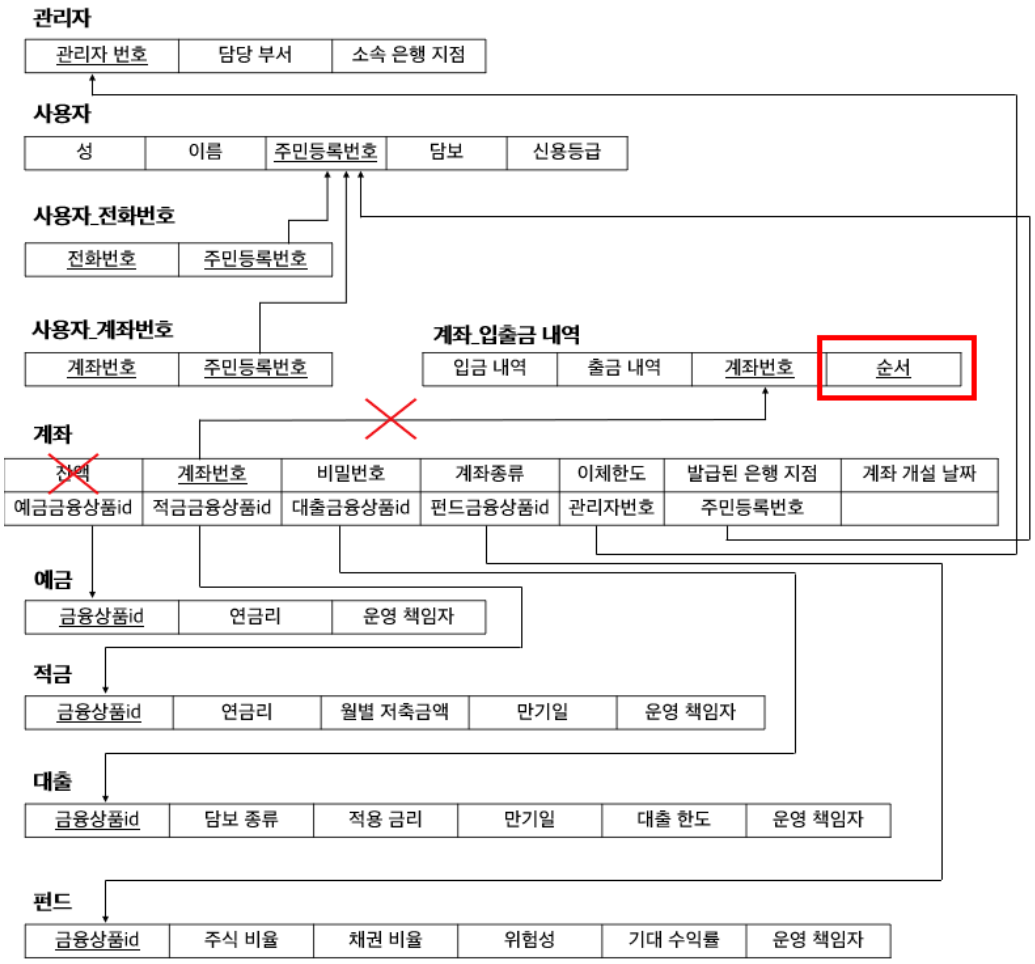
프로젝트 4: DBMS 프로그램 개발

경제금융학부 2019039125 이선민

0. 이전 프로젝트에서 수정사항



1. 계좌_입출금내역 Table에 “순서”라는 Attribute를 추가하고, Primary key를 계좌번호, 순서로 바꿈

→ 다른 시점에서 같은 금액을 입금 혹은 출금할 수 있기 때문

2. 계좌 Table에 “잔액”이라는 Attribute를 삭제함

→ 계좌_입출금내역에서 도출할 수 있기 때문. 후에 코드에 이를 구하는 함수를 구현함

3. 계좌 Table에서 계좌_입출금내역 Table을 reference하는 Foreign key를 해제함

→ 계좌 Table에 입금, 출금 내역을 굳이 알 필요가 없다고 생각, 필요한 잔액 Attribute는 2에서 설명한 것처럼 후에 이를 구하는 함수를 구현함

1. 프로그램 코드에 대한 상세 설명 및 사용되는 SQL문 상세

▪ CreateTable() → 테이블 생성

(1) 관리자 Table

```
'''CREATE TABLE 관리자(
            관리자번호 INT NOT NULL,
            담당부서 VARCHAR(20) NOT NULL,
            소속은행지점 VARCHAR(20) NOT NULL,
        PRIMARY KEY (관리자번호));'''
```

Primary key는 관리자 번호

(2) 사용자 Table

```
'''CREATE TABLE 사용자(
            성 VARCHAR(2) NOT NULL,
            이름 VARCHAR(3) NOT NULL,
            주민등록번호 VARCHAR(30) NOT NULL,
            담보 VARCHAR(50),
            신용등급 INT,
        PRIMARY KEY (주민등록번호));'''
```

Primary key는 주민등록번호

(3) 사용자_전화번호 Table

```
'''CREATE TABLE 사용자_전화번호(
            전화번호 VARCHAR(30) NOT NULL,
            주민등록번호 VARCHAR(30) NOT NULL,
        PRIMARY KEY (전화번호, 주민등록번호),
        FOREIGN KEY (주민등록번호) REFERENCES 사용자(주민등록번호)) ON UPDATE CASCADE;'''
```

Primary key는 전화번호, 주민등록번호
Foreign key는 주민등록번호로, 사용자 Table의 주민등록번호를 참조하며, 참조하는 키가 업데이트 되면, 따라서 업데이트 된다.

(4) 사용자_계좌번호 Table

```
'''CREATE TABLE 사용자_계좌번호(
            계좌번호 VARCHAR(50) NOT NULL,
            주민등록번호 VARCHAR(30) NOT NULL,
        PRIMARY KEY (계좌번호, 주민등록번호),
        FOREIGN KEY (주민등록번호) REFERENCES 사용자(주민등록번호)) ON UPDATE CASCADE;'''
```

Primary key는 계좌번호, 주민등록번호,
Foreign key는 주민등록번호로, 사용자 Table의 주민등록번호를 참조하며, 참조하는 키가 업데이트 되면, 따라서 업데이트 된다.

(5) 예금 Table

```
'''CREATE TABLE 적금(
            금융상품 ID INT NOT NULL,
            연금리 DECIMAL(10,2) NOT NULL,
            월별저축금액 BIGINT NOT NULL,
            만기일 DATE NOT NULL,
            운영책임자 INT NOT NULL,
          PRIMARY KEY (금융상품 ID));'''
```

Primary key는 금융상품ID

(6) 적금 Table

```
'''CREATE TABLE 적금(
            금융상품 ID INT NOT NULL,
            연금리 DECIMAL(10,2) NOT NULL,
            월별저축금액 BIGINT NOT NULL,
            만기일 DATE NOT NULL,
            운영책임자 INT NOT NULL,
          PRIMARY KEY (금융상품 ID));'''
```

Primary key는 금융상품ID

(7) 대출 Table

```
'''CREATE TABLE 대출(
            금융상품 ID INT NOT NULL,
            담보종류 VARCHAR(50) NOT NULL,
            적용금리 DECIMAL(10,2) NOT NULL,
            만기일 DATE NOT NULL,
            대출한도 BIGINT NOT NULL,
            운영책임자 INT NOT NULL,
          PRIMARY KEY (금융상품 ID));'''
```

Primary key는 금융상품ID

(8) 펀드 Table

```
'''CREATE TABLE 펀드(
        금융상품 ID INT NOT NULL,
        주식비율 FLOAT NOT NULL DEFAULT 0,
        채권비율 FLOAT NOT NULL DEFAULT 1,
        위험성 CHAR(1) NOT NULL,
        기대수익률 DECIMAL(10,3)  NOT NULL,
        운영책임자 INT NOT NULL,
    PRIMARY KEY (금융상품 ID)
  );'''
```

Primary key는 금융상품ID

(9) 계좌 Table

```
'''CREATE TABLE 계좌(
        계좌번호 VARCHAR(50) NOT NULL,
        비밀번호 CHAR(4) NOT NULL,
        계좌종류 CHAR(2) NOT NULL,
        이체한도 BIGINT NOT NULL,
        발급은행지점 VARCHAR(20) NOT NULL,
        계좌개설날짜 DATE NOT NULL,
        예금금융상품 ID INT,
        적금금융상품 ID INT,
        대출금융상품 ID INT,
        펀드금융상품 ID INT,
        관리자번호 INT NOT NULL,
        주민등록번호 VARCHAR(30) NOT NULL,
    PRIMARY KEY (계좌번호),
    FOREIGN KEY (예금금융상품 ID) REFERENCES 예금(금융상품 ID),
    FOREIGN KEY (적금금융상품 ID) REFERENCES 적금(금융상품 ID),
    FOREIGN KEY (대출금융상품 ID) REFERENCES 대출(금융상품 ID),
    FOREIGN KEY (펀드금융상품 ID) REFERENCES 펀드(금융상품 ID),
    FOREIGN KEY (관리자번호) REFERENCES 관리자(관리자번호),
    FOREIGN KEY (주민등록번호) REFERENCES 사용자(주민등록번호)) ON UPDATE CASCADE;'''
```

Primary key는 계좌번호

Foreign key에는 예금금융상품ID가 있고, 예금 Table의 금융상품ID를 참조한다.

Foreign key에는 적금금융상품ID가 있고, 적금 Table의 금융상품ID를 참조한다.

Foreign key에는 대출금융상품ID가 있고, 대출 Table의 금융상품ID를 참조한다.

Foreign key에는 펀드금융상품ID가 있고, 펀드 Table의 금융상품ID를 참조한다.

Foreign key에는 관리자번호가 있고, 관리자 Table의 관리자번호를 참조한다.

Foreign key에는 주민등록번호가 있고, 사용자 Table의 주민등록번호를 참조하며, 참조하는 키가 업데이트 되면,

따라서 업데이트 된다.

-만일 test database에 table이 존재하면, CreateTable()을 호출하지 않고, 이미 존재하는 스키마를 이용해 이후 코드를 진행하지만, test database에 table이 존재하지 않으면, CreateTable()을 호출한다.

-이번 과제에서는 DB dump를 받아서 시연하기 때문에 이미 Create 되어있는 Table을 이용하므로, 이 함수를 호출하지 않는다. 하지만, 이 Table의 구조를 설명하고자 이 함수(코드)를 넣었다.

#Manager Authorized (관리자 권한인 함수들)

▪ Register_Manager() → 관리자 정보 추가하기

```
'''INSERT INTO 관리자 VALUES({}, "{}", "{}");'''.format(input1, input2, input3)
```

input1은 입력 받은 관리자번호, input2는 입력 받은 담당부서, input3은 입력 받은 소속은행지점

▪ Delete_Manager(manager_number) → 관리자 정보 삭제하기

```
'''DELETE FROM 관리자 WHERE 관리자번호={};'''.format(manager_number)
```

manager_number은 지우려고 하는 관리자의 관리자번호

▪ Update_Manager_Information(manager_number) → 관리자의 개인 정보를 업데이트함

```
'''UPDATE 관리자 SET 담당부서 = '{}', 소속은행지점 = '{}'
WHERE 관리자번호 = {};'''.format(input1, input2, manager_number)
```

input1은 새로운 담당부서, input2는 새로운 소속은행지점, manager_number은 본인(관리자)의 관리자 번호

▪ Account_Management() → 사용자들의 계좌를 관리하기

(1) 사용자의 계좌 잔액 확인하기

```
'''SELECT  계좌번호, SUM(입금내역)-SUM(출금내역) AS 잔액
  FROM 계좌_입출금내역 GROUP BY 계좌번호 HAVING 잔액 >= {};'''.format(input1)
```

input1로 확인할 금액의 최소 금액을 입력한다. 따라서 이 코드를 실행하면 입력한 금액 이상의 잔액을 가지고 있는 계좌의 잔액이 나온다.

```
cursor.execute(sql)
resultset = cursor.fetchall()
for row in resultset:
    print('계좌번호: "{}" 잔액:{}'.format(row[0], row[1]))
```

(2) 사용자의 계좌 입출금 내역을 수정하기

```
'''UPDATE 계좌_입출금내역 SET 입금내역={}, 출금내역={}
WHERE 계좌번호="{}" AND 순서={}'''.format(input3, input4, input1, input2)
```

input3은 새로운 입금내역 정보, input4은 새로운 출금내역 정보, input1은 수정하려는 계좌번호,
input2는 수정하려는 계좌 입출금 내역의 순서 (몇 번째 입출금 내역인지)

▪ View_Customers() → 모든 사용자의 정보를 확인

(1) 모든 사용자의 정보를 확인하기

```
'''SELECT * FROM 사용자 ORDER BY 성 ASC, 이름 ASC;'''
```

사용자 Table에 있는 모든 attribute의 instance를 보여준다.
성, 이름 순서로 오름차순 되어있는 결과를 보여준다.

(2) 일부 조건에 맞는 사용자 정보를 확인하기
(2-1) 입력한 성을 가진 사용자의 정보 확인하기

```
'''SELECT * FROM 사용자 WHERE 성 = "{}" ORDER BY 성 ASC, 이름 ASC;'''.format(input1)
```

input1은 관리자가 입력한 성이다. 이 코드의 결과로 입력한 성을 가진 사용자의 정보를 보여준다.
이때도 마찬가지로 성, 이름 순서로 오름차순 되어 있는 결과를 보여준다.

(2-2) 입력한 년도에 태어난 사용자의 정보 확인하기

```
'''SELECT * FROM 사용자 WHERE 주민등록번호 LIKE '{}%' ORDER BY 성 ASC, 이름 ASC;'''.format(input2)
```

input2는 확인하고 싶은 사용자의 태어난 년도이다. 이는 주민등록번호의 앞 4자리가 사용자의 태어난 년도임을 이용한 것이다. 이 코드의 결과로 입력한 년도에 태어난 사용자의 정보를 보여준다.
이때도 마찬가지로 성, 이름 순서로 오름차순 되어 있는 결과를 보여준다.

▪ Add_Financial_Products(manager_number) → 금융상품을 추가

여기서 운영책임자는 해당 금융상품을 만든 관리자를 가리킨다.

(1) 예금 금융상품을 추가한 경우

```
'''INSERT INTO 예금(금융상품 id, 연금리, 운영책임자) VALUES ({}, {}, {});'''.format(input1, input2, manager_number)
```

input1은 입력 받은 금융상품ID, input2는 입력 받은 연금리, manager_number은 책임자의 관리자번호이다.

(2) 적금 금융상품을 추가한 경우

```
'''INSERT INTO 적금(금융상품 id, 연금리, 월별저축금액, 만기일, 운영책임자)
VALUES ({}, {}, {}, "{}", {});'''.format(input1, input2, input3, input4, manager_number)
```

input1은 입력 받은 금융상품ID, input2는 입력 받은 연금리, input3은 입력 받은 월별저축금액,
input4는 입력 받은 만기일, manager_number은 책임자의 관리자번호이다.

(3) 대출 금융상품을 추가한 경우

```
'''INSERT INTO 대출(금융상품 id, 담보종류, 적용금리, 만기일, 대출한도, 운영책임자)
VALUES ({}, "{}", {}, "{}", {}, {});'''.format(input1, input2, input3, input4, input5, manager_number)
```

input1은 입력 받은 금융상품ID, input2는 입력 받은 담보종류, input3은 입력 받은 적용금리,
input4는 입력 받은 만기일, input5은 입력 받은 대출한도, manager_number은 책임자의 관리자번호이다.

(4) 펀드 금융상품을 추가한 경우

```
'''INSERT INTO 펀드(금융상품 id, 주식비율, 채권비율, 위험성, 기대수익률, 운영책임자)
VALUES ({}, {}, {}, "{}", {}, {});'''.format(input1, input2, input3, input4, input5, manager_number)
```

input1은 입력 받은 금융상품ID, input2는 입력 받은 주식비율, input3은 유도된 채권비율 (= 1-주식비율)
input4는 유도된 위험성 (주식비율이 0.7초과면 “상”, 주식비율이 0.4초과 0.7이하이면 “중”, 이외는 “하”)
input5는 입력 받은 기대수익률, input6은 책임자의 관리자번호이다.

■ Delete_Financial_Products(manager_number) → 본인(관리자)이 관리하고 있는 금융상품을 삭제

먼저, 자신이 책임지고 있는 금융상품들을 보여줌 (Show_Part(manager_number)함수 호출, 후에 설명)
만일, 해당 금융상품을 사용하고 있는 계좌가 있는 경우, 해당 금융상품을 삭제할 수 없다.
또, 본인이 해당 금융상품의 운영관리자가 아니라면, 해당 금융상품을 삭제할 수 없다.

(1) 예금 상품을 삭제하는 경우

```
'''SELECT * FROM 계좌 WHERE 예금금융상품 ID = {};'''.format(number)
```

해당 금융상품을 사용하는 계좌가 있는지 확인

```
'''DELETE FROM 예금 WHERE 금융상품 ID = {} and 운영책임자 = {};'''.format(number, manager_number)
```

number은 삭제하려는 금융상품ID, manager_number은 본인(관리자)의 관리자번호

(2) 적금 상품을 삭제하는 경우

```
'''SELECT * FROM 계좌 WHERE 적금금융상품 ID = {};'''.format(number)
```

해당 금융상품을 사용하는 계좌가 있는지 확인

```
'''DELETE FROM 적금 WHERE 금융상품 ID = {} and 운영책임자 = {};'''.format(number, manager_number)
```

number은 삭제하려는 금융상품ID, manager_number은 본인(관리자)의 관리자번호

(3) 대출 상품을 삭제하는 경우

```
'''SELECT * FROM 계좌 WHERE 대출금융상품 ID = {};'''.format(number)
```

해당 금융상품을 사용하는 계좌가 있는지 확인

```
'''DELETE FROM 대출 WHERE 금융상품 ID = {} and 운영책임자 = {};'''.format(number, manager_number)
```

number은 삭제하려는 금융상품ID, manager_number은 본인(관리자)의 관리자번호

(4) 펀드 상품을 삭제하는 경우

```
'''SELECT * FROM 계좌 WHERE 펀드금융상품 ID = {};'''.format(number)
```

해당 금융상품을 사용하는 계좌가 있는지 확인

```
'''DELETE FROM 펀드 WHERE 금융상품 ID = {} and 운영책임자 = {};'''.format(number, manager_number)
```

number은 삭제하려는 금융상품ID, manager_number은 본인(관리자)의 관리자번호

■ Show_All() → 모든 금융상품의 정보를 보여줌

금융상품의 정보와 해당 금융상품을 이용하고 있는 계좌의 수를 보여준다.

(1) 예금

```
sql = '''SELECT * FROM 예금;'''
cursor.execute(sql)
resultset = cursor.fetchall()
for row in resultset:
    sql0 = '''SELECT COUNt(*) FROM 계좌 WHERE 예금금융상품 ID = {}'''.format(row[0])
     cursor.execute(sql0)
     result = cursor.fetchone()[0]
     print('금융상품 ID: {0} | 연금리: {1}% | 운영책임자: {2} | 해당 금융상품을 이용하는 계좌의 수: {3}'.format(row[0],
row[1]*100, row[2], result))
```

(2) 적금

```
sql1 = '''SELECT * FROM 적금'''
cursor.execute(sql1)
resultset = cursor.fetchall()
for row in resultset:
    sql0 = '''SELECT COUNt(*) FROM 계좌 WHERE 적금금융상품 ID = {}'''.format(row[0])
    cursor.execute(sql0)
     result = cursor.fetchone()[0]
     print('금융상품 ID: {0} | 연금리: {1}% | 월별 저축금액: {2} | 만기일: {3} | 운영책임자: {4} | 해당 금융상품을 이용하는
계좌의 수: {5}'.format(row[0], row[1]*100, row[2], row[3], row[4], result))
```

(3) 대출

```
sql2 = '''SELECT * FROM 대출'''
cursor.execute(sql2)
resultset = cursor.fetchall()
for row in resultset:
    sql0 = '''SELECT COUNt(*) FROM 계좌 WHERE 대출금융상품 ID = {}'''.format(row[0])
     cursor.execute(sql0)
     result = cursor.fetchone()[0]
     print('금융상품 ID: {0} | 담보 종류: {1} | 적용 금리: {2}% | 만기일: {3} | 대출 한도: {4} | 운영책임자: {5} | 해당
금융상품을 이용하는 계좌의 수: {6}'.format(row[0], row[1], row[2]*100, row[3], row[4], row[5], result))
```

(4) 펀드

```
sql3 = '''SELECT * FROM 펀드'''
cursor.execute(sql3)
 resultset = cursor.fetchall()
for row in resultset:
    sql0 = '''SELECT COUNt(*) FROM 계좌 WHERE 펀드금융상품 ID = {}'''.format(row[0])
     cursor.execute(sql0)
     result = cursor.fetchone()[0]
     print('금융상품 ID: {0} | 주식:채권비율: {1}:{2} | 위험성: {3} | 기대수익률: {4}% | 운영책임자: {5} | 해당 금융상품을
이용하는 계좌의 수: {6}'.format(row[0], row[1], row[2], row[3], row[4]*100, row[5], result))
```

■Show_Part(manager_number) → 본인(관리자)가 책임지고 있는 금융상품을 보여줌

여기서 manager_number은 본인(관리자)의 관리자 번호이다.
본인이 관리하지 않는 금융상품 종류에는 “No 금융상품정보” 라 print한다. (Ex: No Deposit)
print하는 방식은 위에 설명한 Show_All() 함수와 동일하므로 생략한다.

(1) 예금

```
'''SELECT * FROM 예금 WHERE 운영책임자 = {};'''.format(manager_number)
```

(2) 적금

```
'''SELECT * FROM 적금 WHERE 운영책임자 = {};'''.format(manager_number)
```

(3) 대출

```
'''SELECT * FROM 대출 WHERE 운영책임자 = {};'''.format(manager_number)
```

(4) 펀드

```
'''SELECT * FROM 펀드 WHERE 운영책임자 = {};'''.format(manager_number)
```

■Show_Financial_Products(manager_number) → 금융상품을 보여주는 함수

(1) 모든 금융자산을 보여주는 경우 → Show_All() 함수 호출

(2) 자신이 책임지고 있는 금융상품을 보여주는 경우 → Show_Part(manager_number) 함수 호출

■Update_Financial_Product(manager_number) → 본인(관리자)이 관리하고 있는 금융상품의 정보를 업데이트함

먼저, 자신이 책임지고 있는 금융상품들을 보여줌 (Show_Part(manager_number)함수 호출, 위에 설명)
관리자는 본인이 책임지고 있는 금융상품만을 업데이트 할 수 있음

(1) 예금상품을 업데이트 할 경우

```
'''UPDATE 예금 SET 연금리 = {} WHERE 금융상품 ID = {} and 운영책임자 = {};'''.format(input1, number,
manager_number)
```

input1은 새로운 연금리, number은 업데이트 할 금융상품ID, manager_number은 본인(관리자)의 관리자번호

(2) 적금 상품을 업데이트 할 경우

```
'''UPDATE 적금 SET 연금리 = {}, 월별저축금액 = {}, 만기일 = "{}"
 WHERE 금융상품 ID = {} and 운영책임자 = {};'''.format(input1, input2, input3, number, manager_number)
```

input1은 새로운 연금리, input2는 새로운 월별저축금액, input3은 새로운 만기일,
number은 업데이트 할 금융상품ID, manager_number은 본인(관리자)의 관리자번호

(3) 대출 상품을 업데이트 할 경우

```
'''UPDATE 대출 SET 담보종류 = "{}", 적용금리 = {}, 만기일 = "{}", 대출한도 = {}
WHERE 금융상품 ID = {} and 운영책임자 = {};'''.format(input1, input2, input3, input4, number, manager_number)
```

input1은 새로운 담보종류, input2는 새로운 적용금리, input3은 새로운 만기일, input4은 새로운 대출한도,
number은 업데이트 할 금융상품ID, manager_number은 본인(관리자)의 관리자번호

(4) 펀드 상품을 업데이트 할 경우

```
'''UPDATE 펀드 SET 주식비율 = {}, 채권비율 = {}, 위험성 = "{}", 기대수익률 = {}
WHERE 금융상품 ID = {} and 운영책임자 = {};'''.format(input1, input2, input3, input4, number, manager_number)
```

input1은 새로운 주식비율, input2는 새로운 채권비율, input3은 새로운 위험성, input4은 새로운 기대수익률,
number은 업데이트 할 금융상품ID, manager_number은 본인(관리자)의 관리자번호
금융상품을 추가할 때와 마찬가지로, 채권비율, 위험성은 주식비율로부터 그 값이 유도된다.

#Customer Authorized (사용자 권한인 함수들)

▪Add_New_Customers() → 새로운 사용자 등록

```
'''INSERT INTO 사용자(성,이름, 주민등록번호, 담보, 신용등급)
VALUES ("{}","{}", "{}", "{}", {});'''.format(input1, input2, input3, input4, input5)
'''INSERT INTO 사용자_전화번호(전화번호, 주민등록번호)
VALUES ("{}","{}");'''.format(input6, input3)
```

input1은 성, input2는 이름, input3은 주민등록번호, input4는 담보, input5는 신용등급, input6은 전화번호

사용자와 사용자_전화번호 두 개의 Table에 데이터를 입력한다.

```
while(1):
        Select = input("전화번호를 더 추가하시겠습니까? [y or n]: ")
        if(Select == 'y'):
            input7 = input("전화번호: ")
            sql2='''INSERT INTO 사용자_전화번호(전화번호, 주민등록번호)
            VALUES ("{}","{}");'''.format(input7, input3)
            cursor.execute(sql2)
            connection.commit()
        else:
            return
```

만일 전화번호를 한 개 이상 등록하고 싶으면, 전화번호를 추가적으로 더 입력하면 된다. 여기서 input7은 추가적으로 입력한 전화번호

▪Check(password, account) → 계좌번호가 있는 계좌번호인지, 비밀번호가 맞는지 확인하는 함수

```
sql = '''SELECT 비밀번호 FROM 계좌 WHERE 계좌번호 = '{}';'''.format(account)
```

account 는 계좌번호이다.

만일 cursor.execute(sql)이 0 이라면 (해당계좌번호의 계좌가 존재하지 않을 때) print("Wrong account number!)

만일 이렇게 데이터베이스에서 구한 비밀번호가 입력한 비밀번호와 다른 경우 print("Wrong password!")

해당 계좌번호를 가지고 있는 계좌가 존재하며, 사용자가 입력한 비밀번호와 설정한 비밀번호가 같으면 1 을 반환하고, 그게 아니라면 0 을 반환한다.

■Deposit_Withdrawal(account) → 입금 혹은 출금을 할 경우

사용자는 설정한 이체한도 내에서 출금을 할 수 있다.
Check(password, account)가 1 을 반환할 때 (해당 계좌가 존재하며, 비밀번호가 맞음)만 이 함수를 호출할 수 있다.

```
'''SELECT * FROM 계좌_입출금내역 WHERE 계좌번호 = '{}';'''.format(account)
```

계좌_입출금 내역에는 해당 계좌에서 몇 번의 입출금이 있었는지를 기록하는 순서 attribute 가 있다.
위는 그 순서 attribute 를 구하는 쿼리문이며, 결과값을 time 이라는 변수에 저장한다.

(1) 입금

```
'''INSERT INTO 계좌_입출금내역(입금내역, 계좌번호, 순서) VALUES({},'{}',{});'''.format(money, account, time)
```

money 는 입금할 금액, account 는 계좌번호, time 은 위에 구한 순서이다.

(2) 출금

```
"SELECT 이체한도 FROM 계좌 WHERE 계좌번호 = '{}';".format(account)
```

위는 해당 계좌의 이체한도를 계좌 Table 에서 구하는 쿼리문이며, 결과값을 limit 이라는 변수에 저장한다.

```
'''INSERT INTO 계좌_입출금내역(출금내역, 계좌번호, 순서) VALUES({},'{}',{});'''.format(money, account, time)
```

money 는 출금할 금액, account 는 계좌번호, time 은 위에 구한 순서이다.
if 문에 의해 출금할 금액(money)가 이체한도(limit)보다 작을 때 출금을 진행할 수 있다.

■Check_Balance(account) → 계좌의 잔액을 확인하는 함수

Check(password, account)가 1 을 반환할 때 (해당 계좌가 존재하며, 비밀번호가 맞음)만 이 함수를 호출할 수 있다.

```
"SELECT SUM(입금내역) FROM 계좌_입출금내역 WHERE 계좌번호 = '{}';'''.format(account)
'''SELECT SUM(출금내역) FROM 계좌_입출금내역 WHERE 계좌번호 = '{}';'''.format(account)
```

첫 쿼리문을 이용해 해당 계좌의 총 입금금액을 구할 수 있으며, 두번째 쿼리문을 이용해 해당 계좌의 총 출금금액을 구할 수 있다.

```
print("The balance of current account: {} won".format(Deposit_sum-Withdraw_sum))
```

위와 같이 입금금액에서 출금금액을 빼 계좌의 잔액을 print 한다.

■Check_Account_Info(ID) → 본인(사용자)가 등록한 계좌의 정보를 보여주는 함수

먼저, 본인(사용자)가 보유하고 있는 계좌번호를 보여준다.

```
'''SELECT * FROM 사용자_계좌번호 WHERE 주민등록번호 = {};'''.format(ID)
```

ID 는 사용자의 주민등록번호

이를 토대로 사용자는 확인하고 싶은 계좌번호를 입력하고, 이를 num 이라는 변수에 저장한다.

```
'''SELECT 계좌종류 FROM 계좌 WHERE 계좌번호 = '{}';'''.format(num)
```

num 은 확인하려는 계좌번호, 위 쿼리문을 토대로 해당 계좌의 계좌종류를 알아낸다.

(1) 해당 계좌가 예금계좌인 경우 (예금금융상품에 등록한 경우)

```
sql = '''SELECT 금융상품 ID, 연금리, 운영책임자
FROM 예금, 계좌 WHERE 금융상품 ID = 예금금융상품 ID and 계좌번호 ='{}';'''.format(num)
cursor.execute(sql)
result = cursor.fetchall()
for row in result:
    print("금융상품 ID: {}, 연금리: {}%, 운영책임자: {}".format(row[0], row[1]*100, row[2]))
```

(2) 해당 계좌가 적금계좌인 경우 (적금금융상품에 등록한 경우)

```
sql = '''SELECT 금융상품 ID, 연금리, 월별저축금액, 만기일, 운영책임자
FROM 적금, 계좌 WHERE 금융상품 ID = 적금금융상품 ID and 계좌번호 ='{}';'''.format(num)
cursor.execute(sql)
result = cursor.fetchall()
for row in result:
    print("금융상품 ID: {}, 연금리: {}%, 월별저축금액: {}, 만기일: {}, 운영책임자: {}".format(row[0], row[1]*100, row[2],
row[3], row[4]))
```

(3) 해당 계좌가 대출계좌인 경우 (대출금융상품에 등록한 경우)

```
sql = '''SELECT 금융상품 ID, 담보종류, 적용금리, 만기일, 대출한도, 운영책임자
FROM 대출, 계좌 WHERE 금융상품 ID = 대출금융상품 ID and 계좌번호 ='{}';'''.format(num)
cursor.execute(sql)
result = cursor.fetchall()
for row in result:
   print("금융상품 ID: {}, 담보종류: {}, 적용금리: {}%, 만기일: {}, 대출한도: {}, 운영책임자: {}".format(row[0], row[1],
row[2]*100, row[3], row[4], row[5]))
```

(4) 해당 계좌가 펀드계좌인 경우 (펀드금융상품에 등록한 경우)

```
sql = '''SELECT 금융상품 ID, 주식비율, 채권비율, 위험성, 기대수익률, 운영책임자
FROM 펀드, 계좌 WHERE 금융상품 ID = 펀드금융상품 ID and 계좌번호 ='{}';'''.format(num)
cursor.execute(sql)
result = cursor.fetchall()
for row in result:
  print("금융상품 ID: {}, 비율: {}:{}, 위험성: {}, 기대수익률: {}%, 운영책임자: {}".format(row[0], row[1], row[2], row[3],
row[4]*100, row[5]))
```

■Open_Account(ID) → 새로운 계좌를 개설하는 함수

이미 등록된 사용자가 새로운 계좌를 개설하는 함수이다. 만일 사용자가 사전에 등록되지 않았다면, 이 함수는 진행되지 않는다.
여기서 새로운 계좌를 개설한다는 말은 금융상품을 하나 주문한다는 것과 동일한 말이다. 사용자는 계좌를 개설할 때 반드시 계좌 종류를 선택해야 하며, 이는 금융상품 종류 중에 한 가지이다.

(1) 예금 상품을 주문한 경우

```
'''SELECT * FROM 예금'''
"SELECT 운영책임자 FROM 예금 WHERE 금융상품 ID = {}".format(input8)
```

먼저 현재 존재하는 예금 상품들을 보여준다.
input8 은 사용자가 선택한 금융상품 ID 이다.
예금 Table 로부터 운영책임자를 구하고 이를 input9 변수에 저장한다.

```
'''ALTER TABLE 계좌 DROP FOREIGN KEY 계좌_ibfk_5;''' #관리자번호
'''ALTER TABLE 계좌 ADD CONSTRAINT 계좌_ibfk_5 FOREIGN KEY (관리자번호) REFERENCES 관리자(관리자번호);'''
```

foreign key constrain 때문에 foreign key 인 관리자번호, 주민등록번호, 예금금융상품 ID 를 잠시 drop 했다가 다시 foreign key 설정하는 과정을 거쳐야한다.
위와 같은 과정을 주민등록번호, 예금금융상품 ID 에도 반복 (길어서 생략함)

```
'''INSERT INTO 계좌(계좌번호, 비밀번호, 계좌종류, 이체한도, 발급은행지점, 계좌개설날짜, 예금금융상품 ID, 관리자번호,
주민등록번호)
VALUES ("{}", "{}", "{}", {}, "{}", "{}", {}, {}, "{}");'''.format(input1, input2, input3, input4, input5, input6, input8, input9,
input7)
```

input1 은 계좌번호, input2 는 비밀번호, input3 은 계좌종류, input4 는 이체한도, input5 는 발급은행지점, input6 은 계좌개설날짜, input8 은 선택한 예금금융상품 ID, input9 는 관리자번호, input7 은 주민등록번호이다.

(2) 적금 상품을 주문한 경우

```
'''SELECT * FROM 적금;'''
"SELECT 운영책임자 FROM 적금 WHERE 금융상품 ID = {}".format(input8)
```

먼저 현재 존재하는 적금 상품들을 보여준다.
input8 은 사용자가 선택한 금융상품 ID 이다.
적금 Table 로부터 운영책임자를 구하고 이를 input9 변수에 저장한다.

```
'''ALTER TABLE 계좌 DROP FOREIGN KEY 계좌_ibfk_5;''' #관리자번호
'''ALTER TABLE 계좌 ADD CONSTRAINT 계좌_ibfk_5 FOREIGN KEY (관리자번호) REFERENCES 관리자(관리자번호);'''
```

foreign key constrain 때문에 foreign key 인 관리자번호, 주민등록번호, 적금금융상품 ID 를 잠시 drop 했다가 다시 foreign key 설정하는 과정을 거쳐야한다.
위와 같은 과정을 주민등록번호, 적금금융상품 ID 에도 반복 (길어서 생략함)

```
'''INSERT INTO 계좌(계좌번호, 비밀번호, 계좌종류, 이체한도, 발급은행지점, 계좌개설날짜, 적금금융상품 ID, 관리자번호, 주민등록번호)
VALUES ("{}", "{}", "{}", {}, "{}", "{}", {}, {}, "{}");'''.format(input1, input2, input3, input4, input5, input6, input8, input9, input7)
```

input1 은 계좌번호, input2 는 비밀번호, input3 은 계좌종류, input4 는 이체한도, input5 는 발급은행지점, input6 은 계좌개설날짜, input8 은 선택한 적금금융상품 ID, input9 는 관리자번호, input7 은 주민등록번호이다.

(3)(4) 대출 상품을 주문한 경우, 펀드 상품을 주문한 경우
(2)를 보면 알겠지만, (1)과 동일한 과정을 거치면 된다. 코드가 매우 유사하므로 대출 상품과 펀드 상품을 주문한 경우의 코드 설명은 생략하도록 하겠다.

```
'''INSERT INTO 사용자_계좌번호 VALUES("{}","{}");'''.format(input1,input7)
'''INSERT INTO 계좌_입출금내역(계좌번호) VALUES("{}");'''.format(input1)
```

이렇게 만든 계좌와 사용자의 주민등록번호를 사용자_계좌번호 Table 에 입력한다.
또, 해당 계좌번호를 계좌_입출금내역 Table 에 입력한다. 계좌_입출금내역 Table 은 입금내역, 출금내역, 순서의 default 가 0 으로 설정되어 있으므로 계좌번호만 입력하면 된다.

■Delete_Account(account) → 계좌를 해제하는 경우

Check(password, account)가 1 을 반환할 때 (해당 계좌가 존재하며, 비밀번호가 맞음)만 이 함수를 호출할 수 있다.

```
'''DELETE FROM 사용자_계좌번호 WHERE 계좌번호 = '{}';'''.format(account)
'''DELETE FROM 계좌_입출금내역 WHERE 계좌번호 = '{}';'''.format(account)
'''DELETE FROM 계좌 WHERE 계좌번호 = '{}';'''.format(account)
```

차례대로 사용자_계좌번호 Table, 계좌_입출금내역 Table, 계좌 Table 에서 해당 instance 를 삭제한다.
여기서 account 는 사용자로부터 입력 받은 계좌번호이다.

■Update_Customers(ID) → 본인(사용자)의 개인정보를 업데이트하는 경우

이미 등록된 사용자만 자신의 정보를 업데이트할 수 있다. 따라서 등록되지 않은 사용자의 주민등록번호를 입력한 경우, 이 함수는 진행되지 않는다.

```
'''UPDATE 사용자 SET 성='{}', 이름='{}', 담보='{}', 신용등급={}
WHERE 주민등록번호='{}';'''.format(input1, input2, input3, input4, ID)
'''UPDATE 사용자_전화번호 SET 전화번호='{}' WHERE 주민등록번호 = '{}' and 전화번호 = '{}';'''.format(input5, ID, input6)
```

input1 은 새로운 성, input2 는 새로운 이름, input3 은 새로운 담보, input4 는 새로운 신용등급, input5 는 새로운 전화번호, input6 은 교체할 전화번호, ID 는 사용자의 주민등록번호이다.

2. 수행되는 기능 스크린샷

-CreateTable()을 실행하지 않은 경우
(과제에서는 DB dump에서 이 파이썬 코드를 실행하는 것이므로, 다음과 같이 코드가 떠야한다.)

```
Table is already exist, Use table that already exists
```

-CreateTable()을 실행한 경우
(이는 DB dump가 없는 것이므로, DB dump가 있는지 다시 확인해야한다.)

```
Create Table
```

1. Main Menu

```
--------Menu--------
1. Manager Menu
2. Customer Menu
3. Exit the program
--------------------
Input:
```

1을 입력한 경우, 관리자 메뉴로 이동,
2을 입력한 경우, 사용자 메뉴로 이동,
3을 입력한 경우, 프로그램 종료함

2. Manager Menu

```
------------[Manager Menu]------------
0. Return to previous menu
1. Register Manager
2. Delete Manager
3. Update Manager Information
4. Account Management
5. View Customer Information
6. Add Financial Product
7. Delete Financial Product
8. Show Financial Products
9. Update Financial Product
10. Exit the program
--------------------------------------
input:
```

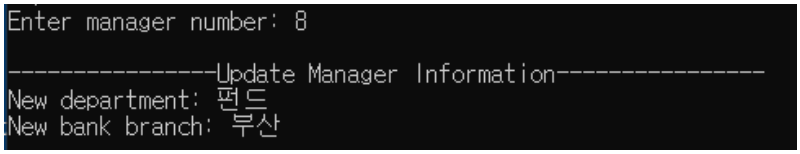
0을 입력한 경우, main menu로 이동
1을 입력한 경우, Register Manager, 관리자를 추가할 수 있음
2를 입력한 경우, Delete Manager, 관리자 정보를 삭제할 수 있음
3을 입력한 경우, Update Manager Information, 관리자의 개인정보를 업데이트할 수 있음
4를 입력한 경우, Account Management, 사용자의 계좌를 관리할 수 있음
5을 입력한 경우, View Customer Information, 모든 고객의 정보를 확인할 수 있음
6을 입력한 경우, Add Financial Product, 새로운 금융상품을 추가할 수 있음
7을 입력한 경우, Delete Financial Product, 금융상품을 삭제할 수 있음

8을 입력한 경우, Show Financial Product, 현재 존재하는 금융상품 정보를 볼 수 있음
9를 입력한 경우, Update Financial Product, 자신이 관리하는 금융상품의 정보를 업데이트할 수 있음
10을 입력한 경우, 프로그램을 종료함

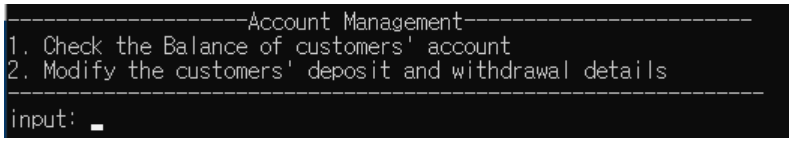
① Register Manager
관리자를 추가하는 function
추가하려는 관리자의 정보, 담당 부서, 소속 은행 지점을 입력하면 된다.

② Delete Manager
관리자를 삭제하는 function
삭제하려는 관리자의 번호를 입력하면 된다.

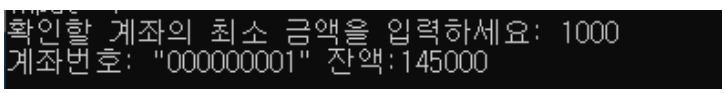
③ Update Manager Information
관리자 정보를 업데이트하는 function
업데이트하고자 하는 관리자의 번호를 입력하고, 변경된 부서와 소속은행지점을 입력하면 된다.



④ Account Management
사용자의 계좌를 관리하는 function



→ 1을 입력(해당 금액 이상의 잔액을 가지고 있는 계좌 확인하는 경우)
확인할 계좌의 최소 금액을 입력하면 된다.

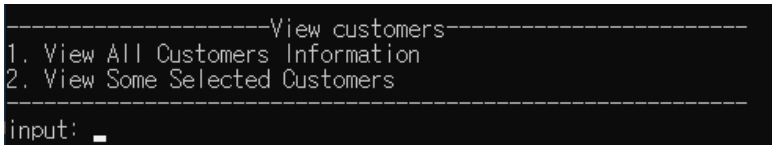


→ 2를 입력(선택한 입출금 내역 수정하는 경우)

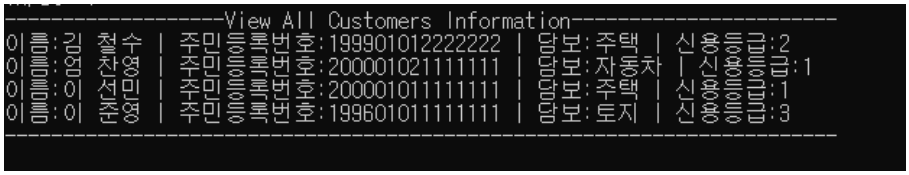
수정하려는 계좌번호, 수정하려는 입출금내역 순서, 바뀐 입금내역, 바뀐 출금내역을 입력하면 된다.

⑤ View Customer Information

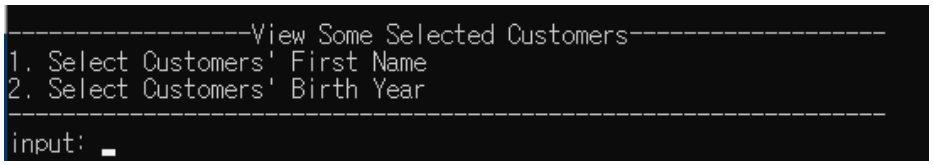
사용자의 정보를 확인하는 function



→ 1을 입력(모든 사용자의 정보를 확인하는 경우)



→ 2을 입력(일부 조건에 맞는 사용자의 정보를 확인하는 경우)



→ 2-1을 입력(입력한 성을 가진 사용자의 정보를 확인하는 경우)

확인하고 싶은 사용자의 성을 입력하면 된다.

→ 2-2을 입력(입력한 년도에 태어난 사용자의 정보를 확인하는 경우)

확인하고 싶은 사용자의 태어난 년도를 입력하면 된다.

⑥ Add Financial Product

금융상품을 추가하는 function

먼저 관리자 번호를 입력하면 다음과 같은 메뉴가 나타난다.

```
Enter manager number: 6

----Add Financial Products----
1. Deposit
2. Installment Saving
3. Loans
4. Funds
------------------------------
input:
```

→ 1을 입력 (예금상품을 만드는 경우)

금융상품ID와 연금리를 입력하면 된다.

```
금융상품ID (1로 시작하는 네자리수): 1004
연금리: 0.02
```

→ 2을 입력(적금상품을 만드는 경우)

금융상품ID, 연금리, 월별저축금액, 만기일을 입력하면 된다.

```
금융상품ID (2로 시작하는 네자리수): 2003
연금리: 0.1
월별저축금액: 10000000
만기일 (EX:2022-01-01): 2025-01-03
```

→ 3을 입력(대출상품을 만드는 경우)

금융상품ID, 담보종류, 적용 금리, 만기일, 대출 한도를 입력하면 된다.

```
금융상품ID (3로 시작하는 네자리수): 3004
담보 종류: 토지
적용 금리: 0.04
만기일 (EX:2022-01-01): 2025-01-02
대출한도: 40000000
```

→ 4를 입력(펀드상품을 만드는 경우)

금융상품ID, 주식비율, 기대수익률을 입력하면 된다.

```
금융상품ID (4로 시작하는 네자리수): 4003
주식 비율: 0.9
기대 수익률: 0.09
```

⑦ Delete Financial Product

금융상품을 삭제하는 function

먼저 관리자번호를 입력하면, 본인이 책임지고 있는 금융상품들의 목록이 뜬다.

```
Enter manager number: 6

----------Financial Products that you are responsibility for----------
§ 1. Deposit §
금융상품ID: 1003 | 연금리: 1.00% | 운영책임자: 6
금융상품ID: 1004 | 연금리: 2.00% | 운영책임자: 6

§ 2. Installment Savings §
No Installment Saving

§ 3. Loans §
No Loan

§ 4. Funds §
금융상품ID: 4003 | 주식:채권비율: 0.9:0.1 | 위험성: 상 | 기대수익률: 9.0% | 운영책임자: 6
------------------------------------------------------------------------------------------------------------------------
```

삭제하고자 하는 금융상품의 종류와 금융상품ID를 입력하면 된다.

```
----------Delete Financial Product----------
Types of financial products to be deleted: 펀드
Enter the number of financial product: 4003
```

```
----------Financial Products that you are responsibility for----------
§ 1. Deposit §
금융상품ID: 1003 | 연금리: 1.00% | 운영책임자: 6
금융상품ID: 1004 | 연금리: 2.00% | 운영책임자: 6

§ 2. Installment Savings §
No Installment Saving

§ 3. Loans §
No Loan

§ 4. Funds §
No Fund
------------------------------------------------------------------------------------------------------------------------
```

지워진 것을 확인할 수 있다.

cf. 만일 1번 관리자가 본인이 책임지고 있지 않은 금융상품(예금, 1004)을 삭제하려고 한다면?

```
Enter manager number: 1

----------Financial Products that you are responsibility for----------
§ 1. Deposit §
금융상품ID: 1000 | 연금리: 1.00% | 운영책임자: 1

§ 2. Installment Savings §
금융상품ID: 2000 | 연금리: 2.00% | 월별 저축금액: 100000 | 만기일: 2022-01-03 | 운영책임자: 1

§ 3. Loans §
No Loan

§ 4. Funds §
금융상품ID: 4000 | 주식:채권비율: 0.5:0.5 | 위험성: 중 | 기대수익률: 7.000% | 운영책임자: 1
------------------------------------------------------------------------------------------------------------------------

----------Delete Financial Product----------
Types of financial products to be deleted: 예금
Enter the number of financial product: 1004
You don't have an authority to delete this product
```

“You don’t have an authority to delete this product”이라고 뜨면서

```
----------All Financial Products----------
§ 1. Deposit §
금융상품ID: 1000 | 연금리: 1.00% | 운영책임자: 1
금융상품ID: 1001 | 연금리: 3.00% | 운영책임자: 2
금융상품ID: 1002 | 연금리: 2.00% | 운영책임자: 3
금융상품ID: 1003 | 연금리: 1.00% | 운영책임자: 6
금융상품ID: 1004 | 연금리: 2.00% | 운영책임자: 6
```

1004번 금융상품이 지워지지 않은 것을 볼 수 있다.

cf. 만일 해당 금융상품(1000)을 이용하고 있는 계좌가 존재하는데, 금융상품을 삭제하려고 한다면?

```
-------------------------All Financial Products-------------------------------
§ 1. Deposit §
금융상품ID: 1000 | 연금리: 1.00% | 운영책임자: 1 | 해당 금융상품을 이용하는 계좌의 수: 2
금융상품ID: 1001 | 연금리: 3.00% | 운영책임자: 2 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 1002 | 연금리: 2.00% | 운영책임자: 3 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 1003 | 연금리: 1.00% | 운영책임자: 6 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 1004 | 연금리: 2.00% | 운영책임자: 6 | 해당 금융상품을 이용하는 계좌의 수: 0
```

1번 관리자가 1000번 금융상품을 삭제하고자 시도하면,

```
--------------------Delete Financial Product--------------------
Types of financial products to be deleted: 예금
Enter the number of financial product: 1000
There's an account that uses the product
```

“There’s an account that uses the product”이라고 뜨면서

```
-------------------------All Financial Products-------------------------------
§ 1. Deposit §
금융상품ID: 1000 | 연금리: 1.00% | 운영책임자: 1 | 해당 금융상품을 이용하는 계좌의 수: 2
금융상품ID: 1001 | 연금리: 3.00% | 운영책임자: 2 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 1002 | 연금리: 2.00% | 운영책임자: 3 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 1003 | 연금리: 1.00% | 운영책임자: 6 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 1004 | 연금리: 2.00% | 운영책임자: 6 | 해당 금융상품을 이용하는 계좌의 수: 0
```

지워지지 않은 것을 확인할 수 있다.

⑧ Show Financial Product

금융 상품들을 확인하는 function

먼저 관리자번호를 입력한다.

```
Enter manager number: 1

------------------Show Financial Products------------------
1. All Financial Products
2. Financial Products that you are responsibility for
-----------------------------------------------------------
input:
```

→ 1을 입력(모든 금융상품의 정보를 보여주는 경우)

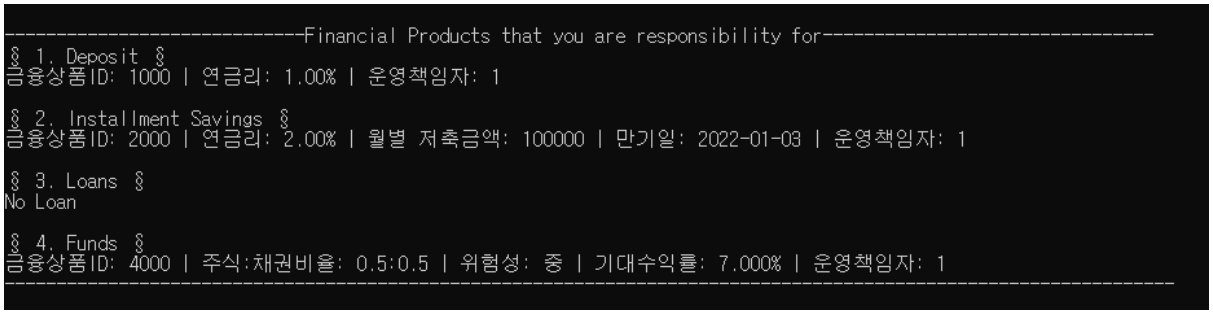
```
-------------------------All Financial Products-------------------------------
§ 1. Deposit §
금융상품ID: 1000 | 연금리: 1.00% | 운영책임자: 1 | 해당 금융상품을 이용하는 계좌의 수: 2
금융상품ID: 1001 | 연금리: 3.00% | 운영책임자: 2 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 1002 | 연금리: 30.00% | 운영책임자: 3 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 1003 | 연금리: 1.00% | 운영책임자: 6 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 1004 | 연금리: 2.00% | 운영책임자: 6 | 해당 금융상품을 이용하는 계좌의 수: 0

§ 2. Installment Savings §
금융상품ID: 2000 | 연금리: 2.00% | 월별 저축금액: 100000 | 만기일: 2022-01-03 | 운영책임자: 1 | 해당 금융상품을 이용하는 계좌의 수: 2
금융상품ID: 2001 | 연금리: 3.00% | 월별 저축금액: 200000 | 만기일: 2021-12-30 | 운영책임자: 2 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 2002 | 연금리: 1.00% | 월별 저축금액: 10000 | 만기일: 2023-01-01 | 운영책임자: 4 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 2003 | 연금리: 10.00% | 월별 저축금액: 10000000 | 만기일: 2025-01-03 | 운영책임자: 7 | 해당 금융상품을 이용하는 계좌의 수: 1

§ 3. Loans §
금융상품ID: 3000 | 담보 종류: 주택 | 적용 금리: 1.00% | 만기일: 2025-09-05 | 대출 한도: 40000000 | 운영책임자: 2 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 3001 | 담보 종류: 자동차 | 적용 금리: 3.00% | 만기일: 2024-05-06 | 대출 한도: 5000000 | 운영책임자: 3 | 해당 금융상품을 이용하는 계좌의 수: 3
금융상품ID: 3002 | 담보 종류: 주택 | 적용 금리: 2.00% | 만기일: 2023-01-01 | 대출 한도: 1000000 | 운영책임자: 5 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 3003 | 담보 종류: 주택 | 적용 금리: 5.00% | 만기일: 2030-04-01 | 대출 한도: 3000000 | 운영책임자: 5 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 3004 | 담보 종류: 토지 | 적용 금리: 4.00% | 만기일: 2025-01-02 | 대출 한도: 40000000 | 운영책임자: 4 | 해당 금융상품을 이용하는 계좌의 수: 0

§ 4. Funds §
금융상품ID: 4000 | 주식:채권비율: 0.5:0.5 | 위험성: 중 | 기대수익률: 7.000% | 운영책임자: 1 | 해당 금융상품을 이용하는 계좌의 수: 1
금융상품ID: 4001 | 주식:채권비율: 0.1:0.9 | 위험성: 하 | 기대수익률: 1.000% | 운영책임자: 3 | 해당 금융상품을 이용하는 계좌의 수: 2
금융상품ID: 4002 | 주식:채권비율: 0.4:0.6 | 위험성: 하 | 기대수익률: 2.000% | 운영책임자: 5 | 해당 금융상품을 이용하는 계좌의 수: 1
-----------------------------------------------------------------------------------------------------
```

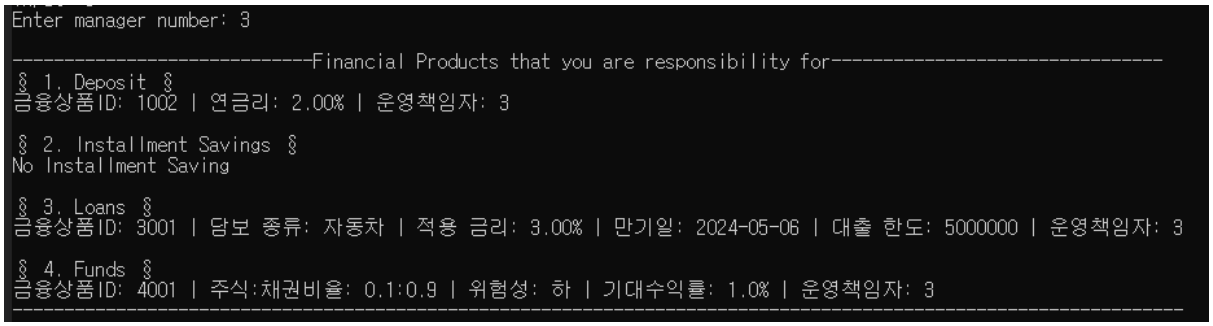
→ 2를 입력(해당 관리자번호, (예시는 1)가 책임지고 있는 금융상품 정보를 보여주는 경우)



⑨ Update Financial Product

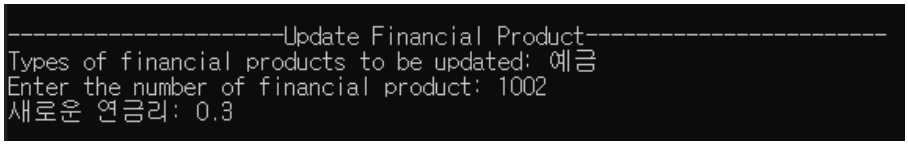
금융상품 정보를 업데이트하는 function

먼저 관리자번호를 입력한다. 그러면 본인이 책임지고 있는 금융상품 목록을 보여준다.

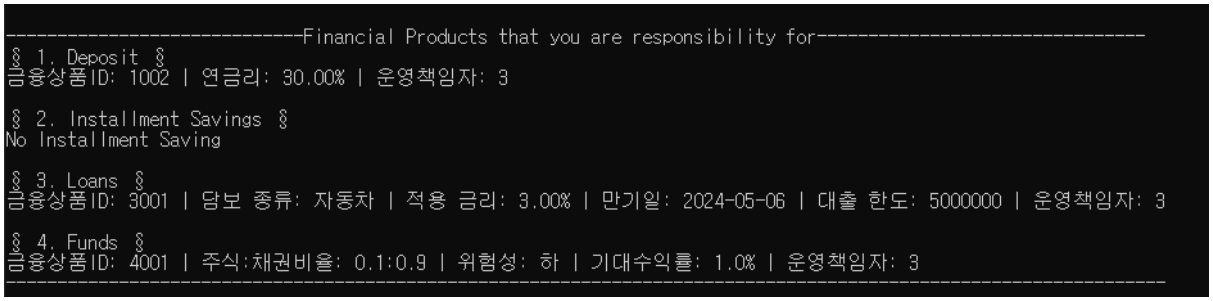


변경하고자 하는 금융상품의 종류와 금융상품의 번호를 입력하고

각각의 금융상품 종류에 해당하는 변경 값을 입력해주면 된다.



(해당 예시는 예금 금융상품 중 하나인 1002번 상품의 정보를 업데이트함)



1002번 금융상품의 정보가 바르게 업데이트된 것을 확인할 수 있다.

삭제와 마찬가지로, 본인이 책임지고 있는 금융상품만 업데이트할 수 있다.

3. Customer Menu

```
---------------[Customer Menu]-------------------
0. Return to previous menu
1. Deposit and Withdraw money
2. Check your account balance
3. Check registered financial product information
4. Register your information
5. Open your account (Order financial products)
6. Delete your account
7. Update Customer Information
8. Exit the program
-------------------------------------------------
input:
```

0을 입력한 경우, main menu로 이동

1을 입력한 경우, Deposit and Withdraw money, 계좌에 돈을 입출금

2를 입력한 경우, Check your account money, 계좌의 잔액을 확인

3을 입력한 경우, Check your registered financial product information,

해당 계좌가 가입되어 있는 금융상품의 정보를 확인

4를 입력한 경우, Register your information, 사용자 등록

5를 입력한 경우, Open your account, 계좌 만들기, 즉 금융상품 주문하기

6을 입력한 경우, Delete your account, 계좌 삭제하기

7을 입력한 경우, Update Customer information, 사용자 정보 변경하기

8을 입력한 경우, Exit the program, 프로그램 종료하기

① Deposit and Withdraw money

사용자 계좌에 돈을 입출금하는 function

먼저 입출금하려는 계좌번호와 비밀번호를 입력한다.

```
Please enter your account: 000000002
Please enter your account password: 0000
```

```
------------------[Choose Deposit or Withdraw]----------------------
1. Deposit
2. Withdraw
input: 1
How much? 300000
```

→ 1을 입력(입금하는 경우), 얼마를 입금할지 입력하면 된다.

```
------------------[Choose Deposit or Withdraw]----------------------
1. Deposit
2. Withdraw
input: 2
How much? 10000
```

→ 2를 입력(출금하는 경우), 얼마를 출금할지 입력하면 된다.

cf. 계좌의 비밀번호가 틀린 경우 (000000002 계좌의 비밀번호는 0000이다)

```
Please enter your account: 000000002
Please enter your account password: 1111
Wrong password
```

“Wrong password”라고 뜨면서 입출금이 일어나지 않는다.

cf. 존재하지 않는 계좌번호를 입력한 경우 (000000021 계좌는 존재하지 않음)

```
Please enter your account: 000000021
Please enter your account password: 0000
Wrong account number!
```

“Wrong account number!”라고 뜨면서 입출금이 일어나지 않는다.

cf. 설정한 이체한도보다 더 큰 돈을 출금하려고 하는 경우

(000000002 계좌의 이체한도는 300000이다.)

```
Please enter your account: 000000002
Please enter your account password: 0000


------------------[Choose Deposit or Withdraw]----------------------
1. Deposit
2. Withdraw
input: 2
How much? 5000000
It exceeded the transfer limit
```

“It exceeded the transfer limit”라고 뜨며 입출금이 일어나지 않는다.

② Check your account money

계좌의 잔액을 확인하는 function

먼저 잔액을 확인하고자 하는 계좌번호와 비밀번호를 입력한다.

```
Please enter your account: 000000002
Please enter your account password: 0000


----------------Check your account balance----------------
The balance of current account: 290000 won
```

위에 입출금 할 때와 마찬가지로 계좌의 비밀번호가 틀리거나 존재하지 않은 계좌번호를 입력한 경우,

이 함수는 진행되지 않는다.

③ Check your registered financial product information,

해당 계좌가 가입되어 있는 금융상품의 정보를 확인하는 function

먼저 본인의 주민등록번호를 입력한다.

```
Please enter your personal ID: 200001011111111
```

본인이 소유하고 있는 계좌번호를 보여주고, 이중에서 확인하고자 하는 계좌의 계좌번호를 입력하면,

그 계좌로 가입된 금융상품의 정보가 나온다.

```
----------------Check registered financial product information----------------
보유하고 있는 계좌번호: 000000001
보유하고 있는 계좌번호: 000000002
보유하고 있는 계좌번호: 000000003
보유하고 있는 계좌번호: 000000004
-----------------------------------------
확인하고 싶은 계좌번호: 000000002
-----------------------------------------
금융상품ID: 1001, 연금리: 3.00%, 운영책임자: 2
```

④ Register your information

사용자를 등록하는 function

등록하고자 하는 성, 이름, 주민등록번호, 담보, 신용등급, 전화번호를 입력하면 된다.

이때, 여러 개의 전화번호를 입력할 수 있다. (y를 입력할 경우, 전화번호 추가 가능)

```
--------Add New Customer---------
성: 김
이름: 영희
주민등록번호: 199502031111111
담보: 토지
신용등급: 4
전화번호: 01099999999
전화번호를 더 추가하시겠습니까? [y or n]: y
전화번호: 01099999998
전화번호를 더 추가하시겠습니까? [y or n]: n
```

⑤ Open your account

계좌 만들기, 즉 금융상품을 주문하는 function

사용자의 주민등록번호를 입력하고, 만들려는 계좌번호, 그 계좌의 비밀번호, 계좌종류, 이체한도,

발급된 은행지점, 계좌 개설 날짜를 입력한다.

그러면 해당 계좌 종류의 금융상품 목록이 나오고, 이중에서 금융상품을 고르면 된다

```
Please enter your personal ID: 199502031111111

----------Open Account-----------
계좌 번호: 000000021
비밀번호 (4자리수): 0000
계좌 종류: 펀드
이체 한도: 4000000
발급된 은행 지점: 경주
계좌 개설 날짜 (EX: 2021-11-27): 2021-12-01
금융상품ID: 4000 | 주식:채권비율: 0.5:0.5 | 위험성: 중 | 기대수익률: 7.000% | 운영책임자: 1
금융상품ID: 4001 | 주식:채권비율: 0.1:0.9 | 위험성: 하 | 기대수익률: 1.000% | 운영책임자: 3
금융상품ID: 4002 | 주식:채권비율: 0.4:0.6 | 위험성: 하 | 기대수익률: 2.000% | 운영책임자: 5
선택한 금융상품ID: 4000
```

위의 예시는 펀드 계좌를 만들 경우이다. 예금, 적금, 대출 금융상품도 마찬가지로 실행된다.

cf. 만일 등록되지 않은 사용자가 계좌를 개설하려고 하는 경우

"You are not registered. Please register your personal information first"라고 뜨며, 계좌 개설이 되지 않는다.

⑥ Delete your account

계좌를 삭제하는 function

삭제하려는 계좌의 계좌번호와 비밀번호를 입력하면 된다

위에 나왔던 function처럼 계좌의 비밀번호가 틀렸거나, 존재하지 않는 계좌번호를 입력하면, 계좌 삭제가 진행되지 않는다.

⑦ Update Customer information

사용자 정보를 변경하는 function

변경할 성, 이름, 담보, 신용등급, 전화번호를 입력한다.

여기서, 전화번호는 여러 개 저장할 수 있기 때문에 삭제할 전화번호도 입력해 주어야한다.