날짜를 입력한다. // 입력완료

```
select *
from payment
where not exists (
    select 1
    from payment
    where payment_date between '20200101' and '20231231');
```

- 2023년 11월 4일에 두 건의 대출(두 개의 대출계좌)에 대해 각각의 상환이 일어나도록 자료를 입력한다. //이재현, 김영훈

```
select *
from payment
```

```
where payment_date='20231104';
```

- 하나의 예금계좌(account)에 예금되는 최대 금액은 10억원이다. 최대 금액을 예금한 예금계좌가 반드시 한 개 존재하도록 자료를 입력한다. //이주연

- 대출하지 않은 고객 중에서 예금계좌를 두 개 가지고 있는 고객이 생기도록 자료를 입력한다. //김영훈

- 대출한 고객 중에서 예금계좌를 갖지 않는 고객이 생기도록 자료를 입력한다. //김선우

- 자신을 관리하는 직원이 없는 고객은 반드시 하나의 예금계좌만 가지도록 자료를 입력한다.//이상연, 손영재, 지창민

- 직원(employee)의 전화 번호는 지역번호 없이 3자리 국번이나 4자리 국번으로 된 국번과 4자리 번호를 숫자로 입력한다. 국번과 번호 사이에는 대시('-') 기호로 구분한다. 모든 직원의 국번이 하나의 국번으로 이루어지면 안되며, 서로 다른 국번이 네 개 이상 존재하도록 한다. 또한, 반드시 3자리 국번을 가지는 전화번호와 4자리 국번을 가지는 전화번호가 존재하여야 한다. //입력완료

- 직원의 근무 시작일은 1999년 1월 1일부터 2023년 11월 4일 사이의 날짜를 입력한다. //입력완료

- 근무 기간(employment_length)은 근무 시작일로부터 2023년 11월 4일까지의 근무 기간을 근무년수로 산정하여 정수로 입력하며 소수점 이하는 버린다. 예를 들어, 2023년 1월 1일에 입사한 사람은 근무 기간이 0이다. 또한, 2018년 11월 5일에 입사한 사람은 근무 기간이 5가 된다. //입력완료

- 직원 중에는 관리자(manager)가 없는 직원이 한 사람 존재한다. //김민정

- 직원 두 명을 관리하는 한 사람의 관리자(manager)가 있도록 자료를 입력한다. // 다니엘

- 관리자가 없는 직원은 부양 가족(dependent)이 네 명 존재하도록 자료를 입력한다.

- 두 명 이상 관리하는 관리자(manager)의 경우 부양 가족은 반드시 두 명 이상 존재하도록 자료를 입력한다.

- 부양가족이 없는 직원이 존재하도록 자료를 입력한다. //카리나 김민지 강해린 장원영

- 부양가족이 있는 경우, 부양자의 성과 다른 성을 가진 부양 가족 이름을 반드시 하나 이상 입력한다. //다니엘, 김민정

5. 테이블 및 튜플 작업 전 설계

Customer			
customer_id	customer_name	customer_street	customer_city
1	이주연	마포구 합정동 1번지	서울
2	이재현	연수구 송도동 2번지	인천
3	이상연	유성구 봉명동 3번지	대전
4	김영훈	연수구 동춘동 4번지	인천
5	김선우	해운대구 좌동 5번지	부산
6	지창민	수성구 황금동 6번지	대구
7	손영재	양천구 목동 7번지	서울

Loan	
loan_number	amount
1	1000
2	50
3	100
4	60
5	300
6	100
7	10
8	210

Account	
account_number	balance
1	1000
2	20
3	40
4	30
5	10
6	20
7	10

Employee				
employee_id	employee_name	telephone_number	employee_length	start_date
1	김민지	02-300-1234	2	2021-01-01
2	강해린	02-330-1234	5	2018-09-01
3	다니엘	02-400-1233	10	2013-01-01
4	카리나	032-533-1245	1	2022-01-01
5	김민정	0404-562-1313	3	2020-09-01
6	장원영	02-226-1234	4	2019-09-01

- start date:입사 날짜

Employee_Dependent	
employee_id	dependent_name
3	다길동
3	도지혜
5	김한지
5	김하늘
5	김진주
5	강하나

Branch		
branch_name	branch_city	assets
당진지점	충청	1000
대전지점	대전	200
마포지점	서울	1500
분당지점	성남	500
송도지점	인천	500
화곡지점	서울	3000

Payment			
loan_number(FK Loan)	payment_number	payment_amount	payment_date
5	1	100	2020-10-07
2	2	30	2021-02-03
5	3	210	2021-02-03
5	4	100	2022-11-11
2	5	20	2023-11-04
4	6	50	2023-11-04

- payment_number: 대출 계좌의 대출금을 갚을 때 상환 순서를 기록함
- primary key: (loan_number, payment_number)

Borrower	
customer_id (FK Customer)	loan_number (FK Loan)
1	1
2	2
5	3
4	4
5	5
1	6
5	7
1	8
4	2

Depositor	
customer_id (FK Customer)	account_number (FK Account)
1	1
2	2
4	3
4	4
3	5
6	6
7	7

Manager_Worker	
employee_manager (FK Employee)	employee_worker (FK Employee)
	5
2	1
3	2
3	6
6	4

Loan_Branch	
loan_number (Loan)	branch_name (Branch)
1	마포지점
2	송도지점
3	분당지점
4	송도지점
5	분당지점
6	화곡지점
7	분당지점
8	대전지점

Customer_Banker	
customer_id (FK Customer)	employee_id (FK Employee)
3	
1	1
2	1
4	2
	3
5	4