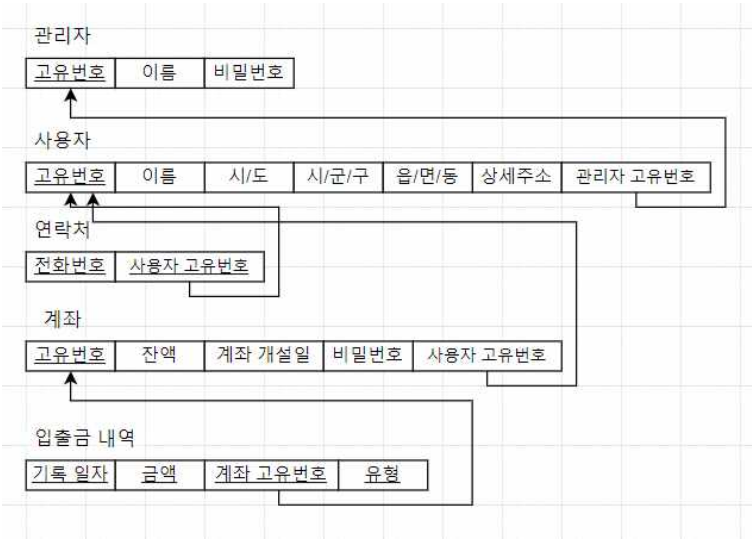
프로젝트 4: DBMS 프로그램 개발

2018067233 윤동빈

0. relational schema 변경 사항



변경된 relational schema

- 관리자 table '이름', '비밀번호' 추가 : 관리자에 대한 정보가 부족한 것 같아 추가

- 사용자 table '이름' 추가 : 사용자에 대한 정보가 부족한 것 같아 추가
- 사용자 table '신용등급' 삭제 : 원래는 잔액으로 설정하려 했으나 대출 시스템에 대한 정보가 불충분하고 이해도가 떨어져서 삭제
- 사용자 table '계좌 개수' 삭제 : pymysql에서 cursor.rowcount로 체크가 가능해서 삭제
- 사용자 table 주소 관련 attributes 이름 변경 : 국내 표준 체계로 변경

- 최근 1주일/1개월/전체 입출금 내역 tables를 1개의 입출금 내역 table로 관리
  : 날짜가 지날 때마다 각 table에 속한 tuples가 실시간으로 관리돼야 하는데 어려울 것 같다고 판단돼 1개의 입출금 내역 table로 관리하도록 수정
- 입출금 내역 table에 '입출금 내역'에서 '기록 일자', '금액', '유형'으로 세분화
  '기록 일자' : 내역은 일자를 기준으로 구분돼야 생각해서 추가
  '금액' : 입금액 또는 출금액 정보
  '유형' : 입금/출금에 대한 정보

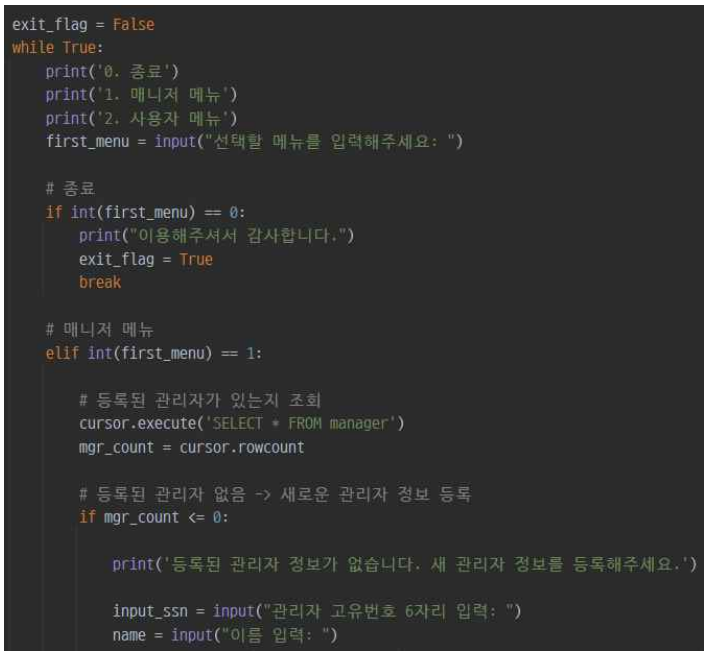
- 계좌 table '비밀번호' 추가 : 이체 시 비밀번호를 입력해야 할 필요가 있다고 느껴 추가

1. 프로그램 코드에 대한 자세한 설명

- 함수
input_date() : 날짜 8자리를 입력받는 함수
input_balance() : 금액을 입력받는 함수

- 함수 외 코드
각 상황마다 메뉴가 제시되고 입력이 들어오면 선택한 메뉴의 설명에 따라 진행되도록 한다.
뒤로가기, 종료를 제외한 메뉴 선택 시 해당 메뉴로 이동하도록 하고, 잘못된 입력(메뉴 선택 또는 로그인 등)이 들어오면 다시 입력을 받도록 한다.

메인에서 제시되는 메뉴는 3가지로 다음과 같다.
매니저 메뉴 이용 시 등록된 매니저 정보가 없다면 새로운 매니저 정보를 생성하도록 한다.
매니저 정보가 존재한다면 로그인 후 진행하도록 한다.
로그인 실패 시 다시 입력받도록 한다.



```
exit_flag = False
while True:
    print('0. 종료')
    print('1. 매니저 메뉴')
    print('2. 사용자 메뉴')
    first_menu = input("선택할 메뉴를 입력해주세요: ")

    # 종료
    if int(first_menu) == 0:
        print("이용해주셔서 감사합니다.")
        exit_flag = True
        break

    # 매니저 메뉴
    elif int(first_menu) == 1:

        # 등록된 관리자가 있는지 조회
        cursor.execute('SELECT * FROM manager')
        mgr_count = cursor.rowcount

        # 등록된 관리자 없음 -> 새로운 관리자 정보 등록
        if mgr_count <= 0:

            print('등록된 관리자 정보가 없습니다. 새 관리자 정보를 등록해주세요.')

            input_ssn = input("관리자 고유번호 6자리 입력: ")
            name = input("이름 입력: ")
```

위에서 첫 번째 메뉴 입력 변수를 *first_menu*로 했듯이 내부 메뉴로 진행될 때마다 순차적으로 *second_menu, third_menu, fourth_menu*의 변수명을 가진다.

각 상황에서 잘못된 입력이 들어올 경우 다시 입력받기 위해 while loop에서 입력이 진행되며 정상적인 입력이 수행됐을 경우 빠져 나올 수 있도록 break문으로 처리돼 있다.

모든 상황에서 공통적으로 작성된 코드 설명을 위해 *계좌 생성* 코드를 예시로 들면,
필요한 정보(customer_ssn)를 입력받고 적절한 SQL문 생성 후 DB로 쿼리를 날린다.
입력된 정보와 관련된 정보가 DB에 존재하지 않는 경우에 대해 예외처리를 위해
cursor.rowcount를 활용한다. (if customer_cnt <= 0)

```
# 계좌 생성
elif int(third_menu) == 1:

    while True:

        # 고유번호 확인
        customer_ssn = input("사용자 고유번호 6자리 입력: ")
        cursor.execute('SELECT * FROM customer WHERE Ssn=' + customer_ssn)
        customer_cnt = cursor.rowcount

        # 일치하는 정보 없음
        if customer_cnt <= 0:
            print('일치하는 사용자 정보가 없습니다. 다시 입력해주세요.')
            continue

        while True:

            # 계좌번호 중복 확인
            account_number = input('등록할 계좌번호 6자리 입력: ')
            cursor.execute('SELECT * FROM account WHERE Number=' + account_number)
            account_cnt = cursor.rowcount

            # 계좌번호 중복
            if account_cnt > 0:
                print('중복되는 계좌번호가 있습니다. 다시 입력해주세요.')
                continue
```

무언가를 삭제하는 상황인 경우, relational schema에서 볼 수 있듯이 table끼리
foreign key constraint가 있어 관련 tuple은 함께 삭제돼야 하는 경우가 있다.
관리자 계정, *사용자 계정*, *계좌*를 삭제하는 상황이 해당되며 다음은 *계좌 삭제* 코드이다.

```
# 계좌번호 중복 확인
account_number = input('삭제할 계좌번호 6자리 입력: ')
cursor.execute('SELECT * FROM account WHERE Number=' + account_number)
account_cnt = cursor.rowcount

# 계좌번호 없음
if account_cnt <= 0:
    print('해당되는 계좌가 없습니다. 다시 입력해주세요.')
    continue

password = input('비밀번호 입력: ')

# 사용자 고유번호 foreign key 설정 해제
cursor.execute('SET foreign_key_checks = 0')
connection.commit()

# 관련 입출금 내역 삭제
cursor.execute('DELETE FROM history WHERE Anum=' + account_number)
connection.commit()

# 계좌 삭제
cursor.execute(
    'DELETE FROM account WHERE Number=' + account_number + ' AND Password=' + password)
connection.commit()

# 사용자 고유번호 foreign key 설정
cursor.execute('SET foreign_key_checks = 1')
connection.commit()

print('계좌 삭제가 성공적으로 완료되었습니다.')
break
```

*사용자 메뉴*에서 *출금*의 경우 잔액보다 많은 금액을 출금할 수 없다.
이에 대해 예외 처리를 하도록 구현된 코드는 다음과 같다.
*본인 계좌 이체, 타인 계좌 이체*에 공통으로 구현된 내용이다.

```
# 출금
elif int(mode) == 0:

    while True:

        from_account = input('출금할 계좌번호 6자리 입력: ')
        cursor.execute(
            'SELECT * FROM account WHERE Number=' + from_account + ' AND Ssn=' + user_ssn)
        from_account_exist = cursor.rowcount
        account = cursor.fetchone()

        # 존재하지 않음
        if from_account_exist <= 0:
            print('존재하지 않는 계좌번호입니다. 다시 입력해주세요.')
            continue

        # 비밀번호 확인
        account_password = input('비밀번호 입력: ')
        if account_password != account[3]:
            print('비밀번호가 일치하지 않습니다. 다시 입력해주세요.')
            continue

        amount = input('출금할 금액 입력: ')
        if int(amount) < 0:
            print('금액은 반드시 0 이상의 숫자여야 합니다. 다시 입력해주세요.')
            continue
        elif int(amount) > int(account[1]):
            print('잔액이 부족합니다. 다시 입력해주세요.')
            print('해당 계좌의 잔액은 ' + str(account[1]) + '원 입니다.')
            continue
```

구현된 메뉴의 **목차**는 다음과 같다.

0. 종료
1. 매니저 메뉴
    0. 종료
    1. 뒤로가기
    2. 신규 사용자 등록
    3. 사용자 계좌 관리
        0. 뒤로가기
        1. 계좌 생성
        2. 계좌 삭제
    4. 사용자 입출금 내역 관리
        0. 뒤로가기
        1. 입출금 내역 조회
        2. 입출금 내역 수정
        3. 입출금 내역 삭제
    5. 사용자 정보 수정
        0. 뒤로가기
        1. 이름
        2. 주소
        3. 관리자 고유번호
        4. 연락처 추가
        5. 연락처 수정
        6. 연락처 삭제
    6. 사용자 정보 삭제
    7. 사용자 정보 조회
    8. 관리자 계정 생성
    9. 관리자 정보 수정
        0. 뒤로가기
        1. 이름
        2. 비밀번호
    10. 관리자 정보 삭제
    11. 관리자 정보 조회
2. 사용자 메뉴
    0. 종료
    1. 뒤로가기
    2. 본인 계좌로 이체
    3. 타인 계좌로 이체

2. 수행되는 기능 스크린샷

메인 화면

```
C:₩Users₩동비니₩PycharmProjects₩DBSTest1>python 2018062733_윤동빈_P4.py
0. 종료
1. 매니저 메뉴
2. 사용자 메뉴
선택할 메뉴를 입력해주세요:
```

[메니저 메뉴]

```
0. 종료
1. 뒤로가기
2. 신규 사용자 등록
3. 사용자 계좌 관리
4. 사용자 입출금 내역 관리
5. 사용자 정보 수정
6. 사용자 정보 삭제
7. 사용자 정보 조회
8. 관리자 계정 생성
9. 관리자 정보 수정
10. 관리자 정보 삭제
11. 관리자 정보 조회
선택할 메뉴를 입력해주세요:
```

[메니저 메뉴] - [신규 사용자 등록]

```
선택할 메뉴를 입력해주세요: 2
사용자 고유번호 6자리 입력: 777777
이름 입력: Kim
주소 시/도 입력: 서울시
주소 시/군/구 입력: 성북구
주소 읍/면/동 입력: 쌍문동
상세주소 입력: waw
연락처 입력 (예) 01012341234 : 01078457845
사용자 등록이 성공적으로 완료되었습니다.
```

[매니저 메뉴] - [사용자 계좌 관리]

```
0. 뒤로가기
1. 계좌 생성
2. 계좌 삭제
```

[매니저 메뉴] - [사용자 계좌 관리] - [계좌 생성]

```
선택할 메뉴를 입력해주세요: 1
사용자 고유번호 6자리 입력: 777777
등록할 계좌번호 6자리 입력: 777777
날짜 입력 (예) 2021년 5월 7일 -> 20210507 : 20210808
비밀번호 입력: 777777
계좌 등록이 성공적으로 완료되었습니다.
```

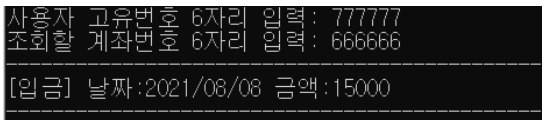
[매니저 메뉴] - [사용자 계좌 관리] - [계좌 삭제]

```
선택할 메뉴를 입력해주세요: 2
사용자 고유번호 6자리 입력: 777777
삭제할 계좌번호 6자리 입력: 777777
비밀번호 입력: 777777
계좌 삭제가 성공적으로 완료되었습니다.
```

[매니저 메뉴] - [사용자 계좌 관리] - [사용자 입출금 내역 관리]

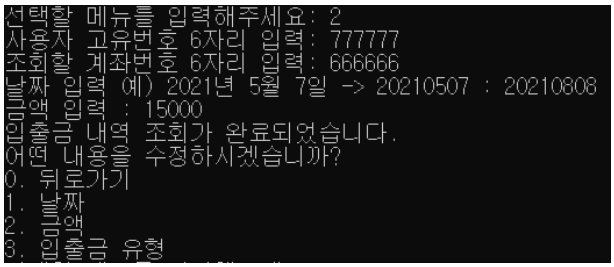
```
0. 뒤로가기
1. 입출금 내역 조회
2. 입출금 내역 수정
3. 입출금 내역 삭제
```

[매니저 메뉴] - [사용자 계좌 관리] - [사용자 입출금 내역 관리] - [입출금 내역 조회]



[매니저 메뉴] - [사용자 계좌 관리] - [사용자 입출금 내역 관리] - [입출금 내역 수정]

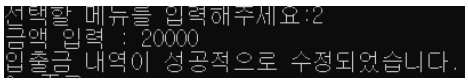
사용자 고유번호, 계좌번호, 날짜, 금액을 입력받아 수정할 입출금 내역을 설정



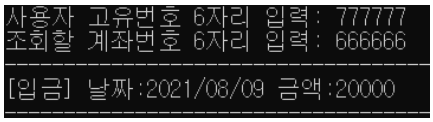
[매니저 메뉴] - [사용자 계좌 관리] - [사용자 입출금 내역 관리] - [입출금 내역 수정] - [날짜]

날짜 수정 결과

[매니저 메뉴] - [사용자 계좌 관리] - [사용자 입출금 내역 관리] - [입출금 내역 수정] - [금액]



금액 수정 결과



[매니저 메뉴] - [사용자 계좌 관리] - [사용자 입출금 내역 관리] - [입출금 내역 수정] - [입출금 유형]

입출금 유형 수정 결과

[매니저 메뉴] - [사용자 계좌 관리] - [사용자 입출금 내역 관리] - [입출금 내역 삭제]

```
선택할 메뉴를 입력해주세요: 3
사용자 고유번호 6자리 입력: 777777
조회할 계좌번호 6자리 입력: 666666
날짜 입력 예) 2021년 5월 7일 -> 20210507 : 20210809
금액 입력 : 20000
입출금 내역이 성공적으로 삭제되었습니다.
```

삭제 후에는 입출금 내역이 없으므로 결과는 다음과 같음

```
사용자 고유번호 6자리 입력: 777777
조회할 계좌번호 6자리 입력: 666666
입출금 내역이 없습니다.
```

[매니저 메뉴] - [사용자 정보 수정]

사용자 고유번호 입력받고 메뉴 진입

```
선택할 메뉴를 입력해주세요: 5
사용자 고유번호 6자리 입력: 777777
0. 뒤로가기
1. 이름
2. 주소
3. 관리자 고유번호
4. 연락처 추가
5. 연락처 수정
6. 연락처 삭제
수정할 정보 입력:
```

[매니저 메뉴] - [사용자 정보 수정] - [이름]

```
수정할 정보 입력: 1
이름 입력: yoon
이름 수정이 성공적으로 완료되었습니다.
```

수정 전 사용자 정보

```
고유번호: 777777 | 이름: Kim | 주소: 서울시 성북구 쌍문동 waw | 관리자 고유번호: 111111 | 대표
[1번째 계좌] 계좌번호: 666666 | 잔액: 35000 | 계좌개설일: 2021/08/08 | 비밀번호: 666666
```

수정 후 사용자 정보

```
고유번호: 777777 | 이름: yoon | 주소: 서울시 성북구 쌍문동 waw | 관리자 고유번호: 111111 | 대표
[1번째 계좌] 계좌번호: 666666 | 잔액: 35000 | 계좌개설일: 2021/08/08 | 비밀번호: 666666
```

[매니저 메뉴] - [사용자 정보 수정] - [주소]

```
수정할 정보 입력: 2
주소 시/도 입력: 경기도
주소 시/군/구 입력: 수원시
주소 읍/면/동 입력: 파장동
상세주소 입력: rre
주소 수정이 성공적으로 완료되었습니다.
```

수정 전 사용자 정보

```
고유번호: 777777 | 이름: yoon | 주소: 서울시 성북구 쌍문동 waw | 관리자 고유번호: 111111 | 대표
[1번째 계좌] 계좌번호: 666666 | 잔액: 35000 | 계좌개설일: 2021/08/08 | 비밀번호: 666666
```

수정 후 사용자 정보

```
고유번호: 777777 | 이름: yoon | 주소: 경기도 수원시 파장동 rre | 관리자 고유번호: 111111 | 대표
[1번째 계좌] 계좌번호: 666666 | 잔액: 35000 | 계좌개설일: 2021/08/08 | 비밀번호: 666666
```

[매니저 메뉴] - [사용자 정보 수정] - [관리자 고유번호]

```
수정할 정보 입력: 3
관리자 고유번호 6자리 입력: 999999
관리자 고유번호 수정이 성공적으로 완료되었습니다.
```

수정 전 사용자 정보

```
-----------------------------------------------------------------------------------------------------
고유번호: 777777 | 이름: yoon | 주소: 경기도 수원시 파장동 rre | 관리자 고유번호: 111111 | 대표
[1번째 계좌] 계좌번호: 666666 | 잔액: 35000 | 계좌개설일: 2021/08/08 | 비밀번호: 666666
-----------------------------------------------------------------------------------------------------
```

수정 후 사용자 정보

```
-----------------------------------------------------------------------------------------------------
고유번호: 777777 | 이름: yoon | 주소: 경기도 수원시 파장동 rre | 관리자 고유번호: 999999 | 대표
[1번째 계좌] 계좌번호: 666666 | 잔액: 35000 | 계좌개설일: 2021/08/08 | 비밀번호: 666666
-----------------------------------------------------------------------------------------------------
```

[매니저 메뉴] - [사용자 정보 수정] - [연락처 추가]

```
수정할 정보 입력: 4
추가할 연락처 입력: 01077777777
```

수정 전 사용자 정보

```
-----------------------------------------------------------------------------------------------------
고유번호: 777777 | 이름: yoon | 주소: 경기도 수원시 파장동 rre | 관리자 고유번호: 999999 | 대표 전화번호: 01078457845
[1번째 계좌] 계좌번호: 666666 | 잔액: 35000 | 계좌개설일: 2021/08/08 | 비밀번호: 666666
-----------------------------------------------------------------------------------------------------
```

수정 후 사용자 정보

```
-----------------------------------------------------------------------------------------------------
고유번호: 777777 | 이름: yoon | 주소: 경기도 수원시 파장동 rre | 관리자 고유번호: 999999 | 대표 전화번호: 01077777777
[1번째 계좌] 계좌번호: 666666 | 잔액: 35000 | 계좌개설일: 2021/08/08 | 비밀번호: 666666
-----------------------------------------------------------------------------------------------------
```

[매니저 메뉴] - [사용자 정보 수정] - [연락처 수정]

```
수정할 정보 입력: 5
해당 사용자의 연락처 목록은 다음과 같습니다.
01077777777
01078457845
어떤 연락처를 수정하시겠습니까?: 01077777777
등록할 번호 입력: 01011111111
```

수정 전 연락처 정보

```
mysql> select * from phone_number;
+--------------+--------+
| phone_number | Ssn    |
+--------------+--------+
| 01066666666  | 010101 |
| 01077777777  | 777777 |
| 01078457845  | 777777 |
| 01012121212  | 961225 |
+--------------+--------+
```

수정 후 연락처 정보

```
mysql> select * from phone_number;
+--------------+--------+
| phone_number | Ssn    |
+--------------+--------+
| 01066666666  | 010101 |
| 01011111111  | 777777 |
| 01078457845  | 777777 |
| 01012121212  | 961225 |
+--------------+--------+
```

[매니저 메뉴] - [사용자 정보 수정] - [연락처 삭제]

```
수정할 정보 입력: 6
해당 사용자의 연락처 목록은 다음과 같습니다.
01011111111
01078457845
삭제할 연락처 입력: 01011111111
```

수정 전 연락처 정보

```
mysql> select * from phone_number;
+--------------+--------+
| phone_number | Ssn    |
+--------------+--------+
| 01066666666  | 010101 |
| 01011111111  | 777777 |
| 01078457845  | 777777 |
| 01012121212  | 961225 |
+--------------+--------+
```

수정 후 연락처 정보

```
mysql> select * from phone_number;
+--------------+--------+
| phone_number | Ssn    |
+--------------+--------+
| 01066666666  | 010101 |
| 01078457845  | 777777 |
| 01012121212  | 961225 |
+--------------+--------+
```

[매니저 메뉴] - [사용자 정보 삭제]

```
선택할 메뉴를 입력해주세요: 6
사용자 고유번호 6자리 입력: 777777
사용자 정보가 성공적으로 삭제되었습니다.
```

수정 전 사용자 정보

```
------------------------------------------------------------------------------------------------------------------------
고유번호: 010101 | 이름: yoon | 주소: 서울시 성동구 용답동 abc | 관리자 고유번호: 111111 | 대표 전화번호: 01066666666
[1번째 계좌] 계좌번호: 000001 | 잔액: 1000 | 계좌개설일: 2021/10/10 | 비밀번호: 000001
[2번째 계좌] 계좌번호: 010101 | 잔액: 1000 | 계좌개설일: 2021/06/06 | 비밀번호: 010101
------------------------------------------------------------------------------------------------------------------------
고유번호: 777777 | 이름: yoon | 주소: 경기도 수원시 파장동 rre | 관리자 고유번호: 999999 | 대표 전화번호: 01078457845
[1번째 계좌] 계좌번호: 666666 | 잔액: 35000 | 계좌개설일: 2021/08/08 | 비밀번호: 666666
------------------------------------------------------------------------------------------------------------------------
고유번호: 961225 | 이름: 홍길동 | 주소: 경기도 이천시 창전동 cba | 관리자 고유번호: 111111 | 대표 전화번호: 01012121212
------------------------------------------------------------------------------------------------------------------------
```

수정 후 사용자 정보

```
------------------------------------------------------------------------------------------------------------------------
고유번호: 010101 | 이름: yoon | 주소: 서울시 성동구 용답동 abc | 관리자 고유번호: 111111 | 대표 전화번호: 01066666666
[1번째 계좌] 계좌번호: 000001 | 잔액: 1000 | 계좌개설일: 2021/10/10 | 비밀번호: 000001
[2번째 계좌] 계좌번호: 010101 | 잔액: 1000 | 계좌개설일: 2021/06/06 | 비밀번호: 010101
------------------------------------------------------------------------------------------------------------------------
고유번호: 961225 | 이름: 홍길동 | 주소: 경기도 이천시 창전동 cba | 관리자 고유번호: 111111 | 대표 전화번호: 01012121212
------------------------------------------------------------------------------------------------------------------------
```

[매니저 메뉴] - [사용자 정보 조회]

```
------------------------------------------------------------------------------------------------------------------------
고유번호: 010101 | 이름: yoon | 주소: 서울시 성동구 용답동 abc | 관리자 고유번호: 111111 | 대표 전화번호: 01066666666
[1번째 계좌] 계좌번호: 000001 | 잔액: 1000 | 계좌개설일: 2021/10/10 | 비밀번호: 000001
[2번째 계좌] 계좌번호: 010101 | 잔액: 1000 | 계좌개설일: 2021/06/06 | 비밀번호: 010101
------------------------------------------------------------------------------------------------------------------------
고유번호: 961225 | 이름: 홍길동 | 주소: 경기도 이천시 창전동 cba | 관리자 고유번호: 111111 | 대표 전화번호: 01012121212
------------------------------------------------------------------------------------------------------------------------
```

[매니저 메뉴] - [관리자 계정 생성]

```
선택할 메뉴를 입력해주세요: 8
관리자 고유번호 6자리 입력: 222222
이름 입력: mas
비밀번호 입력: 222222
관리자 정보가 성공적으로 등록되었습니다.
```

수정 전 관리자 정보

```
------------------------------------------------------------
고유번호: 111111 | 이름: yoon | 비밀번호: 111111
------------------------------------------------------------
고유번호: 999999 | 이름: master | 비밀번호: 999999
------------------------------------------------------------
```

수정 후 관리자 정보

```
------------------------------------------------------------
고유번호: 111111 | 이름: yoon | 비밀번호: 111111
------------------------------------------------------------
고유번호: 222222 | 이름: mas | 비밀번호: 222222
------------------------------------------------------------
고유번호: 999999 | 이름: master | 비밀번호: 999999
------------------------------------------------------------
```

[매니저 메뉴] - [관리자 정보 수정]

```
선택할 메뉴를 입력해주세요: 9
관리자 고유번호 6자리 입력: 222222
비밀번호 입력: 222222
0. 뒤로가기
1. 이름
2. 비밀번호
수정할 정보를 입력해주세요:
```

[매니저 메뉴] - [관리자 정보 수정] - [이름]

```
수정할 정보를 입력해주세요: 1
원하는 이름 입력: MAS
이름이 성공적으로 등록되었습니다.
```

수정 전 관리자 정보

```
------------------------------------------------------------
고유번호: 111111 | 이름: yoon | 비밀번호: 111111
------------------------------------------------------------
고유번호: 222222 | 이름: mas | 비밀번호: 222222
------------------------------------------------------------
고유번호: 999999 | 이름: master | 비밀번호: 999999
------------------------------------------------------------
```

수정 후 관리자 정보

```
------------------------------------------------------------
고유번호: 111111 | 이름: yoon | 비밀번호: 111111
------------------------------------------------------------
고유번호: 222222 | 이름: MAS | 비밀번호: 222222
------------------------------------------------------------
고유번호: 999999 | 이름: master | 비밀번호: 999999
------------------------------------------------------------
```

[매니저 메뉴] - [관리자 정보 수정] - [비밀번호]

```
수정할 정보를 입력해주세요: 2
원하는 비밀번호 입력: 000000
비밀번호가 성공적으로 등록되었습니다.
```

수정 전 관리자 정보

```
------------------------------------------------------------
고유번호: 111111 | 이름: yoon | 비밀번호: 111111
------------------------------------------------------------
고유번호: 222222 | 이름: MAS | 비밀번호: 222222
------------------------------------------------------------
고유번호: 999999 | 이름: master | 비밀번호: 999999
------------------------------------------------------------
```

수정 후 관리자 정보

```
------------------------------------------------------------
고유번호: 111111 | 이름: yoon | 비밀번호: 111111
------------------------------------------------------------
고유번호: 222222 | 이름: MAS | 비밀번호: 000000
------------------------------------------------------------
고유번호: 999999 | 이름: master | 비밀번호: 999999
------------------------------------------------------------
```

[매니저 메뉴] - [관리자 정보 삭제]

```
선택할 메뉴를 입력해주세요: 10
삭제할 관리자 고유번호 6자리 입력: 222222
비밀번호 입력: 000000
성공적으로 삭제되었습니다.
```

수정 전 관리자 정보

```
------------------------------------------------------------
고유번호: 111111 | 이름: yoon | 비밀번호: 111111
------------------------------------------------------------
고유번호: 222222 | 이름: MAS | 비밀번호: 000000
------------------------------------------------------------
고유번호: 999999 | 이름: master | 비밀번호: 999999
------------------------------------------------------------
```

수정 후 관리자 정보

```
------------------------------------------------------------
고유번호: 111111 | 이름: yoon | 비밀번호: 111111
------------------------------------------------------------
고유번호: 999999 | 이름: master | 비밀번호: 999999
------------------------------------------------------------
```

[매니저 메뉴] - [관리자 정보 조회]

```
------------------------------------------------------------
고유번호: 111111 | 이름: yoon | 비밀번호: 111111
------------------------------------------------------------
고유번호: 999999 | 이름: master | 비밀번호: 999999
------------------------------------------------------------
```

[사용자 메뉴]

```
선택할 메뉴를 입력해주세요: 2
0. 종료
1. 뒤로가기
2. 본인 계좌로 이체
3. 타인 계좌로 이체
선택할 메뉴를 입력해주세요:
```

[사용자 메뉴] - [본인 계좌로 이체]

```
선택할 메뉴를 입력해주세요: 2
사용자 고유번호 6자리 입력: 010101
출금은 0, 입금은 1, 뒤로가려면 -1을 입력해주세요:
```

[사용자 메뉴] - [본인 계좌로 이체] - [출금]

```
출금은 0, 입금은 1, 뒤로가려면 -1을 입력해주세요: 0
출금할 계좌번호 6자리 입력: 010101
비밀번호 입력: 010101
출금할 금액 입력: 200
날짜 입력 (예) 2021년 5월 7일 -> 20210507 : 20211203
출금이 성공적으로 완료되었습니다.
```

수정 전 계좌 정보

```
-----------------------------------------------
고유번호: 010101 | 이름: yoon | 주소: 서울시
[1번째 계좌] 계좌번호: 000001 | 잔액: 1000 |
[2번째 계좌] 계좌번호: 010101 | 잔액: 2000 |
-----------------------------------------------
```

수정 후 계좌 정보

```
-----------------------------------------------
고유번호: 010101 | 이름: yoon | 주소: 서울시
[1번째 계좌] 계좌번호: 000001 | 잔액: 1000 |
[2번째 계좌] 계좌번호: 010101 | 잔액: 1800 |
-----------------------------------------------
```

입출금 시 내역에도 기록이 됨

```
-----------------------------------------------
[출금] 날짜:2021/12/01 금액:525
[출금] 날짜:2021/12/01 금액:1000
[입금] 날짜:2021/12/02 금액:525
[입금] 날짜:2021/12/02 금액:2525
[입금] 날짜:2021/12/03 금액:200
[출금] 날짜:2021/12/03 금액:200
[입금] 날짜:2021/12/03 금액:1000
[출금] 날짜:2021/12/03 금액:1000
-----------------------------------------------
```

[사용자 메뉴] - [본인 계좌로 이체] - [입금]

```
출금은 0, 입금은 1, 뒤로가려면 -1을 입력해주세요: 1
입금할 계좌번호 6자리 입력: 010101
비밀번호 입력: 010101
입금할 금액 입력: 95000
날짜 입력 예) 2021년 5월 7일 -> 20210507 : 20211203
입금이 성공적으로 완료되었습니다.
```

수정 전 계좌 정보

```
--------------------------------------------------
고유번호: 010101 | 이름: yoon | 주소: 서울시
[1번째 계좌] 계좌번호: 000001 | 잔액: 1000 |
[2번째 계좌] 계좌번호: 010101 | 잔액: 1800 |
--------------------------------------------------
```

수정 후 계좌 정보

```
--------------------------------------------------
고유번호: 010101 | 이름: yoon | 주소: 서울시
[1번째 계좌] 계좌번호: 000001 | 잔액: 1000 |
[2번째 계좌] 계좌번호: 010101 | 잔액: 96800 |
--------------------------------------------------
고유번호: 961225 | 이름: 홍길동 | 주소: 경기도
--------------------------------------------------
```

[사용자 메뉴] - [본인 계좌로 이체] - [뒤로가기]]

```
출금은 0, 입금은 1, 뒤로가려면 -1을 입력해주세요: -1
--------------------
0. 종료
1. 뒤로가기
2. 본인 계좌로 이체
3. 타인 계좌로 이체
선택할 메뉴를 입력해주세요:
```

[사용자 메뉴] - [타인 계좌로 이체]

```
선택할 메뉴를 입력해주세요: 3
사용자 고유번호 6자리 입력: 010101
입금 계좌번호 6자리 입력: 000001
출금 계좌번호 6자리 입력: 010101
비밀번호 입력: 010101
출금할 금액 입력: 50000
날짜 입력 예) 2021년 5월 7일 -> 20210507 : 20211203
```

수정 전 계좌 정보

```
--------------------------------------------------
고유번호: 010101 | 이름: yoon | 주소: 서울시
[1번째 계좌] 계좌번호: 000001 | 잔액: 1000 |
[2번째 계좌] 계좌번호: 010101 | 잔액: 96800 |
--------------------------------------------------
고유번호: 961225 | 이름: 홍길동 | 주소: 경기도
--------------------------------------------------
```

수정 후 계좌 정보

```
--------------------------------------------------
고유번호: 010101 | 이름: yoon | 주소: 서울시
[1번째 계좌] 계좌번호: 000001 | 잔액: 51000 |
[2번째 계좌] 계좌번호: 010101 | 잔액: 46800 |
--------------------------------------------------
```

3. 사용되는 SQL문 명세

[매니저 메뉴] - 매니저 정보 없을 때 새 매니저 계정 등록

```
sql = 'INSERT INTO manager (Ssn, Name, Password) VALUES (%s, %s, %s)'
```

[매니저 메뉴] - 매니저 정보있을 때 로그인을 위한 SQL (ID, PW 체크)

```
sql = 'SELECT * FROM manager WHERE Ssn=' + input_ssn + ' AND Password=' + password
```

[매니저 메뉴] - [신규 사용자 등록] - 중복 체크

```
sql = 'SELECT * FROM customer WHERE Ssn=' + input_ssn
```

[매니저 메뉴] - [신규 사용자 등록] - 정보 등록

```
sql = 'INSERT INTO customer (Ssn, Name, Si_do, Si_gun_gu, Eup_myeon_dong, Detailed_address, ' \
      'Mssn) VALUES (%s, %s, %s, %s, %s, %s, %s)'
```

[매니저 메뉴] - [신규 사용자 등록] - 연락처 등록

```
cursor.execute('INSERT INTO phone_number (phone_number, Ssn) VALUES (%s, %s)',
               (phone_number, input_ssn))
```

[매니저 메뉴] - [사용자 계좌 관리] - [계좌 생성] - 고유번호 확인

```
'SELECT * FROM customer WHERE Ssn=' + customer_ssn
```

[매니저 메뉴] - [사용자 계좌 관리] - [계좌 생성] - 계좌번호 중복 확인

```
'SELECT * FROM account WHERE Number=' + account_number
```

[매니저 메뉴] - [사용자 계좌 관리] - [계좌 생성] - 계좌 정보 등록

```
sql = 'INSERT INTO account (Number, Balance, Open_date, Password, Ssn) ' \
      'VALUES (%s, %s, %s, %s, %s)'
```

[매니저 메뉴] - [사용자 계좌 관리] - [계좌 삭제] - 계좌 정보 삭제

foreign key constraint 설정 해제 후 해당 계좌와 관련된 입출금 내역 먼저 삭제 이후 계좌 삭제 진행 및 foreign key constraint 설정

```
# 사용자 고유번호 foreign key 설정 해제
cursor.execute('SET foreign_key_checks = 0')
connection.commit()

# 관련 입출금 내역 삭제
cursor.execute('DELETE FROM history WHERE Anum=' + account_number)
connection.commit()

# 계좌 삭제
cursor.execute(
    'DELETE FROM account WHERE Number=' + account_number + ' AND Password=' + password)
connection.commit()

# 사용자 고유번호 foreign key 설정
cursor.execute('SET foreign_key_checks = 1')
connection.commit()
```

[매니저 메뉴] - [사용자 입출금 내역 관리] - [입출금 내역 조회]

고유번호 확인

```
cursor.execute('SELECT * FROM customer WHERE Ssn=' + customer_ssn)
```

계좌 조회

```
cursor.execute('SELECT * FROM account WHERE Number=' + account_number +
               ' AND Ssn=' + customer_ssn)
```

입출금 내역 조회

```
sql = 'SELECT * FROM history WHERE Anum=' + account_number
```

[매니저 메뉴] - [사용자 입출금 내역 관리] - [입출금 내역 수정]

고유번호 확인

```
cursor.execute('SELECT * FROM customer WHERE Ssn=' + customer_ssn)
```

계좌 조회

```
cursor.execute('SELECT * FROM account WHERE Number=' + account_number +
               ' AND Ssn=' + customer_ssn)
```

수정할 입출금 내역 특정

```
sql = 'SELECT * FROM history WHERE Anum=' + account_number + ' AND Date=' + date + \
      ' AND Amount=' + balance
```

날짜 업데이트

```
cursor.execute('UPDATE history SET Date = %s WHERE Anum = %s AND Date = %s '
               'AND Amount = %s', (new_date, account_number, date, balance))
```

금액 업데이트

```
cursor.execute('UPDATE history SET Amount = %s WHERE Anum = %s AND Date = %s '
               'AND Amount = %s', (new_balance, account_number, date, balance))
```

입출금 유형 업데이트

입금이면 출금으로, 출금이면 입금으로 수정

```
if target_history[3] == '입금':

    cursor.execute(
        'UPDATE history SET Type = %s WHERE Anum = %s AND Date = %s '
        'AND Amount = %s', ('출금', account_number, date, balance)
    )
    connection.commit()

else:

    cursor.execute(
        'UPDATE history SET Type = %s WHERE Anum = %s AND Date = %s '
        'AND Amount = %s', ('입금', account_number, date, balance)
    )
    connection.commit()
```

[매니저 메뉴] - [사용자 입출금 내역 관리] - [입출금 내역 삭제]

고유번호 확인

```
cursor.execute('SELECT * FROM customer WHERE Ssn=' + customer_ssn)
```

계좌 조회

```
cursor.execute('SELECT * FROM account WHERE Number=' + account_number +
               ' AND Ssn=' + customer_ssn)
```

수정할 입출금 내역 특정

```
sql = 'SELECT * FROM history WHERE Anum=' + account_number + ' AND Date=' + date + \
      ' AND Amount=' + balance
```

입출금 내역 삭제

```
cursor.execute('DELETE FROM history WHERE Anum=' + account_number + ' AND Date=' + date + \
  ' AND Amount=' + balance)
```

[매니저 메뉴] - [사용자 정보 수정]

고유번호 확인

```
cursor.execute('SELECT * FROM customer WHERE Ssn=' + customer_ssn)
```

이름 수정

```
cursor.execute('UPDATE customer SET Name = %s WHERE Ssn = %s', (name, customer_ssn))
```

주소 수정

```
cursor.execute(
    'UPDATE customer SET Si_do = %s, Si_gun_gu = %s, Eup_myeon_dong = %s, Detailed_address = %s '
    'WHERE Ssn = %s', (si_do, si_gun_gu, eup_myeon_dong, detailed_address, customer_ssn))
```

관리자 고유번호 수정

```
cursor.execute('UPDATE customer SET Mssn = %s WHERE Ssn = %s', (mgr_ssn, customer_ssn))
```

연락처 추가

```
cursor.execute('INSERT INTO phone_number (phone_number, Ssn) VALUES (%s, %s)',
               (phone_number, customer_ssn))
```

일치하는 연락처가 존재하는지 조회

```
cursor.execute('SELECT * FROM phone_number WHERE phone_number=' + current_number)
```

연락처 수정

```
cursor.execute('UPDATE phone_number SET phone_number = %s WHERE phone_number = %s',
               (update_number, current_number))
```

삭제할 연락처가 존재하는지 조회

```
cursor.execute('SELECT * FROM phone_number WHERE phone_number=' + phone_number)
```

연락처 삭제

```
cursor.execute('DELETE FROM phone_number WHERE phone_number=' + phone_number)
```

[매니저 메뉴] - [사용자 정보 삭제]

고유번호 확인

```
cursor.execute('SELECT * FROM customer WHERE Ssn=' + customer_ssn)
```

foreign key constraint 설정 해제 후 해당 사용자 관련 정보(입출금 내역, 계좌, 연락처) 먼저 삭제

사용자 정보 삭제 및 foreign key constraint 설정

```
# 사용자 고유번호 foreign key 설정 해제
cursor.execute('SET foreign_key_checks = 0')
connection.commit()

# 관련 입출금 내역 삭제
cursor.execute('SELECT * FROM account WHERE Ssn=' + customer_ssn)
accounts = cursor.fetchall()

for account in accounts:
    account_number = account[0]
    cursor.execute('DELETE FROM history WHERE Anum=' + account_number)
    connection.commit()

# 관련 계좌 삭제
cursor.execute('DELETE FROM account WHERE Ssn=' + customer_ssn)
connection.commit()

# 관련 연락처 삭제
cursor.execute('DELETE FROM phone_number WHERE Ssn=' + customer_ssn)
connection.commit()

# 사용자 정보 삭제
cursor.execute('DELETE FROM customer WHERE Ssn=' + customer_ssn)
connection.commit()

# 사용자 고유번호 foreign key 설정
cursor.execute('SET foreign_key_checks = 1')
connection.commit()
```

[매니저 메뉴] - [사용자 정보 조회]

모든 사용자(customer) 정보 조회

```
cursor.execute('SELECT * FROM customer')
```

각 사용자의 phone_number 조회

```
cursor.execute('SELECT * FROM phone_number WHERE Ssn=' + Ssn)
```

각 사용자의 계좌 조회

```
cursor.execute('SELECT * FROM account WHERE Ssn=' + Ssn)
```

[매니저 메뉴] - [관리자 계정 생성]

```
sql = 'INSERT INTO manager (Ssn, Name, Password) VALUES (%s, %s, %s)'
```

[매니저 메뉴] - [관리자 정보 수정]

수정할 관리자 정보 탐색

```
cursor.execute('SELECT * FROM manager WHERE Ssn=' + mgr_ssn + ' AND Password=' + password)
```

이름 수정

```
cursor.execute('UPDATE manager SET Name = %s WHERE Ssn = %s', (name, mgr_ssn))
```

비밀번호 수정

```
cursor.execute('UPDATE manager SET Password = %s WHERE Ssn = %s',
               (password, mgr_ssn))
```

[매니저 메뉴] - [관리자 계정 삭제]

삭제할 관리자 정보 탐색

```
cursor.execute('SELECT * FROM manager WHERE Ssn=' + mgr_ssn + ' AND Password=' + password)
```

foreign key constraint 설정 해제 후 해당 관리자와 관련된 사용자가 존재할 경우 기존의 관리자 번호로 수정 및 foreign key constraint 설정

```
# 사용자의 관리자 고유번호 foreign key 설정 해제
cursor.execute('SET foreign_key_checks = 0')
connection.commit()

cursor.execute('SELECT * FROM manager')
mgr_count = cursor.rowcount
fir_mgr = cursor.fetchone()

# 관리자가 최소 인원(1명)이면 삭제 금지
if mgr_count <= 1:
    print('관리자는 최소 1명 있어야 합니다. 삭제를 종료합니다.')

# 해당 관리자 고유번호가 등록된 사용자 탐색 후 재설정
else:

    while fir_mgr[0] == mgr_ssn:
        fir_mgr = cursor.fetchone()

    cursor.execute('UPDATE customer SET Mssn = %s WHERE Mssn = %s', (fir_mgr[0], mgr_ssn))
    connection.commit()

    cursor.execute('DELETE FROM manager WHERE Ssn=' + mgr_ssn)
    connection.commit()
    print('성공적으로 삭제되었습니다.')

# 사용자의 관리자 고유번호 foreign key 설정
cursor.execute('SET foreign_key_checks = 1')
connection.commit()
```

[매니저 메뉴] - [관리자 계정 조회]

모든 관리자 계정 조회

```
cursor.execute('SELECT * FROM manager')
```

[사용자 메뉴] - [본인 계좌 이체]

사용자 정보 존재하는지 확인

```
cursor.execute('SELECT * FROM customer WHERE Ssn=' + user_ssn)
```

출금할 계좌 특정

```
cursor.execute(
    'SELECT * FROM account WHERE Number=' + from_account + ' AND Ssn=' + user_ssn)
```

출금 시 잔액 업데이트

```
sql = 'UPDATE account SET Balance = %s WHERE Number = %s'
cursor.execute(sql, (int(account[1]) - int(amount), from_account))
```

출금 내역 입력

```
sql = 'INSERT INTO history (Date, Amount, Anum, Type) VALUES (%s, %s, %s, %s)'
cursor.execute(sql, (date, amount, from_account, '출금'))
```

입금할 계좌 특정

```
cursor.execute(
    'SELECT * FROM account WHERE Number=' + from_account + ' AND Ssn=' + user_ssn)
```

입금 시 잔액 업데이트

```
sql = 'UPDATE account SET Balance = %s WHERE Number = %s'
cursor.execute(sql, (int(account[1]) + int(amount), from_account))
```

입금 내역 입력

```
sql = 'INSERT INTO history (Date, Amount, Anum, Type) VALUES (%s, %s, %s, %s)'
cursor.execute(sql, (date, amount, from_account, '입금'))
```

[사용자 메뉴] - [타인 계좌 이체]

출금할 계좌의 사용자 정보 조회

```
cursor.execute('SELECT * FROM customer WHERE Ssn=' + user_ssn)
```

입금할 계좌 존재 여부 확인

```
cursor.execute('SELECT * FROM account WHERE Number=' + to_account)
```

출금할 계좌 특정

```
cursor.execute(
    'SELECT * FROM account WHERE Number=' + from_account + ' AND Ssn=' + user_ssn)
```

입금 및 출금 잔액 업데이트

```
sql = 'UPDATE account SET Balance = %s WHERE Number = %s'
cursor.execute(sql, (int(account[1]) - int(amount), from_account))
connection.commit()

sql = 'UPDATE account SET Balance = %s WHERE Number = %s'
cursor.execute(sql, (int(to_account_info[1]) + int(amount), to_account))
connection.commit()
```

입출금 내역 입력

```
# 출금 내역 입력
sql = 'INSERT INTO history (Date, Amount, Anum, Type) VALUES (%s, %s, %s, %s)'
cursor.execute(sql, (date, amount, from_account, '출금'))
connection.commit()

# 입금 내역 입력
sql = 'INSERT INTO history (Date, Amount, Anum, Type) VALUES (%s, %s, %s, %s)'
cursor.execute(sql, (date, amount, to_account, '입금'))
connection.commit()
```