은행 서비스 DBMS

프로젝트 4:

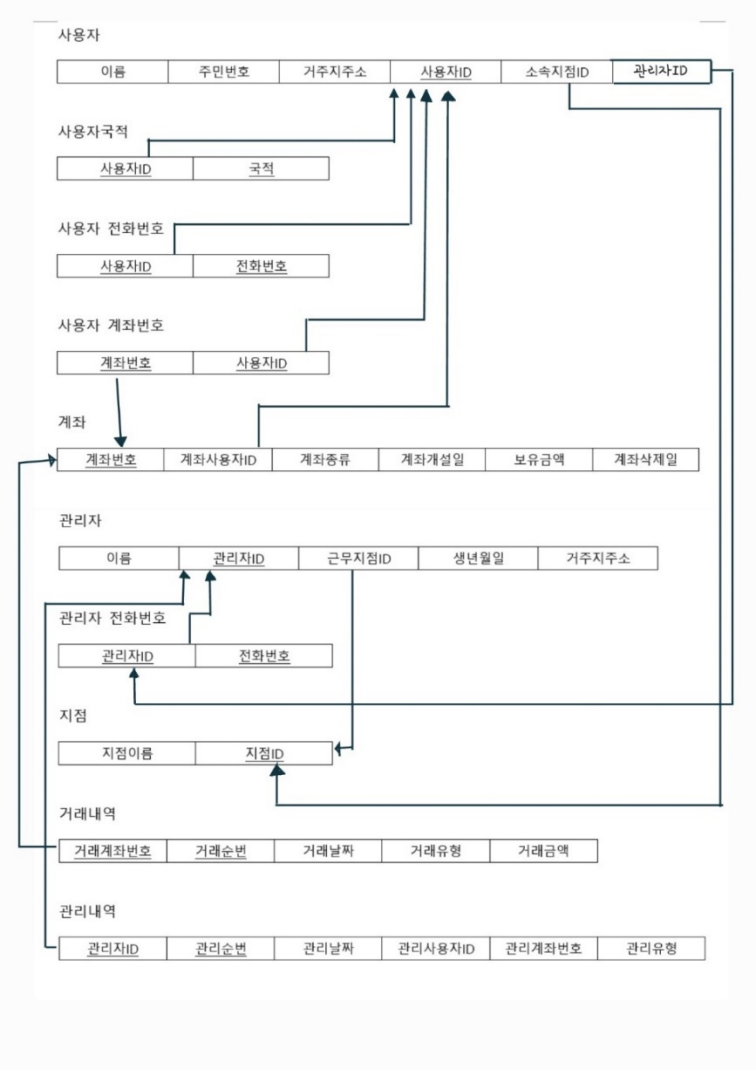
DBMS 프로그램 개발

2018008859

이민준

목차

1. 기존 Schema 및 수정 사항



① "계좌"의 "계좌삭제일"을 불필요하다고 판단되어 삭제합니다.

② "사용자 계좌번호" Table을 불필요하다고 판단되어 삭제합니다.

③ "거래내역"의 Primary Key를 "거래순번" 1개로 변경합니다.

④ "관리내역"의 Primary Key를 "관리순번" 1개로 변경합니다.

⑤ "관리내역"의 "관리사용자ID"를 불필요하다고 판단되어 삭제합니다.

⑥ "관리내역"의 "관리계좌번호"가 "계좌"의 "계좌번호"를 Foreign Key로 삼도록 조건을 추가합니다.

2. 프로그램 코드 설명 및 SQL문 명세

- main function

```
922  def main():
923      create_bank_dbms()
924
925      print("Welcome to Bank DBMS")
926
927      while True:
928          print()
929          print("==============================")
930          print("Select Service:")
931          print("0.Exit")
932          print("1.User")
933          print("2.Administrator")
934          print("==============================")
935          print("Input: ", end='')
936          command = int(input())
937          if command == 0:
938              break
939          elif command == 1:
940              user_interface()
941          elif command == 2:
942              admin_interface()
943          else:
944              print("Wrong Input! Please try again")
945
945
946      connection.commit()
947      connection.close()
948
949
950  if __name__ == '__main__':
951      main()
```

- create_bank_dbms()로 만들고자 하는 DB가 없을 시, create

- Bank DBMS의 시작 Interface 구현

- input command에 따라 0, 1, 2를 입력 받고 해당하는 Interface로 이동

- 0을 입력 받아 while문 종료 시, 마지막으로 commit 후 connection close

- create_bank_dbms()

```
15  def create_bank_dbms():
16      cursor.execute('CREATE DATABASE IF NOT EXISTS BANK;')
17      cursor.execute('use BANK')
```

- CREATE DATABASE IF NOT EXISTS BANK

√. BANK라는 DATABASE가 존재하지 않는다면 BANK 생성

√. BRANCH

```
19      cursor.execute('CREATE TABLE IF NOT EXISTS BRANCH ('
20                     'branch_name VARCHAR(20) NOT NULL,'
21                     'branch_id INT NOT NULL AUTO_INCREMENT,'
22                     'PRIMARY KEY (branch_id),'
23                     'UNIQUE (branch_name) );')
```

- CREATE TABLE IF NOT EXISTS BRANCH

√. BRANCH라는 table이 존재하지 않는다면 생성

- branch_name: 최대 20개의 char를 받을 수 있는 Unique Attribute

- branch_id: int 형식의 BRANCH table의 Primary Key

- INSERT시, branch_name만 입력하면 branch_id는 auto_increment 옵션에 의해

오름차순으로 자동 설정 (AUTO_INCREMENT)

√. ADMINISTRATOR

```
25      cursor.execute('CREATE TABLE IF NOT EXISTS ADMINISTRATOR ('
26                     'administrator_name VARCHAR(20) NOT NULL,'
27                     'administrator_id INT NOT NULL AUTO_INCREMENT,'
28                     'branch_id INT NOT NULL DEFAULT 1,'
29                     'birth_date DATE NOT NULL,'
30                     'administrator_address VARCHAR(20) NOT NULL,'
31                     'PRIMARY KEY (administrator_id),'
32                     'FOREIGN KEY (branch_id) REFERENCES BRANCH(branch_id) '
33                     'ON DELETE SET DEFAULT ON UPDATE CASCADE );')
```

- CREATE TABLE IF NOT EXISTS ADMINISTRATOR

√. ADMINISTRATOR라는 table이 존재하지 않는다면 생성

- administrator_name: 최대 20개의 char을 받을 수 있는 Attribute

- administrator_id: int 형식의 ADMINISTRATOR table의 Primary Key

- INSERT시, administrator_id를 제외한 attribute을 입력하면 administrator_id는 auto_increment 옵션에 의해 오름차순으로 자동 설정 (AUTO_INCREMENT)

- branch_id: int 형식의 BRANCH의 Primary Key인 branch_id를 Foreign Key로 가져온 DEFAULT 값이 1인 Attribute

- Foreign Key로 가리키던 BRANCH의 branch_id가 DELETE 될 경우 DEFAULT 값인 1로 변경 (ON DELETE SET DEFAULT)

- Foreign Key로 가리키던 BRANCH의 branch_id가 UPDATE 될 경우 변경된 값에 따라감 (ON UPDATE CASCADE)

- birth_date: date 형식의 Attribute

- administrator_address: 최대 20개의 char을 받을 수 있는 Attribute

√. ADMINISTRATOR_PHONE_NUMBER

```
83      cursor.execute('CREATE TABLE IF NOT EXISTS ADMINISTRATOR_PHONE_NUMBER ('
84                     'administrator_id INT NOT NULL,'
85                     'phone_number VARCHAR(13) NOT NULL,'
86                     'PRIMARY KEY (administrator_id, phone_number),'
87                     'FOREIGN KEY (administrator_id) REFERENCES ADMINISTRATOR(administrator_id)'
88                     'ON DELETE CASCADE );')
```

- CREATE TABLE IF NOT EXISTS ADMINISTRATOR_PHONE_NUMBER

√. ADMINISTRATOR_PHONE_NUMBER라는 table이 존재하지 않는다면 생성

- administrator_id: int 형식의 ADMINISTRATOR의 Primary Key인 administrator_id를 Foreign Key로 가져온 Attribute

- Foreign Key로 가리키던 USER의 administrator_id가 DELETE 될 경우 같이 삭제됨 (ON DELETE CASCADE)

- phone_number: 최대 13개의 char을 받을 수 있는 Attribute

- administrator_id와 phone_number가 Primary Key를 이룸

√. USER

```
35      cursor.execute('CREATE TABLE IF NOT EXISTS USER ('
36                     'user_name VARCHAR(20) NOT NULL,'
37                     'user_ssn VARCHAR(14) NOT NULL,'
38                     'user_address VARCHAR(20) NOT NULL,'
39                     'user_id INT NOT NULL AUTO_INCREMENT,'
40                     'belong_branch_id INT NOT NULL DEFAULT 1,'
41                     'treated_administrator_id INT NOT NULL DEFAULT 1,'
42                     'PRIMARY KEY (user_id),'
43                     'FOREIGN KEY (belong_branch_id) REFERENCES BRANCH(branch_id)'
44                     'ON DELETE SET DEFAULT ON UPDATE CASCADE,'
45                     'FOREIGN KEY (treated_administrator_id) REFERENCES ADMINISTRATOR(administrator_id)'
46                     'ON DELETE SET DEFAULT ON UPDATE CASCADE,'
47                     'UNIQUE(user_ssn) );')
```

- CREATE TABLE IF NOT EXISTS USER

√. USER라는 table이 존재하지 않는다면 생성

- user_name: 최대 20개의 char을 받을 수 있는 Attribute

- user_ssn: 최대 14개의 char을 받을 수 있는 Unique Attribute

- user_address: 최대 20개의 char을 받을 수 있는 Attribute

- user_id: int 형식의 USER table의 Primary Key

- INSERT시, user_id를 제외한 Attribute을 입력하면 user_id는 auto_increment 옵션에 의해 오름차순으로 자동 설정 (AUTO_INCREMENT)

- belong_branch_id: int 형식의 BRANCH의 Primary Key인 branch_id를 Foreign Key로 가져온 DEFAULT 값이 1인 Attribute

- Foreign Key로 가리키던 BRANCH의 branch_id가 DELETE 될 경우 DEFAULT 값인 1로 변경 (ON DELETE SET DEFAULT)

- Foreign Key로 가리키던 BRANCH의 branch_id가 UPDATE 될 경우 변경된 값에 따라감 (ON UPDATE CASCADE)

- treated_administrator_id: int 형식의 ADMINISTRATOR의 Primary Key인 administrator_id를 Foreign Key로 가져온 DEFAULT 값이 1인 Attribute

- Foreign Key로 가리키던 ADMINISTRATOR의 administrator_id가 DELETE 될 경우 DEFAULT 값인 1로 변경 (ON DELETE SET DEFAULT)

-Foreign Key로 가리키던 ADMINISTRATOR의 administrator_id가 UPDATE 될 경우 변경된 값에 따라감 (ON UPDATE CASCADE)

√. USER_NATIONALITY

```
49      cursor.execute('CREATE TABLE IF NOT EXISTS USER_NATIONALITY ('
50                     'user_id INT NOT NULL,'
51                     'nationality VARCHAR(56) NOT NULL,'
52                     'PRIMARY KEY (user_id, nationality),'
53                     'FOREIGN KEY (user_id) REFERENCES USER(user_id)'
54                     'ON DELETE CASCADE );')
```

- CREATE TABLE IF NOT EXISTS USER_NATIONALITY

√. USER_NATIONALITY라는 table이 존재하지 않는다면 생성

- user_id: int 형식의 USER의 Primary Key인 user_id를 Foreign Key로 가져온 Attribute

- Foreign Key로 가리키던 USER의 user_id가 DELETE 될 경우 같이 삭제됨 (ON DELETE CASCADE)

- nationality: 최대 56개의 char을 받을 수 있는 Attribute

- user_id와 nationality가 Primary Key를 이룸

√. USER_PHONE_NUMBER

```
56      cursor.execute('CREATE TABLE IF NOT EXISTS USER_PHONE_NUMBER ('
57                     'user_id INT NOT NULL,'
58                     'phone_number VARCHAR(13) NOT NULL,'
59                     'PRIMARY KEY (user_id, phone_number),'
60                     'FOREIGN KEY (user_id) REFERENCES USER(user_id)'
61                     'ON DELETE CASCADE );')
```

- CREATE TABLE IF NOT EXISTS USER_PHONE_NUMBER

√. USER_PHONE_NUMBER라는 table이 존재하지 않는다면 생성

- user_id: int 형식의 USER의 Primary Key인 user_id를 Foreign Key로 가져온 Attribute

- Foreign Key로 가리키던 USER의 user_id가 DELETE 될 경우 같이 삭제됨 (ON DELETE CASCADE)

- phone_number: 최대 13개의 char을 받을 수 있는 Attribute

- user_id와 phone_number가 Primary Key를 이룸

√. ACCOUNT

```
63      cursor.execute('CREATE TABLE IF NOT EXISTS ACCOUNT ('
64                     'account_number VARCHAR(17) NOT NULL,'
65                     'account_user_id INT NOT NULL,'
66                     'account_type VARCHAR(20) NOT NULL,'
67                     'account_opening_date DATE NOT NULL,'
68                     'account_balance INT NOT NULL DEFAULT 0,'
69                     'PRIMARY KEY (account_number),'
70                     'FOREIGN KEY (account_user_id) REFERENCES USER(user_id)'
71                     'ON DELETE CASCADE );')
```

- CREATE TABLE IF NOT EXISTS ACCOUNT

√. ACCOUNT라는 table이 존재하지 않는다면 생성

- account_number: 최대 17개의 char을 받을 수 있는 Primary Key

- account_user_id: int 형식의 USER의 Primary Key인 user_id를 Foreign Key로 가져온 Attribute

- Foreign Key로 가리키던 USER의 user_id가 DELETE 될 경우 같이 삭제됨 (ON DELETE CASCADE)

- account_type: 최대 20개의 char을 받을 수 있는 Attribute

- account_opening_date: date 형식의 Attribute

- account_balance: int 형식의 DEFAULT 값이 0인 Attribute

√. TRANSACTION_BREAKDOWN

```
73      cursor.execute('CREATE TABLE IF NOT EXISTS TRANSACTION_BREAKDOWN ('
74                     'transaction_account_number VARCHAR(17) NOT NULL,'
75                     'transaction_index INT NOT NULL AUTO_INCREMENT,'
76                     'transaction_date DATE NOT NULL,'
77                     'transaction_type VARCHAR(20) NOT NULL,'
78                     'transaction_amount INT NOT NULL,'
79                     'PRIMARY KEY (transaction_index),'
80                     'FOREIGN KEY (transaction_account_number) REFERENCES ACCOUNT(account_number)'
81                     'ON DELETE CASCADE );')
```

- CREATE TABLE IF NOT EXISTS TRANSACTION_BREAKDOWN

√. TRANSACTION_BREAKDOWN라는 table이 존재하지 않는다면 생성

- transaction_account_number: 최대 17개의 char을 받을 수 있는 ACCOUNT의 Primary Key인 account_number를 Foreign Key로 가져온 Attribute

- Foreign Key로 가리키던 ACCOUNT의 account_number가 DELETE 될 경우 같이 삭제됨 (ON DELETE CASCADE)

- transaction_index: int 형식의 TRANSACTION_BREAKDOWN table의 Primary Key

- INSERT시, transaction_index를 제외한 attribute을 입력하면 transaction_index는 auto_increment 옵션에 의해 오름차순으로 자동 설정 (AUTO_INCREMENT)

- transaction_date: date 형식의 Attribute

- transaction_type: 최대 20개의 char을 받을 수 있는 Attribute

- transaction_amount: int 형식의 Attribute

√. ADMINISTRATION_BREAKDOWN

```
90      cursor.execute('CREATE TABLE IF NOT EXISTS ADMINISTRATION_BREAKDOWN ('
91                     'administrator_id INT NOT NULL,'
92                     'administration_index INT NOT NULL AUTO_INCREMENT,'
93                     'administration_date DATE NOT NULL,'
94                     'treated_account_number VARCHAR(17) NOT NULL,'
95                     'administration_type VARCHAR(20) NOT NULL,'
96                     'PRIMARY KEY (administration_index),'
97                     'FOREIGN KEY (treated_account_number) REFERENCES ACCOUNT(account_number) '
98                     'ON DELETE CASCADE, '
99                     'FOREIGN KEY (administrator_id) REFERENCES ADMINISTRATOR(administrator_id) '
100                    'ON DELETE CASCADE);')
101     connection.commit()
```

- CREATE TABLE IF NOT EXISTS ADMINISTRATION_BREAKDOWN

√. ADMINISTRATION_BREAKDOWN라는 table이 존재하지 않는다면 생성

- administrator_id: int 형식의 ADMINISTRATOR의 Primary Key인 administrator_id를 Foreign Key로 가져온 Attribute

- Foreign Key로 가리키던 ADMINISTRATOR의 administrator_id가 DELETE 될 경우 같이 삭제됨 (ON DELETE CASCADE)

- administration_index: int 형식의 ADMINISTRATION_BREAKDOWN table의 Primary Key

- INSERT시, administration_index를 제외한 attribute을 입력하면 administration_index는 auto_increment 옵션에 의해 오름차순으로 자동 설정 (AUTO_INCREMENT)

- administration_date: date 형식의 Attribute

- treated_account_number: 최대 17개의 char을 받을 수 있는 ACCOUNT의 Primary Key인 account_number를 Foreign Key로 가져온 Attribute

- Foreign Key로 가리키던 ACCOUNT의 account_number가 DELETE 될 경우 같이 삭제됨 (ON DELETE CASCADE)

- administration_type: 최대 20개의 char을 받을 수 있는 Attribute

- user_interface()

```python
def user_interface():
    print()
    print("==============================")
    print("User Interface")
    print("0.Return to Previous Menu")
    print("1.Deposit/Withdrawal")
    print("2.Check Deposit/Withdrawal Details")
    print("3.User Registration")
    print("4.User Deletion")
    print("5.User Info Change")
    print("6.Account Registration")
    print("7.Account Deletion")
    print("==============================")
    print("Input: ", end='')
    user_command = int(input())
```

0. Return to Previous Menu

```python
    if user_command == 0:
        return
```

- 이전 메뉴로 돌아감

1. Deposit/Withdrawal

```python
elif user_command == 1:  # Deposit/Withdrawal
    account_number = input("Please enter your account number: ")
    cursor.execute("SELECT * FROM account WHERE account_number=%s;", account_number)
    result = cursor.fetchone()
    if result is None:
        print("There's no account for that account number! Please try again")
        return

    tr_type = input("Please enter your transaction type(D:Deposit/W:Withdrawal): ")
    if tr_type != "D" and tr_type != "W":
        print("Please enter one of D or P")
        return

    tr_amount = int(input("Please enter your transaction amount: "))

    tr_date = datetime.today().strftime('%Y-%m-%d')

    if tr_type == "D":
        cursor.execute("UPDATE account SET account_balance = account_balance + %s WHERE account_number = %s",
                       (tr_amount, account_number))
        cursor.execute("INSERT INTO "
                       "TRANSACTION_BREAKDOWN"
                       "(transaction_account_number,transaction_date,transaction_type,transaction_amount) "
                       "values(%s,%s,\"Deposit\",%s)", (account_number, tr_date, tr_amount))
```

```python
        print("Deposit Successful!")
    elif tr_type == "W":
        cursor.execute("SELECT account_balance FROM account WHERE account_number=%s", account_number)
        result = cursor.fetchone()
        balance = result[0]
        if balance < tr_amount:
            print("The withdrawal amount is higher than the balance")
            print("This account balance is " + str(balance))
            return
        cursor.execute("UPDATE account SET account_balance = account_balance - %s WHERE account_number = %s",
                       (tr_amount, account_number))
        cursor.execute("INSERT INTO "
                       "TRANSACTION_BREAKDOWN"
                       "(transaction_account_number,transaction_date,transaction_type,transaction_amount) "
                       "values(%s,%s,\"Withdrawal\",%s)", (account_number, tr_date, tr_amount))
        print("Withdrawal Successful!")
```

- 거래를 진행할 account_number 입력 받음

- "SELECT * FROM account WHERE account_number=%s", account_number

√. ACCOUNT table에서 입력받은 account_number가 존재하는지 select

- select 결과 없을 시에, 문구와 함께 return

- tr_type을 입력 받음

- tr_type이 "D"나 "W"가 아닐 경우 입력 실패 문구와 함께 return

- tr_amount 입력 받음

- tr_date는 datetime 모듈의 함수를 이용해 오늘의 날짜로 지정

- tr_type == "D": Deposit

- "UPDATE account SET account_balance = account_balance + %s WHERE account_number=%s", (tr_amount, account_number)

√. ACCOUNT table에서 account_number에 해당하는 tuple의 account_balance를 tr_amount만큼 증가

- "INSERT INTO TRANSACTION_BREAKDOWN (transaction_account_number,transaction_date,transaction_type,transaction_amount)

values(%s,%s,Deposit,%s)", (account_number, tr_date, tr_amount)

√. 거래에 관련된 value들을 TRANSACTION_BREAKDOWN에 insert

- tr_type == "W": Withdrawal

- "SELECT account_balance FROM account WHERE account_number=%s", account_number

√. ACCOUNT table에서 해당하는 account_number의 account_balance select

- 출금액이 잔액보다 높을 경우 출금 실패 문구와 함께 return

- "UPDATE account SET account_balance = account_balance - %s WHERE account_number=%s", (tr_amount, account_number)

√. ACCOUNT table에서 account_number에 해당하는 tuple의 account_balance를 tr_amount만큼 감소

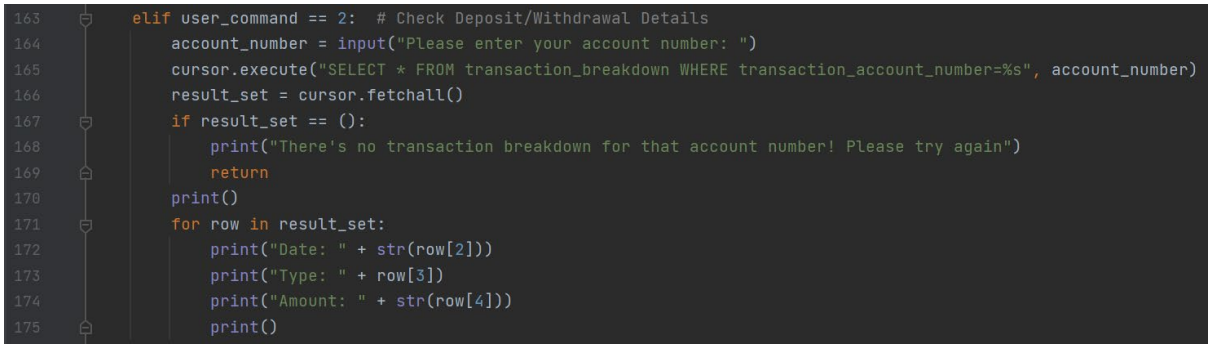
- "INSERT INTO TRANSACTION_BREAKDOWN

(transaction_account_number,transaction_date,transaction_type,transaction_amount)

values(%s,%s,Withdrawal,%s)", (account_number, tr_date, tr_amount)

√. 거래에 관련된 value들을 TRANSACTION_BREAKDOWN에 insert

2. Check Deposit/Withdrawal Details



```
163     elif user_command == 2:  # Check Deposit/Withdrawal Details
164         account_number = input("Please enter your account number: ")
165         cursor.execute("SELECT * FROM transaction_breakdown WHERE transaction_account_number=%s", account_number)
166         result_set = cursor.fetchall()
167         if result_set == ():
168             print("There's no transaction breakdown for that account number! Please try again")
169             return
170         print()
171         for row in result_set:
172             print("Date: " + str(row[2]))
173             print("Type: " + row[3])
174             print("Amount: " + str(row[4]))
175             print()
```

- 검색을 원하는 account_number 입력 받음

- "SELECT * FROM transaction_breakdown WHERE transaction_account_number=%s", account_number

√. TRANSACTION_BREAKDOWN table에서 검색을 원하는 account_number에 해당하는 tuple select

- 입력받은 account_number에 해당하는 tuple이 1개도 없다면 문구와 함께 return

- 입력받은 account_number에 대한 tuple이 존재하면 그 내역들을 출력

3. User Registration

```
177    elif user_command == 3:  # User Registration
178        user_name = input("Enter user's name you want to register(Up to 20 letters): ")
179        if len(user_name) > 20:
180            print("You can enter up to 20 letters of your name. Please try again")
181            return
182        user_ssn = input("Enter user's ssn you want to register(ex.990101-1234567): ")
183        if len(user_ssn) != 14 or user_ssn[6] != "-":
184            print("You should enter your ssn according to the style. Please try again.(ex.990101-1234567)")
185            return
186        user_address = input("Enter user's address you want to register(ex.Seoul): ")
187        if len(user_address) > 20:
188            print("You can enter up to 20 letters of your address. Please try again.")
189            return
190
191        user_branch = int(input("Enter user's branch id you want to register: "))
192        cursor.execute("SELECT * FROM branch WHERE branch_id=%s;", user_branch)
193        result = cursor.fetchone()
194        if result is None:
195            print("There's no branch for that id! Please try again.")
196            return
197
198        user_administrator_id = int(input("Enter the id of administrator manage the user: "))
199        cursor.execute("SELECT * FROM administrator WHERE administrator_id=%s", user_administrator_id)
```

```
200        result1 = cursor.fetchone()
201        if result1 is None:
202            print("There's no administrator for that id! Please try again.")
203            return
204
205        print("Enter user's nationalities you want to register")
206        print("if more than two, please enter them all with comma")
207        print("ex.Korea,USA")
208        user_nationalities = input("nationality: ")
209        nationalities_list = user_nationalities.split(",")
210        for n in nationalities_list:
211            if len(n) > 56:
212                print("You can enter up to 56 letters of your nationality. Please try again.")
213                return
214
215        print("Enter user's phone numbers you want to register")
216        print("if more than two, please enter them all with comma")
217        print("ex.010-1234-5678,010-2345-6789")
218        user_phone_numbers = input("phone number: ")
219        phone_numbers_list = user_phone_numbers.split(",")
220        for p in phone_numbers_list:
221            if len(p) != 13 or p[3] != "-" or p[8] != "-":
222                print("You should enter your phone number according to the style. Please try again.(ex.010-1234-5678)")
223                return
```

```
225        cursor.execute("INSERT INTO "
226                       "USER(user_name,user_ssn,user_address,belong_branch_id,treated_administrator_id) "
227                       "values(%s,%s,%s,%s,%s)",
228                       (user_name, user_ssn, user_address, user_branch, user_administrator_id))
229        connection.commit()
230
231        # 방금 register한 user의 id 탐색
232        cursor.execute("SELECT MAX(user_id) AS registered_id FROM user")
233        result = cursor.fetchone()
234        user_id = result[0]
235
236        for nation in nationalities_list:
237            cursor.execute("INSERT INTO USER_NATIONALITY values(%s,%s)", (user_id, nation))
238
239        for num in phone_numbers_list:
240            cursor.execute("INSERT INTO USER_PHONE_NUMBER values(%s,%s)", (user_id, num))
241
242        print("User registration Successful!")
```

- user_name 입력받음(문자열의 길이가 20보다 크면 문구와 함께 return)

- user_ssn 입력받음(문자열의 길이가 14가 아니거나 user_ssn[6]이 "-"가 아니라면 문구와 함께 return)

- user_address 입력받음(문자열의 길이가 20보다 크면 문구와 함께 return)

- user_branch 입력받음

- "SELECT * FROM branch WHERE branch_id=%s", user_branch

√. BRANCH table에서 입력받은 user_branch가 존재하는지 select

- 입력받은 user_branch가 존재하지 않는다면 문구와 함께 return

- user_administration_id 입력받음

- "SELECT * FROM administrator WHERE administrator_id=%s", user_administration_id

√. ADMINISTRATOR table에서 입력받은 user_administration_id가 존재하는지 select

- 입력받은 user_administration_id가 존재하지 않는다면 문구와 함께 return

- user_nationalities 입력받음(여러 개라면 comma(,)로 이루어진 문자열을 입력 받은 후 split 하여 nationalities_list에 저장 후, 각 nationality 문자열의 길이가 56보다 크면 문구와 함께 return)

- user_phone_numbers 입력받음(여러 개라면 comma(,)로 이루어진 문자열을 입력 받은 후 split 하여 phon_numbers_list에 저장 후, 각 phone_number 문자열의 길이가 13이 아니거나 phone_number[3], phone_number[8]이 "-"가 아니라면 문구와 함께 return)

- "INSERT INTO USER(user_name,user_ssn,user_address,belong_branch_id,treated_administrator_id) values(%s,%s,%s,%s,%s)", (user_name, user_ssn, user_address, user_branch, user_administration_id)

√. 입력 받은 정보들을 USER에 insert

- SELECT MAX(user_id) AS registered_id FROM user

√. USER table의 user_id 중 가장 큰 값, 즉 방금 insert한 tuple의 user_id를 select(user_id는 auto_increment 옵션으로 인해 가장 나중에 만들어진 tuple의 id가 제일 크다.)

- "INSERT INTO USER_NATIONALITY values(%s,%s)", (user_id, nation)

- "INSERT INTO USER_PHONE_NUMBER values(%s,%s)", (user_id, num)

√. USER_NATIONALITY와 USER_PHONE_NUMBER에 입력받은 value들을 insert

4. User Deletion

```
244     elif user_command == 4:  # User Deletion
245         user_id = int(input("Enter user's id you want to delete: "))
246         cursor.execute("SELECT user_id FROM user WHERE user_id=%s", user_id)
247         result = cursor.fetchone()
248         if result is None:
249             print("There is no user for that user id!")
250         else:
251             cursor.execute("SELECT account_balance FROM account WHERE account_user_id=%s", user_id)
252             result_set = cursor.fetchall()
253             if result_set != ():
254                 for row in result_set:
255                     if row[0] != 0:
256                         print("This user have some accounts whose balance is not 0")
257                         print("User can be deleted only when all of having account's balance are 0")
258                         return
259             cursor.execute("DELETE FROM user WHERE user_id=%s;", user_id)
260             print("user's id for \"" + str(user_id) + "\" has been deleted")
```

- user_id 입력 받음

- "SELECT user_id FROM user WHERE user_id=%s", user_id

√. USER table에 해당하는 user_id가 존재하는지 select

- 입력받은 user_id가 USER table에 존재하지 않는다면 문구와 함께 return

- "SELECT account_balance FROM account WHERE account_user_id=%s", user_id

√. 해당 USER의 ACCOUNT들의 balance를 모두 select

- balance가 0이 아닌 ACCOUNT가 하나라도 존재한다면 문구와 함께 return

- "DELETE FROM user WHERE user_id=%s", user_id

√. USER table에서 입력 받은 user_id에 해당하는 tuple 삭제

5. User Info Change

```
262     elif user_command == 5:  # User Info Change
263         user_id = int(input("Enter user's id you want to change: "))
264         cursor.execute("SELECT user_id FROM user WHERE user_id=%s", user_id)
265         result = cursor.fetchone()
266         if result is None:
267             print("There is no user for that user id!")
268         else:
269             category = input("Which info you want to change?"
270                              "(N:Name, A:Address, B:Branch, Admin:Administrator, Nation:Nationality, P:PhoneNumber): ")
```

- user_id 입력 받음

- "SELECT user_id FROM user WHERE user_id=%s", user_id

√. USER table에 해당하는 user_id가 존재하는지 select

- 입력 받은 user_id가 USER table에 존재하지 않는다면 문구와 함께 return

- 정보 변경을 원하는 category 입력 받고 category에 따라 정보 변경 진행

```
271             if category == "N":
272                 new_name = input("Enter new name(Up to 20 letters): ")
273                 if len(new_name) > 20:
274                     print("You can enter up to 20 letters of your name. Please try again.")
275                     return
276                 cursor.execute("UPDATE user SET user_name=%s WHERE user_id=%s",
277                                (new_name, user_id))
278                 print("Name change successful!")
```

- Category == "N": user_name 변경

- new_name 입력 받음(문자열의 길이가 20보다 크면 문구와 함께 return)

- "UPDATE user SET user_name=%s WHERE user_id=%s", (new_name, user_id)

√. USER table에서 입력 받은 user_id에 해당하는 tuple의 user_name 변경

```
280             elif category == "A":
281                 new_address = input("Enter new address(ex.Seoul): ")
282                 if len(new_address) > 20:
283                     print("You can enter up to 20 letters of your address. Please try again.")
284                     return
285                 cursor.execute("UPDATE user SET user_address=%s WHERE user_id=%s",
286                                (new_address, user_id))
287                 print("Address change successful!")
```

- Category == "A": user_address 변경

- new_address 입력 받음(문자열의 길이가 20보다 크면 문구와 함께 return)

- "UPDATE user SET user_address=%s WHERE user_id=%s", (new_address, user_id)

√. USER table에서 입력 받은 user_id에 해당하는 tuple의 user_address 변경

```
289         elif category == "B":
290             new_branch = int(input("Enter new branch_id: "))
291             cursor.execute("SELECT * FROM branch WHERE branch_id = %s", new_branch)
292             result = cursor.fetchone()
293             if result is None:
294                 print("There is no such branch id!")
295                 return
296             cursor.execute("UPDATE user SET belong_branch_id=%s WHERE user_id=%s",
297                            (new_branch, user_id))
298             print("Branch change successful!")
```

- Category == "B": belong_branch_id 변경

- new_branch 입력 받음

- "SELECT * FROM branch WHERE branch_id=%s", new_branch

√. BRANCH table에서 입력 받은 new_branch가 존재하는지 select

- 입력 받은 user_branch가 존재하지 않는다면 문구와 함께 return

- "UPDATE user SET belong_branch_id=%s WHERE user_id=%s", (new_branch, user_id)

√. USER table에서 입력 받은 user_id에 해당하는 tuple의 belong_branch_id 변경

```
300         elif category == "Admin":
301             new_admin = int(input("Enter new administrator_id: "))
302             cursor.execute("SELECT * FROM administrator WHERE administrator_id = %s", new_admin)
303             result = cursor.fetchone()
304             if result is None:
305                 print("There is no such administrator id!")
306                 return
307             cursor.execute("UPDATE user SET treated_administrator_id=%s WHERE user_id=%s",
308                            (new_admin, user_id))
309             print("Administrator change successful!")
```

- Category == "Admin": treated_administrator_id 변경

- new_admin 입력 받음

- "SELECT * FROM administrator WHERE administrator_id=%s", new_admin

√. ADMINISTRATOR table에서 입력 받은 new_admin이 존재하는지 select

- 입력 받은 new_admin이 존재하지 않는다면 문구와 함께 return

- "UPDATE user SET treated_administratorh_id=%s WHERE user_id=%s",(new_admin, user_id)

√. USER table에서 입력 받은 user_id에 해당하는 tuple의 treated_administrator_id 변경

```
311         elif category == "Nation":
312             nation_command = input("Enter the command you want(Add:A, Delete: D): ")
313             if nation_command == "A":
314                 new_nation = input("Enter your new nationality: ")
315                 if len(new_nation) > 56:
316                     print("You can enter up to 56 letters of your nationality. Please try again.")
317                     return
318                 cursor.execute("INSERT INTO user_nationality values(%s,%s)", (user_id, new_nation))
319                 print("Nationality addition successful!")
320             elif nation_command == "D":
321                 origin_nation = input("Enter your existing nationality: ")
322                 cursor.execute("SELECT nationality FROM user_nationality WHERE user_id=%s and nationality=%s",
323                                (user_id, origin_nation))
324                 result = cursor.fetchone()
325                 if result is None:
326                     print("There is no such nationality!")
327                     return
328                 cursor.execute("DELETE FROM user_nationality WHERE user_id=%s and nationality=%s",
329                                (user_id, origin_nation))
330                 print("Nationality deletion successful!")
331             else:
332                 print("Please enter A or D")
```

- Category == "Nation": nationality 추가 or 삭제

- nation_command 입력 받음

- nation_command == "A": nationality 추가

- new_nation 입력 받음(문자열의 길이가 56보다 크면 문구와 함께 return)

- "INSERT INTO user_nationality values(%s,%s)", (user_id, new_nation)

√. USER_NATIONALITY table에 입력 받은 user_id와 nationality tuple insert

- nation_command == "D": nationality 삭제

- origin_nation 입력 받음

- "SELECT nationality FROM user_nationality WHERE user_id=%s and nationality=%s",

(user_id, origin_nation)

√. USER_NATIONALITY table에서 해당 user_id와 nationality를 가진 tuple select

- user_id에 대해 입력 받은 origin_nation이 존재하지 않는다면 문구와 함께 return

- "DELETE FROM user_nationality WHERE user_id=%s and nationality=%s", (user_id, origin_nation)

√. USER_NATIONALITY table에서 해당 user_id와 nationality를 가진 tuple delete

```
334         elif category == "P":
335             phone_command = input("Enter the command you want(Add:A, Delete: D): ")
336             if phone_command == "A":
337                 new_phone = input("Enter your new phone number(ex.010-1234-5678): ")
338                 if len(new_phone) != 13 or new_phone[3] != "-" or new_phone[8] != "-":
339                     print("You should enter your phone number according to the style. "
340                           "Please try again.(ex.010-1234-5678)")
341                     return
342                 cursor.execute("INSERT INTO user_phone_number values(%s,%s)", (user_id, new_phone))
343                 print("Phone number addition successful!")
344             elif phone_command == "D":
345                 origin_phone = input("Enter your existing phone number: ")
346                 cursor.execute("SELECT phone_number FROM user_phone_number WHERE user_id=%s and phone_number=%s",
347                                (user_id, origin_phone))
348                 result = cursor.fetchone()
349                 if result is None:
350                     print("There is no such phone number!")
351                     return
352                 cursor.execute("DELETE FROM user_phone_number WHERE user_id=%s and phone_number=%s",
353                                (user_id, origin_phone))
354                 print("Phone number deletion successful!")
355             else:
356                 print("Please enter A or D")
```

- Category == "P": phone_number 추가 or 삭제

- phone_command 입력 받음

- phone_command == "A": phone_number 추가

- new_phone 입력받음(문자열의 길이가 13이 아니거나 phone_number[3], phone_number[8]이 "-"가 아니라면 문구와 함께 return)

- "INSERT INTO user_phone_number values(%s,%s)", (user_id, new_phone)

√. USER_PHONE_NUMBER table에 입력 받은 user_id와 phone_number tuple insert

- phone_command == "D": phone_number 삭제

- origin_phone 입력 받음

- "SELECT phone_number FROM user_phone_number

WHERE user_id=%s and phone_number=%s", (user_id, origin_phone)

√. USER_PHONE_NUMBER table에서 해당 user_id와 phone_number를 가진 tuple select

- user_id에 대해 입력 받은 origin_phone이 존재하지 않는다면 문구와 함께 return

- "DELETE FROM user_phone_number WHERE user_id=%s and phone_number=%s", (user_id,origin_phone)

√. USER_PHONE_NUMBER table에서 해당 user_id와 phone_number를 가진 tuple delete

6. Account Registration

```
361     elif user_command == 6:  # Account Registration
362         account_number = input("Enter account number you want to register(Up to 17 letters): ")
363         if len(account_number) > 17:
364             print("You can enter up to 17 letters of your account number. Please try again.")
365             return
366         cursor.execute("SELECT * FROM account WHERE account_number = %s;", account_number)
367         account_result = cursor.fetchone()
368         if account_result is not None:
369             print("That account number already exists")
370             return
371 
372         user_id = int(input("Enter user's id of the account you want to register: "))
373         cursor.execute("SELECT * FROM user WHERE user_id=%s;", user_id)
374         result = cursor.fetchone()
375         if result is None:
376             print("There's no user for that id! Please try again")
377             return
```

- account_number 입력 받음(문자열의 길이가 17보다 크다면 문구와 함께 return)

- "SELECT * FROM account WHERE account_number=%s", account_number

√. ACCOUNT table에서 해당 account_number를 가진 tuple select

- 입력 받은 계좌번호가 이미 존재한다면 문구와 함께 return

- user_id 입력 받음

- "SELECT * FROM user WHERE user_id=%s", user_id

√. USER table에서 해당 user_id를 가진 tuple select

- 입력 받은 user_id가 존재하지 않는다면 문구와 함께 return

```
379         # 계좌 최대 3개까지 개설
380         cursor.execute("SELECT COUNT(*) FROM account WHERE account_user_id=%s", user_id)
381         count = cursor.fetchone()
382         if count[0] >= 3:
383             print("User cannot have more thant 3 accounts")
384             return
385 
386         account_type = ""
387         type_initial = input("Enter the type of account want to register(B:Bankbook, I:Installment saving, C:CMA):")
388         if type_initial == "B":
389             account_type = "Bankbook"
390         elif type_initial == "I":
391             account_type = "Installment"
392         elif type_initial == "C":
393             account_type = "CMA"
394         else:
395             print("Please enter one of B, I, C")
396             return
397 
398         account_opening_date = datetime.today().strftime('%Y-%m-%d')
399 
400         cursor.execute("INSERT INTO "
401                        "ACCOUNT values(%s,%s,%s,%s,0)", (account_number, user_id, account_type, account_opening_date))
```

- “SELECT COUNT(*) FROM account WHERE account_user_id=%s”, user_id

√. ACCOUNT table에서 account_user_id가 입력 받은 user_id와 같은 tuple의 갯수 select

- count가 3 이상이라면 문구와 함께 return(한 명의 사용자는 계좌를 최대 3개까지 개설할 수 있다.)

- account_type 입력 받음(입력 받은 initial에 따라 실제 table에 insert할 문구로 변환)

- account_opening_date는 datetime 모듈의 함수를 이용해 오늘의 날짜로 지정

- “INSERT INTO ACCOUNT values(%s,%s,%s,%s,0)”,

(account_number, user_id, account_type, account_opening_date)

√. ACCOUNT table에 입력 받은 정보들을 insert

7. Account Deletion

```
405     elif user_command == 7:  # Account Deletion
406         account_number = input("Enter account number you want to delete: ")
407         cursor.execute("SELECT account_balance FROM account WHERE account_number=%s", account_number)
408         result = cursor.fetchone()
409         if result is None:
410             print("There is no account for that account number!")
411         else:
412             balance = result[0]
413             if balance != 0:
414                 print("You can delete the account with a balance of 0")
415                 return
416             else:
417                 cursor.execute("DELETE FROM account WHERE account_number=%s;", account_number)
418                 print("account for \"" + account_number + "\" has been deleted")
```

- account_number 입력 받음

- “SELECT account_balance FROM account WHERE account_number=%s”, account_number

√. ACCOUNT table에서 해당 account_number를 가진 tuple의 account_balance select

- 해당 tuple이 존재하지 않는다면 문구와 함께 return

- 존재하는 tuple의 account_balance가 0이 아니라면 문구와 함께 return(사용자는 잔액이 0원인 계좌만 삭제할 수 있다.)

- “DELETE FROM account WHERE account_number=%s”, account_number

√. ACCOUNT table에서 입력받은 account_number에 해당하는 tuple delete

- admin_interface()

```
426     def admin_interface():
427         print()
428         print("==============================")
429         print("Administrator Interface")
430         print("0.Return to Previous Menu")
431         print("1.Administer Account")
432         print("2.Check Administration Details")
433         print("3.User Search")
434         print("4.Account Search")
435         print("5.Administrator Registration")
436         print("6.Administrator Deletion")
437         print("7.Administrator Info Change")
438         print("8.Administrator Search")
439         print("9.Branch Registration")
440         print("10.Branch Deletion")
441         print("11.Branch Info Change")
442         print("12.Branch Search")
443         print("==============================")
444         print("Input: ", end='')
```

0. Return to Previous Menu

```
446         if admin_command == 0:
447             return
```

- 이전 메뉴로 돌아감

1. Administer Account

```
449     elif admin_command == 1:  # Administer Account
450         administrator_id = int(input("PLease enter your administrator id: "))
451         cursor.execute("SELECT * FROM administrator WHERE administrator_id=%s", administrator_id)
452         result = cursor.fetchone()
453         if result is None:
454             print("There's no administrator for that administrator id! Please try again")
455 
456         account_number = input("Please enter the account number you want to administer: ")
457         cursor.execute("SELECT * FROM account WHERE account_number=%s;", account_number)
458         result1 = cursor.fetchone()
459         if result1 is None:
460             print("There's no account for that account number! Please try again")
461             return
462         user_id = result1[1]
463         cursor.execute("SELECT treated_administrator_id FROM user WHERE user_id=%s", user_id)
464         result2 = cursor.fetchone()
465         user_admin_id = result2[0]
466         if administrator_id != user_admin_id:
467             print("Administrators can only administer accounts assigned to them")
468             return
469 
470         admin_date = datetime.today().strftime('%Y-%m-%d')
```

- administrator_id 입력 받음

- "SELECT * FROM administrator WHERE administrator_id=%s", administrator_id

√. ADMINISTRATOR table에서 해당하는 administrator_id를 가진 tuple select

- 해당 tuple이 존재하지 않는다면 문구와 함께 return

- account_number 입력 받음

- "SELECT * FROM account WHERE account_number=%s", account_number

√. ACCOUNT table에서 입력받은 account_number가 존재하는지 select

- account_number가 존재하지 않을 시에, 문구와 함께 return

- "SELECT treated_administrator_id FROM user WHERE user_id=%s", user_id

√. USER table에서 user_id에 해당하는 tuple의 treated_administrator_id select

- 관리를 진행하고자 하는 ADMINISTRATOR의 id와 관리 받는 ACCOUNT의 USER의 treated_administrator_id가 다르다면 문구와 함께 return(관리자는 자신이 담당하는 계좌만 관리할 수 있다.)

- admin_date는 datetime 모듈의 함수를 이용해 오늘의 날짜로 지정

```
472         admin_type = ""
473         type_initial = input("Enter the type of administration"
474                              "(C:Check Issue, A:Account Bookkeeping:, P: Pattern Analysis): ")
475         if type_initial == "C":
476             admin_type = "Check Issue"
477         elif type_initial == "A":
478             admin_type = "Account Bookkeeping"
479         elif type_initial == "P":
480             admin_type = "Pattern Analysis"
481         else:
482             print("Please enter one of C, A, P")
483             return
484
485         cursor.execute("INSERT INTO administration_breakdown"
486                        "(administrator_id,administration_date,treated_account_number,administration_type) "
487                        "values(%s,%s,%s,%s)", (administrator_id, admin_date, account_number, admin_type))
488         print("Administer Successful!")
```

- type_initial 입력 받음(입력 받은 initial에 따라 실제 table에 insert할 문구로 변환)

- "INSERT INTO administration_breakdown

(administrator_id, administration_date, treated_account_number, administration_type)

values(%s,%s,%s,%s)", (administrator_id, admin_date, account_number, admin_type)

√. ADMINISTRATION_BREAKDOWN table에 입력 받은 정보 insert

2. Check Administration Details

```
490     elif admin_command == 2:  # Check Administration Details
491         detail_command = input("Which search option you want?(Admin, Account): ")
492         if detail_command == "Admin":
493             administrator_id = int(input("Please enter administrator id: "))
494             cursor.execute("SELECT * FROM administration_breakdown WHERE administrator_id=%s", administrator_id)
495             result_set = cursor.fetchall()
496             if result_set == ():
497                 print("There's no administration breakdown for that administrator id! Please try again")
498                 return
499             print()
500             for row in result_set:
501                 print("Date: " + str(row[2]))
502                 print("Administrator ID: " + str(row[0]))
503                 print("Account Number: " + row[3])
504                 print("Type: " + row[4])
505                 print()
```

- detail_command 입력 받음

- detail_command == "Admin": administrator_id로 검색

- 검색을 원하는 administrator_id 입력 받음

- "SELECT * FROM administration_breakdown WHERE administrator_id=%s", administrator_id

√. administration_breakdown table에서 검색을 원하는 administrator_id에 해당하는 tuple을 select

- 입력받은 administrator_id에 해당하는 tuple이 1개도 없다면 문구와 함께 return

- 입력받은 administrator_id에 대한 tuple이 존재하면 그 내역들을 출력

```
507         elif detail_command == "Account":
508             account_number = input("Please enter account number: ")
509             cursor.execute("SELECT * FROM administration_breakdown WHERE treated_account_number=%s", account_number)
510             result_set = cursor.fetchall()
511             if result_set == ():
512                 print("There's no administration breakdown for that account number! Please try again")
513                 return
514             print()
515             for row in result_set:
516                 print("Date: " + str(row[2]))
517                 print("Administrator ID: " + str(row[0]))
518                 print("Account Number: " + row[3])
519                 print("Type: " + row[4])
520                 print()
```

- detail_command == "Account": account_number로 검색

- 검색을 원하는 account_number 입력 받음

- "SELECT * FROM administration_breakdown WHERE treated_account_number=%s", account_number

√. administration_breakdown table에서 검색을 원하는 treated_account_number에 해당하는 tuple을 select

- 입력받은 account_number에 해당하는 tuple이 1개도 없다면 문구와 함께 return

- 입력받은 account_number에 대한 tuple이 존재하면 그 내역들을 출력

3. User Search

```
524     elif admin_command == 3:  # User Search
525         search_command = input("Which search option do you want?(A:All, S:Specific): ")
526         if search_command == "A":
527             cursor.execute("SELECT * FROM user;")
528             result_set = cursor.fetchall()
529             if result_set == ():
530                 print("There's no users")
531                 return
532             for row in result_set:
533                 print("Name: " + row[0])
534                 print("Ssn: " + row[1])
535                 print("Address: " + row[2])
536                 print("User ID: " + str(row[3]))
537                 print("Belonging branch id: " + str(row[4]))
538                 print("Treating administrator id: " + str(row[5]))
539                 user_id = row[3]
540 
541                 cursor.execute("SELECT nationality FROM user_nationality WHERE user_id=%s", user_id)
542                 result_nationalities = cursor.fetchall()
543                 print("Nationalities: ", end='')
544                 if result_nationalities == ():
545                     print()
546                 for nation in result_nationalities:
547                     if nation is result_nationalities[-1]:
548                         print(nation[0])
549                     else:
550                         print(nation[0] + ", ", end='')
551 
552                 cursor.execute("SELECT phone_number FROM user_phone_number WHERE user_id=%s", user_id)
553                 result_phones = cursor.fetchall()
554                 print("Phone numbers: ", end='')
555                 if result_phones == ():
556                     print()
557                 for num in result_phones:
558                     if num is result_phones[-1]:
559                         print(num[0])
560                     else:
561                         print(num[0] + ", ", end='')
563                 cursor.execute("SELECT account_number FROM account WHERE account_user_id=%s", user_id)
564                 result_accounts = cursor.fetchall()
565                 print("Account numbers: ", end='')
566                 if result_accounts == ():
567                     print()
568                 for num in result_accounts:
569                     if num is result_accounts[-1]:
570                         print(num[0])
571                     else:
572                         print(num[0] + ", ", end='')
```

- search_command 입력 받음

- search_command == "A": 모든 USER tuple들 출력

- "SELECT * FROM user;"

√. USER table의 모든 tuple들 select

- tuple이 1개도 없다면 문구와 함께 return

- tuple이 존재한다면 모든 tuple들에 대해 정보 출력

- "SELECT nationality FROM user_nationality WHERE user_id=%s", user_id

- "SELECT phone_number FROM user_phone_number WHERE user_id=%s", user_id

- "SELECT account_number FROM account WHERE account_user_id=%s", user_id

√. USER_NATIONALITY, USER_PHONE_NUMBER, ACCOUNT table에서 해당 user_id에 대한 tuple select

- tuple이 존재하지 않는다면 한 줄 띄우고 종료, 존재한다면 comma(,) 단위로 출력

```
575         elif search_command == "S":
576             user_id = int(input("Enter user's id you want to search: "))
577             cursor.execute("SELECT user_id FROM user WHERE user_id=%s", user_id)
578             result = cursor.fetchone()
579             if result is None:
580                 print("There is no user for that user id!")
581             else:
582                 cursor.execute("SELECT * FROM user WHERE user_id=%s", user_id)
583                 result = cursor.fetchone()
584 
585                 print("Name: " + result[0])
586                 print("Ssn: " + result[1])
587                 print("Address: " + result[2])
588                 print("User ID: " + str(result[3]))
589                 print("Belonging branch id: " + str(result[4]))
590                 print("Treating administrator id: " + str(result[5]))
591 
592                 cursor.execute("SELECT nationality FROM user_nationality WHERE user_id=%s", user_id)
593                 result_nationalities = cursor.fetchall()
594                 print("Nationalities: ", end='')
595                 if result_nationalities == ():
596                     print()
597                 for nation in result_nationalities:
598                     if nation is result_nationalities[-1]:
599                         print(nation[0])
```

```
            else:
                print(nation[0] + ", ", end='')

        cursor.execute("SELECT phone_number FROM user_phone_number WHERE user_id=%s", user_id)
        result_phones = cursor.fetchall()
        print("Phone numbers: ", end='')
        if result_phones == ():
            print()
        for num in result_phones:
            if num is result_phones[-1]:
                print(num[0])
            else:
                print(num[0] + ", ", end='')

        cursor.execute("SELECT account_number FROM account WHERE account_user_id=%s", user_id)
        result_accounts = cursor.fetchall()
        print("Account numbers: ", end='')
        if result_accounts == ():
            print()
        for num in result_accounts:
            if num is result_accounts[-1]:
                print(num[0])
            else:
                print(num[0] + ", ", end='')
```

- search_command == "S": 특정 USER tuple 출력

- user_id 입력 받음

- "SELECT user_id FROM user WHERE user_id=%s", user_id

√. USER table에서 해당하는 user_id를 가진 tuple의 user_id select

- 존재하지 않는다면 문구와 함께 return

- "SELECT * FROM user WHERE user_id=%s", user_id

√. USER table에서 해당하는 user_id를 가진 tuple select

- tuple에 대한 정보들 출력

- "SELECT nationality FROM user_nationality WHERE user_id=%s", user_id

- "SELECT phone_number FROM user_phone_number WHERE user_id=%s", user_id

- "SELECT account_number FROM account WHERE account_user_id=%s", user_id

√. USER_NATIONALITY, USER_PHONE_NUMBER, ACCOUNT table에서 해당 user_id에 대한 tuple select

- tuple이 존재하지 않는다면 한 줄 띄우고 종료, 존재한다면 comma(,) 단위로 출력

4. Account Search

```
628     elif admin_command == 4:  # Account Search
629         search_command = input("Which search option do you want?(A:All, S:Specific): ")
630         if search_command == "A":
631             cursor.execute("SELECT * FROM account;")
632             result_set = cursor.fetchall()
633             if result_set == ():
634                 print("There's no accounts")
635                 return
636             for row in result_set:
637                 print("Account number: " + row[0])
638                 print("Account user id: " + str(row[1]))
639                 print("Account type: " + row[2])
640                 print("Account opening date: " + str(row[3]))
641                 print("Account balance: " + str(row[4]))
```

- search_command 입력 받음

- search_command == "A": 모든 ACCOUNT tuple들 출력

- SELECT * FROM account;

√. ACCOUNT table의 모든 tuple들 select

- tuple이 1개도 없다면 문구와 함께 return

- tuple이 존재한다면 모든 tuple들에 대해 정보 출력

```
644         elif search_command == "S":
645             account_number = input("Enter account number you want to search: ")
646             cursor.execute("SELECT account_number FROM account WHERE account_number=%s", account_number)
647             result = cursor.fetchone()
648             if result is None:
649                 print("There is no account for that account_number!")
650             else:
651                 cursor.execute("SELECT * FROM account WHERE account_number=%s", account_number)
652                 result = cursor.fetchone()
653 
654                 print("Account number: " + result[0])
655                 print("Account user id: " + str(result[1]))
656                 print("Account type: " + result[2])
657                 print("Account opening date: " + str(result[3]))
658                 print("Account balance: " + str(result[4]))
```

- search_command == "S": 특정 ACCOUNT tuple 출력

- account_number 입력 받음

- "SELECT account_number FROM account WHERE account_number=%s", account_number

√. ACCOUNT table에서 해당하는 account_number 를 가진 tuple의 account_number select

- 존재하지 않는다면 문구와 함께 return

- “SELECT * FROM account WHERE account_number=%s”, account_number

√. ACCOUNT table에서 해당하는 account_number를 가진 tuple select

- tuple에 대한 정보들 출력

5. Administrator Registration

```
662     elif admin_command == 5:  # Administrator Registration
663         admin_name = input("Enter administrator's name you want to register(Up to 20 letters): ")
664         if len(admin_name) > 20:
665             print("You can enter up to 20 letters of your name. Please try again.")
666             return
667         admin_branch = int(input("Enter administrator's working branch id you want to register: "))
668         cursor.execute("SELECT * FROM branch WHERE branch_id=%s;", admin_branch)
669         result = cursor.fetchone()
670         if result is None:
671             print("There's no branch for that id! Please try again")
672             return
673         admin_birth = input("Enter administrator's birthdate you want to register(ex.2021-01-01): ")
674         if len(admin_birth) != 10 or admin_birth[4] != "-" or admin_birth[7] != "-":
675             print("You should enter your birth date according to the style. Please try again.(ex.2021-01-01)")
676             return
677         admin_address = input("Enter administrator's address you want to register(ex.Seoul): ")
678         if len(admin_address) > 20:
679             print("You can enter up to 20 letters of your address. Please try again.")
680             return
```

```
682         print("Enter administrator's phone numbers you want to register")
683         print("if more than two, please enter them all with comma")
684         print("ex.010-1234-5678,010-2345-6789")
685         admin_phone_numbers = input("phone number: ")
686         phone_numbers_list = admin_phone_numbers.split(",")
687         for p in phone_numbers_list:
688             if len(p) != 13 or p[3] != "-" or p[8] != "-":
689                 print("You should enter your phone number according to the style. Please try again.(ex.010-1234-5678)")
690                 return
691
692         cursor.execute("INSERT INTO "
693                        "ADMINISTRATOR(administrator_name,branch_id,birth_date,administrator_address) "
694                        "values(%s,%s,%s,%s)", (admin_name, admin_branch, admin_birth, admin_address))
695         connection.commit()
```

```
697           # 방금 register한 administrator의 id
698           cursor.execute("SELECT MAX(administrator_id) AS registered_id FROM administrator")
699           result = cursor.fetchone()
700           admin_id = result[0]
701
702           for num in phone_numbers_list:
703               cursor.execute("INSERT INTO ADMINISTRATOR_PHONE_NUMBER values(%s,%s)", (admin_id, num))
704
705           print("Administrator registration Successful!")
```

- admin_name 입력받음(문자열의 길이가 20보다 크면 문구와 함께 return)

- admin_branch 입력받음

- "SELECT * FROM branch WHERE branch_id=%s", admin_branch

√. BRANCH table에서 입력받은 admin_branch가 존재하는지 select

- 입력받은 admin_branch가 존재하지 않는다면 문구와 함께 return

- admin_birth 입력받음(문자열의 길이가 10이 아니거나 admin_birth[4]나 admin_birth[7]이 "-"가 아니라면 문구와 함께 return)

- admin_address 입력받음(문자열의 길이가 20보다 크면 문구와 함께 return)

- admin_phone_numbers 입력받음(여러 개라면 comma(,)로 이루어진 문자열을 입력 받은 후 split 하여 phon_numbers_list에 저장 후, 각 phone_number 문자열의 길이가 13이 아니거나 phone_number[3], phone_number[8]이 "-"가 아니라면 문구와 함께 return)

- "INSERT INTO ADMINISTRATOR

(administrator_name,branch_id,birth_date,administrator_address) values(%s,%s,%s,%s)",

(admin_name, admin_branch, admin_birth, admin_address)

√. 입력 받은 정보들을 ADMINISTRATOR에 insert

- SELECT MAX(administrator_id) AS registered_id FROM administrator

√. ADMINISTRATOR table의 administrator_id 중 가장 큰 값, 즉 방금 insert한 tuple의 administrator_id를 select(administrator_id는 auto_increment 옵션으로 인해 가장 나중에 만들어진 tuple의 id가 제일 크다.)

- "INSERT INTO ADMINISTRATOR_PHONE_NUMBER values(%s,%s)", (admin_id,num)

√. ADMINISTRATOR_PHONE_NUMBER에 입력받은 value들을 insert

6. Administrator Deletion

```
707     elif admin_command == 6:  # Administrator Deletion
708         admin_id = int(input("Enter administrator's id you want to delete: "))
709         cursor.execute("SELECT administrator_id FROM administrator WHERE administrator_id=%s", admin_id)
710         result = cursor.fetchone()
711         if result is None:
712             print("There is no administrator for that administrator_id!")
713         else:
714             cursor.execute("DELETE FROM administrator WHERE administrator_id=%s;", admin_id)
715             print("administrator's id for \"" + str(admin_id) + "\" has been deleted")
```

- admin_id 입력 받음

- "SELECT administrator_id FROM administrator WHERE administrator_id=%s", admin_id

√. ADMINISTRATOR table에서 해당하는 administrator_id에 대한 tuple select

- 존재하지 않을 시에 문구와 함께 return

- "DELETE FROM administrator WHERE administrator_id=%s", admin_id

√. ADMINISTRATOR table에서 해당하는 administrator_id에 대한 tuple delete

7. Administrator Info Change

```
717     elif admin_command == 7:  # Administrator Info Change
718         admin_id = int(input("Enter administrator's id you want to change: "))
719         cursor.execute("SELECT * FROM administrator WHERE administrator_id=%s", admin_id)
720         result = cursor.fetchone()
721         if result is None:
722             print("There is no administrator for that administrator_id!")
723         else:
724             category = input("Which info you want to change?(N:Name, B:Branch, A:Address, P:PhoneNumber):")
```

- admin_id 입력 받음

- "SELECT * FROM administrator WHERE administrator_id=%s", admin_id

√. ADMINISTRATOR table에 해당하는 administrator_id가 존재하는지 select

- 입력 받은 administrator_id가 ADMINISTRATOR table에 존재하지 않는다면 문구와 함께 return

- 정보 변경을 원하는 category 입력 받고 category에 따라 정보 변경 진행

```
725             if category == "N":
726                 new_name = input("Enter new name(Up to 20 letters): ")
727                 if len(new_name) > 20:
728                     print("You can enter up to 20 letters of your name. Please try again.")
729                     return
730                 cursor.execute("UPDATE administrator SET administrator_name=%s WHERE administrator_id=%s",
731                                (new_name, admin_id))
732                 print("Name change successful!")
```

- Category == "N": administrator_name 변경

- new_name 입력 받음(문자열의 길이가 20보다 크면 문구와 함께 return)

- "UPDATE administrator SET administrator_name=%s WHERE administrator _id=%s",

(new_name, admin_id)

√. ADMINISTRATOR table에서 입력 받은 admin _id에 해당하는 tuple의 administrator_name 변경

```
734             elif category == "B":
735                 new_branch = int(input("Enter new branch_id: "))
736                 cursor.execute("SELECT * FROM branch WHERE branch_id=%s", new_branch)
737                 result = cursor.fetchone()
738                 if result is None:
739                     print("There is no such branch id!")
740                     return
741                 cursor.execute("UPDATE administrator SET branch_id=%s WHERE administrator_id=%s",
742                                (new_branch, admin_id))
743                 print("Branch change successful!")
```

- Category == "B": branch_id 변경

- new_branch 입력 받음

- "SELECT * FROM branch WHERE branch_id=%s", new_branch

√. BRANCH table에서 입력 받은 new_branch가 존재하는지 select

- 입력 받은 new_branch가 존재하지 않는다면 문구와 함께 return

- "UPDATE administrator SET branch_id=%s WHERE administrator_id=%s", (new_branch, admin_id)

√. ADMINISTRATOR table에서 입력 받은 admin_id에 해당하는 tuple의 branch_id 변경

```
745             elif category == "A":
746                 new_address = input("Enter new address(ex.Seoul): ")
747                 if len(new_address) > 20:
748                     print("You can enter up to 20 letters of your address. Please try again.")
749                     return
750                 cursor.execute("UPDATE administrator SET administrator_address=%s WHERE administrator_id=%s",
751                                (new_address, admin_id))
752                 print("Address change successful!")
```

- Category == "A": administrator_address 변경

- new_address 입력 받음(문자열의 길이가 20보다 크면 문구와 함께 return)

- "UPDATE administrator SET administrator_address=%s WHERE administrator_id=%s", (new_address, admin_id)

√. ADMINISTRATOR table에서 입력 받은 admin_id에 해당하는 tuple의 administrator_address 변경

```
754             elif category == "P":
755                 phone_command = input("Enter the command you want(Add:A, Delete: D): ")
756                 if phone_command == "A":
757                     new_phone = input("Enter your new phone number: ")
758                     if len(new_phone) != 13 or new_phone[3] != "-" or new_phone[8] != "-":
759                         print(
760                             "You should enter your phone number according to the style. "
761                             "Please try again.(ex.010-1234-5678)")
762                         return
763                     cursor.execute("INSERT INTO administrator_phone_number(administrator_id,phone_number)"
764                                    "values(%s,%s)", (admin_id, new_phone))
765                     print("Phone number addition successful!")
766                 elif phone_command == "D":
767                     origin_phone = input("Enter your existing phone number: ")
768                     cursor.execute("SELECT phone_number FROM administrator_phone_number "
769                                    "WHERE administrator_id=%s and phone_number=%s",
770                                    (admin_id, origin_phone))
771                     result = cursor.fetchone()
772                     if result is None:
773                         print("There's no such phone number!")
774                         return
775                     cursor.execute("DELETE FROM administrator_phone_number "
776                                    "WHERE administrator_id=%s and phone_number=%s",
777                                    (admin_id, origin_phone))
```

- Category == "P": phone_number 추가 or 삭제

- phone_command 입력 받음

- phone_command == "A": phone_number 추가

- new_phone 입력받음(문자열의 길이가 13이 아니거나 phone_number[3], phone_number[8]이 "-"가 아니라면 문구와 함께 return)

- "INSERT INTO administrator_phone_number(administrator_id,phone_number) values(%s,%s)", (admin_id, new_phone)

√. ADMINISTRATOR_PHONE_NUMBER table에 입력 받은 admin_id와 phone_number tuple insert

- phone_command == "D": phone_number 삭제

- origin_phone 입력 받음

- "SELECT phone_number FROM administrator_phone_number WHERE administrator_id=%s and phone_number=%s", (admin_id, origin_phone)

√. ADMINISTRATOR_PHONE_NUMBER table에서 해당 administrator_id와 phone_number를 가진 tuple select

- admin_id에 대해 입력 받은 origin_phone이 존재하지 않는다면 문구와 함께 return

- "DELETE FROM administrator_phone_number WHERE administrator_id=%s and phone_number=%s", (admin_id,origin_phone)

√. ADMINISTRATOR_PHONE_NUMBER table에서 해당 admin_id와 phone_number를 가진 tuple delete

8. Administrator Search

```
785     elif admin_command == 8:  # Administrator Search
786         search_command = input("Which search option do you want?(A:All, S:Specific): ")
787         if search_command == "A":
788             cursor.execute("SELECT * FROM administrator")
789             result_set = cursor.fetchall()
790             if result_set == ():
791                 print("There's no administrators")
792                 return
793             for row in result_set:
794                 print("Name: " + row[0])
795                 print("Administrator id: " + str(row[1]))
796                 print("Branch id: " + str(row[2]))
797                 print("Birth date: " + str(row[3].strftime('%Y-%m-%d')))
798                 print("Address: " + row[4])
799                 admin_id = row[1]
801                 cursor.execute("SELECT phone_number FROM administrator_phone_number WHERE administrator_id=%s",
802                                admin_id)
803                 result_phones = cursor.fetchall()
804                 print("Phone numbers: ", end='')
805                 if result_phones == ():
806                     print()
807                 for num in result_phones:
808                     if num is result_phones[-1]:
809                         print(num[0])
810                     else:
811                         print(num[0] + ", ", end='')
812                 print()
```

- search_command 입력 받음

- search_command == "A": 모든 ADMINISTRATOR tuple들 출력

- "SELECT * FROM administrator;"

√. ADMINISTRATOR table의 모든 tuple들 select

- tuple이 1개도 없다면 문구와 함께 return

- tuple이 존재한다면 모든 tuple들에 대해 정보 출력

- "SELECT phone_number FROM administrator_phone_number WHERE administrator_id=%s", admin_id

√. ADMINISTRATOR_PHONE_NUMBER에서 해당 admin_id에 대한 tuple select

- tuple이 존재하지 않는다면 한 줄 띄우고 종료, 존재한다면 comma(,) 단위로 출력

```
814         elif search_command == "S":
815             admin_id = int(input("Enter administrator's id you want to search: "))
816             cursor.execute("SELECT administrator_id FROM administrator WHERE administrator_id=%s", admin_id)
817             result = cursor.fetchone()
818             if result is None:
819                 print("There is no administrator for that administrator id!")
820             else:
821                 cursor.execute("SELECT * FROM administrator WHERE administrator_id=%s", admin_id)
822                 result = cursor.fetchone()
823 
824                 print("Name: " + result[0])
825                 print("Administrator id: " + str(result[1]))
826                 print("Branch id: " + str(result[2]))
827                 print("Birth date: " + str(result[3].strftime('%Y-%m-%d')))
828                 print("Address: " + result[4])
830                 cursor.execute("SELECT phone_number FROM administrator_phone_number WHERE administrator_id=%s",
831                                admin_id)
832                 result_phones = cursor.fetchall()
833                 print("Phone numbers: ", end='')
834                 if result_phones == ():
835                     print()
836                 for num in result_phones:
837                     if num is result_phones[-1]:
838                         print(num[0])
839                     else:
840                         print(num[0] + ", ", end='')
```

- search_command == "S": 특정 ADMINISTRATOR tuple 출력

- admin_id 입력 받음

- "SELECT administrator_id FROM administrator WHERE administrator_id=%s", admin_id

√. ADMINISTRATOR table에서 해당하는 admin_id를 가진 tuple의 administrator_id select

- 존재하지 않는다면 문구와 함께 return

- "SELECT * FROM administrator WHERE administrator_id=%s", admin_id

√. ADMINISTRATOR table에서 해당하는 admin_id를 가진 tuple select

- tuple에 대한 정보들 출력

- "SELECT phone_number FROM administrator_phone_number WHERE administrator_id=%s", admin_id

√. USER_PHONE_NUMBER table에서 해당 admin_id에 대한 tuple select

- tuple이 존재하지 않는다면 한 줄 띄우고 종료, 존재한다면 comma(,) 단위로 출력

9. Branch Registration

```
844     elif admin_command == 9:  # Branch Registration
845         branch_name = input("Enter the branch name you want to register: ")
846         if len(branch_name) > 20:
847             print("You can enter up to 20 letters of branch name. Please try again.")
848             return
849         cursor.execute("SELECT branch_name FROM branch WHERE branch_name=%s;", branch_name)
850         result = cursor.fetchone()
851 
852         if result is not None:
853             print("The name of the branch already exists")
854             return
855         else:
856             cursor.execute("INSERT INTO BRANCH(branch_name) values(%s)", branch_name)
857             print("Branch Registration Success!")
```

- branch_name 입력 받음(문자열의 길이가 20보다 크면 문구와 함께 return)

- "SELECT branch_name FROM branch WHERE branch_name=%s", branch_name

√. BRANCH table에서 해당하는 branch_name을 가진 tuple select

- 이미 존재하는 이름이라면 문구와 함께 return

- "INSERT INTO BRANCH(branch_name) values(%s)", branch_name

√. BRANCH table에 해당하는 branch_name을 가지는 tuple insert

10. Branch Deletion

```
859     elif admin_command == 10:  # Branch Deletion
860         branch_name = input("Enter the branch name you want to delete: ")
861         cursor.execute("SELECT branch_name FROM branch WHERE branch_name=%s;", branch_name)
862         result = cursor.fetchone()
863         if result is None:
864             print("There's no branch for that name!")
865         else:
866             cursor.execute("DELETE FROM branch WHERE branch_name=%s;", branch_name)
867             print("\"" + branch_name + "\" branch has been deleted")
```

- branch_name 입력 받음

- "SELECT branch_name FROM branch WHERE branch_name=%s", branch_name

√. BRANCH table에서 해당하는 branch_name을 가진 tuple select

- BRANCH table에 존재하지 않는다면 문구 출력

- "DELETE FROM branch WHERE branch_name=%s", branch_name

√. BRANCH tale에서 해당하는 branch_name을 가진 tuple delete

11. Branch Info Change

```
869     elif admin_command == 11:  # Branch Info Change
870         branch_name = input("Enter the branch name you want to change: ")
871         cursor.execute("SELECT branch_name FROM branch WHERE branch_name=%s;", branch_name)
872         result = cursor.fetchone()
873         if result is None:
874             print("There's no branch for that name!")
875         else:
876             new_name = input("Enter the new branch name: ")
877             if len(new_name) > 20:
878                 print("You can enter up to 20 letters of branch name. Please try again.")
879                 return
880             cursor.execute("UPDATE branch SET branch_name=%s WHERE branch_name=%s;", (new_name, branch_name))
881             print("\"" + branch_name + "\" branch has been changed to \"" + new_name + "\" branch")
```

- branch_name 입력 받음

- "SELECT branch_name FROM branch WHERE branch_name=%s", branch_name

√. BRANCH table에서 해당하는 branch_name을 가진 tuple select

- BRANCH table에 존재하지 않는다면 문구 출력

- new_name 입력 받음(문자열의 길이가 20보다 크면 문구와 함께 return)

- "UPDATE branch SET branch_name=%s WHERE branch_name=%s", (new_name, branch_name)

√. BRANCH table에서 해당하는 branch_name을 가진 tuple의 branch_name을 new_name으로 update

12. Branch Search

```
883     elif admin_command == 12:  # Branch Search
884         search_command = input("Which search option do you want?(A:All, S:Specific): ")
885         if search_command == "A":
886             cursor.execute("SELECT * FROM branch;")
887             result_set = cursor.fetchall()
888             if result_set == ():
889                 print("There's no branches")
890                 return
891             for row in result_set:
892                 print("Branch name: " + row[0])
893                 print("Branch ID: " + str(row[1]))
894                 print()
```

- search_command 입력 받음

- search_command == "A": 모든 BRANCH tuple들 출력

- SELECT * FROM branch;

√. BRANCH table의 모든 tuple들 select

- tuple이 1개도 없다면 문구와 함께 return

- tuple이 존재한다면 모든 tuple들에 대해 정보 출력

```
895         elif search_command == "S":
896             branch_search_command = input("Which search criteria you want?(N:branch_name, I:branch_id):")
897             if branch_search_command == "N":
898                 branch_name = input("Enter the branch name you want to search: ")
899                 cursor.execute("SELECT * FROM branch WHERE branch_name=%s;", branch_name)
900                 result = cursor.fetchone()
901                 if result is None:
902                     print("There's no branch for that name!")
903                 else:
904                     print("Branch name: " + result[0])
905                     print("Branch ID: " + str(result[1]))
906 
907             elif branch_search_command == "I":
908                 branch_id = int(input("Enter the branch id you want to search: "))
909                 cursor.execute("SELECT * FROM branch WHERE branch_id=%s;", branch_id)
910                 result = cursor.fetchone()
911                 if result is None:
912                     print("There's no branch for that id!")
913                 else:
914                     print("Branch name: " + result[0])
915                     print("Branch ID: " + str(result[1]))
```

- search_command == "S": 특정 BRANCH tuple 출력

- branch_search_command 입력 받음

- branch_search_command == "N": branch_name으로 검색

- branch_name 입력 받음

- "SELECT * FROM branch WHERE branch_name=%s", branch_name

√. BRANCH table에서 해당하는 branch_name을 가진 tuple select

- BRANCH table에 존재하지 않는다면 문구 출력

- 존재한다면 해당 tuple에 대한 정보 출력

- branch_search_command == "I": branch_id로 검색

- branch_id 입력 받음

- "SELECT * FROM branch WHERE branch_id=%s", branch_id

√. BRANCH table에서 해당하는 branch_id를 가진 tuple select

- BRANCH table에 존재하지 않는다면 문구 출력

- 존재한다면 해당 tuple에 대한 정보 출력

3. 실행 예시

- Bank Interface

```
C:₩Users₩wns21₩PycharmProjects₩pythonProject>python bank.py
Welcome to Bank DBMS

============================
Select Service:
0.Exit
1.User
2.Administrator
============================
Input:
```

- User Interface & Administrator Interface

```
============================
User Interface
0.Return to Previous Menu
1.Deposit/Withdrawal
2.Check Deposit/Withdrawal Details
3.User Registration
4.User Deletion
5.User Info Change
6.Account Registration
7.Account Deletion
============================
Input:
```

```
============================
Administrator Interface
0.Return to Previous Menu
1.Administer Account
2.Check Administration Details
3.User Search
4.Account Search
5.Administrator Registration
6.Administrator Deletion
7.Administrator Info Change
8.Administrator Search
9.Branch Registration
10.Branch Deletion
11.Branch Info Change
12.Branch Search
============================
Input:
```

- User Interface

1. Deposit/Withdrawal (INSERT)

```
Please enter your account number: 110-486-123456
Please enter your transaction type(D:Deposit/W:Withdrawal): D
Please enter your transaction amount: 100000
Deposit Successful!
```

```
Please enter your account number: 110-486-123456
Please enter your transaction type(D:Deposit/W:Withdrawal): W
Please enter your transaction amount: 200000
The withdrawal amount is higher than the balance
This account balance is 150000
```

- 거래를 원하는 계좌번호, 입/출금 옵션, 거래 금액 입력

- 거래를 원하는 account의 잔액보다 출금 금액이 더 높은 경우 출금 실패

2. Check Deposit/Withdrawal Details (SELECT)

```
Please enter your account number: 110-486-123456

Date: 2021-11-30
Type: Deposit
Amount: 100000

Date: 2021-11-30
Type: Withdrawal
Amount: 50000

Date: 2021-12-01
Type: Deposit
Amount: 100000
```

- 계좌번호 입력

3. User Registration (INSERT)

```
Enter user's name you want to register(Up to 20 letters): Messi
Enter user's ssn you want to register(ex.990101-1234567): 910701-1987654
Enter user's address you want to register(ex.Seoul): BuenosAires
Enter user's branch id you want to register: 6
Enter the id of administrator manage the user: 7
Enter user's nationalities you want to register
if more than two, please enter them all with comma
ex.Korea,USA
nationality: Argentina,Spain
Enter user's phone numbers you want to register
if more than two, please enter them all with comma
ex.010-1234-5678,010-2345-6789
phone number: 010-1010-1010
User registration Successful!
```

- Interface에서 정해주는 양식에 따라 등록을 원하는 user의 정보 입력

4. User Deletion (DELETE)

```
Enter user's id you want to delete: 5
user's id for "5"has been deleted
```

```
Enter user's id you want to delete: 4
This user have some accounts whose balance is not 0
User can be deleted only when all of having account's balance are 0
```

- user의 id 입력

- 삭제를 원하는 user가 잔액이 있는 account 보유 시, user 삭제 실패

5. User Info Change (UPDATE)

```
Enter user's id you want to change: 2
Which info you want to change?(N:Name, A:Address, B:Branch, Admin:Administrator, Nation:Nationality, P:PhoneNumber): A
Enter new address(ex.Seoul): Daejeon
Address change successful!
```

- user의 id 입력 후, 변경을 원하는 카테고리를 양식에 맞게 입력

6. Account Registration (INSERT)

```
Enter account number you want to register(Up to 17 letters): 111-875-725534
Enter user's id of the account you want to register: 2
Enter the type of account want to register(B:Bankbook, I:Installment saving, C:CMA):B
Account registration Successful!
```

```
Enter account number you want to register(Up to 17 letters): 123456
Enter user's id of the account you want to register: 1
User cannot have more thant 3 accounts
```

- 등록을 원하는 계좌번호, 계좌 user의 id, 계좌 유형 입력

- 한 사용자는 최대 3개의 account를 가질 수 있으며,

- 이미 account를 3개 가진 user가 계좌 등록을 시도 시, 계좌 등록 실패

7. Account Deletion (DELETE)

```
Enter account number you want to delete: 111-875-725534
account for "111-875-725534"has been deleted
```

```
Enter account number you want to delete: 110-486-123456
You can delete the account with a balance of 0
```

- 삭제를 원하는 계좌번호 입력

- 잔액이 0원인 account만 삭제 가능하므로,

- 잔액이 존재하는 account 삭제 시도 시, account 삭제 실패

- Administrator Interface

1. Administer Account (INSERT)

```
Please enter your administrator id: 3
Please enter the account number you want to administer: 110-486-123456
Enter the type of administration(C:Check Issue, A:Account Bookkeeping:, P: Pattern Analysis): P
Administer Successful!
```

```
Please enter your administrator id: 6
Please enter the account number you want to administer: 110-486-123456
Administrators can only administer accounts assigned to them
```

- administrator의 id, 계좌번호, 관리 유형 입력

- administrator는 자신이 담당하는 user의 account만 관리 가능하며,

- 자신이 담당하는 user 외에 다른 user의 account 관리 시도 시, 관리 실패

2. Check Administration Details (SELECT)

```
Which search option you want?(Admin, Account): Admin
Please enter administrator id: 2

Date: 2021-12-01
Administrator ID: 2
Account Number: 750-910237-90681
Type: Check Issue
```

```
Which search option you want?(Admin, Account): Account
Please enter account number: 110-486-123456

Date: 2021-12-01
Administrator ID: 3
Account Number: 110-486-123456
Type: Pattern Analysis
```

- administrator의 id, 계좌번호 중 원하는 옵션으로 관리 내역 검색

3. User Search (SELECT)

```
Which search option do you want?(A:All, S:Specific): S
Enter user's id you want to search: 1
Name: MJ Lee
Ssn: 990224-1092838
Address: Seoul
User ID: 1
Belonging branch id: 2
Treating administrator id: 3
Nationalities: Korea
Phone numbers: 010-5001-2333
Account numbers: 110-486-123456, 127-30382-293, 590-29383-11307
```

- 모든 user의 정보를 원한다면 A를 입력해 조회 가능

- 특정 user의 정보를 원한다면 S 입력 후, user의 id로 조회 가능

4. Account Search (SELECT)

```
Which search option do you want?(A:All, S:Specific): S
Enter account number you want to search: 110-486-123456
Account number: 110-486-123456
Account user id: 1
Account type: Bankbook
Account opening date: 2021-11-30
Account balance: 150000
```

- 모든 account의 정보를 원한다면 A를 입력해 조회 가능

- 특정 account의 정보를 원한다면 S 입력 후, 계좌번호로 조회 가능

5. Administrator Registration (INSERT)

```
Enter administrator's name you want to register(Up to 20 letters): Bale
Enter administrator's working branch id you want to register: 6
Enter administrator's birthdate you want to register(ex.2021-01-01): 1991-05-21
Enter administrator's address you want to register(ex.Seoul): Wales
Enter administrator's phone numbers you want to register
if more than two, please enter them all with comma
ex.010-1234-5678,010-2345-6789
phone number: 010-1107-1111,010-1111-0911
Administrator registration Successful!
```

- Interface에서 정해주는 양식에 따라 등록을 원하는 administrator의 정보 입력

6. Administrator Deletion (DELETE)

```
Enter administrator's id you want to delete: 8
administrator's id for "8" has been deleted
```

- administrator의 id 입력

- 삭제를 원하는 administrator가 관리하는 계좌가 존재할 시,

- 그 계좌의 administrator는 default administrator로 임시 변경

7. Administrator Info Change (UPDATE)

```
Enter administrator's id you want to change: 4
Which info you want to change?(N:Name, B:Branch, A:Address, P:PhoneNumber):A
Enter new address(ex.Seoul): Osaka
Address change successful!
```

- administrator의 id 입력 후, 변경을 원하는 카테고리를 양식에 맞게 입력

8. Administrator Search (SELECT)

```
Which search option do you want?(A:All, S:Specific): S
Enter administrator's id you want to search: 3
Name: GJ Lim
Administrator id: 3
Branch id: 1
Birth date: 1979-10-21
Address: Suwon
Phone numbers: 010-1128-9046
```

- 모든 administrator의 정보를 원한다면 A를 입력해 조회 가능

- 특정 administrator의 정보를 원한다면 S 입력 후, administrator의 id로 조회 가능

9. Branch Registration (INSERT)

```
Enter the branch name you want to register: Paris
Branch Registration Success!
```

- branch의 이름 입력

10. Branch Deletion (DELETE)

```
Enter the branch name you want to delete: Paris
"Paris" branch has been deleted
```

- branch의 이름 입력

- 삭제한 branch에 속한 user나 administrator가 존재할 시,

- 그 user 또는 administrator의 branch는 default branch로 임시 변경

11. Branch Info Change (UPDATE)

```
Enter the branch name you want to change: Washington
Enter the new branch name: NewYork
"Washington" branch has been changed to "NewYork" branch
```

- 기존 branch 이름 입력 후, 원하는 새로운 branch 이름 입력

12. Branch Search (SELECT)

```
Which search option do you want?(A:All, S:Specific): S
Which search criteria you want?(N:branch_name, I:branch_id):N
Enter the branch name you want to search: NewYork
Branch name: NewYork
Branch ID: 5
```

```
Which search option do you want?(A:All, S:Specific): S
Which search criteria you want?(N:branch_name, I:branch_id):I
Enter the branch id you want to search: 4
Branch name: Beijing
Branch ID: 4
```

- 모든 branch의 정보를 원한다면 A를 입력해 조회 가능

- 특정 branch의 정보를 원한다면 S 입력 후, branch 이름 또는 branch id로 조회 가능