Manhattan College

U.S. Banks Database

Olivia Keohane

Department of Computer Science

CMPT 258 Database Systems

Dr. Agrawal

December 6, 2018

Project Description

In the third and final project of the semester, we were to design the ER diagram, the schema diagram, and a front end application for our choice of a bank, department store, or library. Following a set of guidelines, I designed a bank database that met the required regulations.

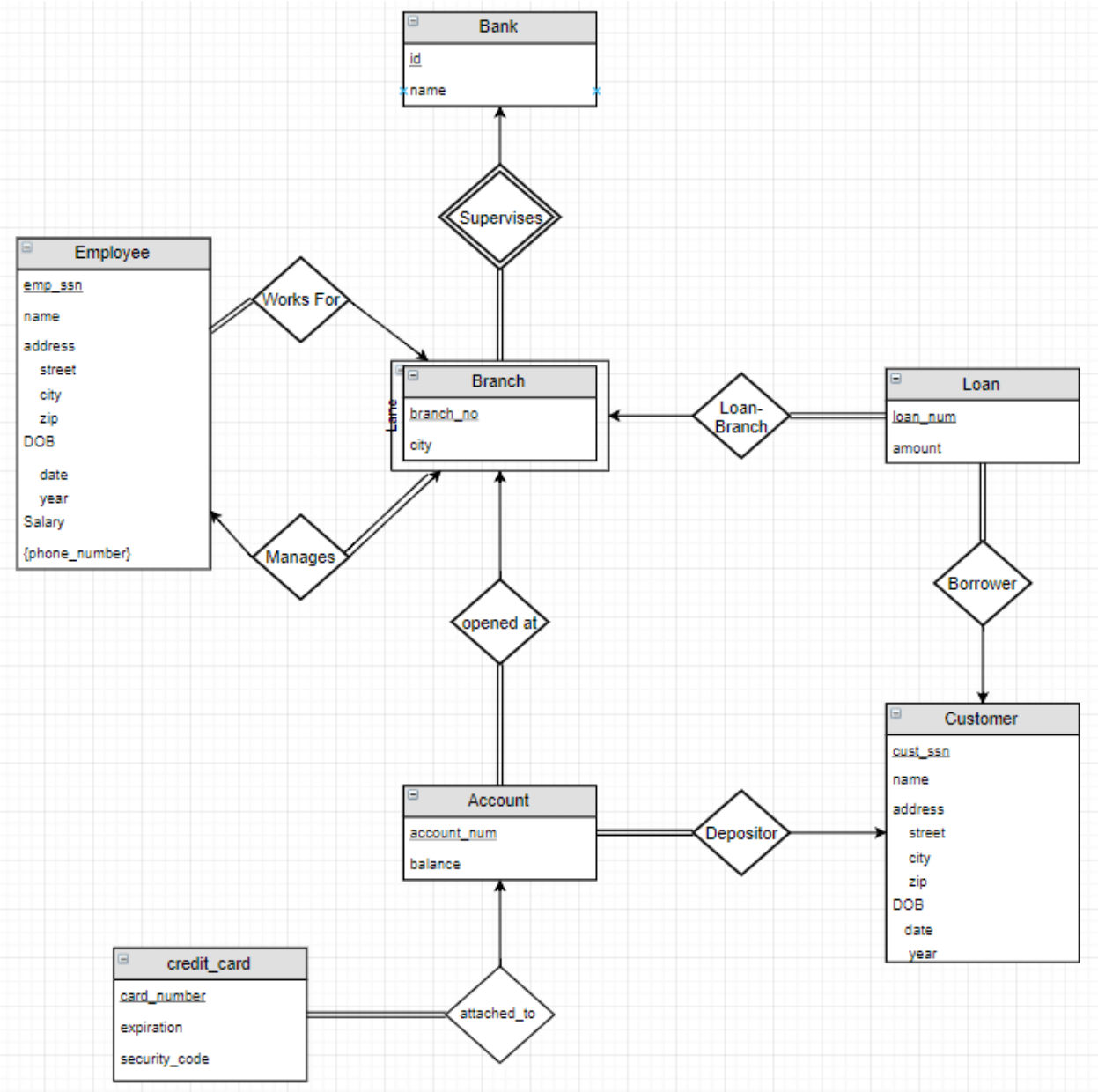
Problems Faced

I had taught myself a little bit of Python in Project #2 but it still provided some minor challenges while completing my code. Writing in SQL syntax through Python proved to be very tedious. I had to pay very close attention to both the syntaxes of Python and SQL. It was very easy to forget something minor while writing the queries, which would often prevent the code from proceeding until the minor issue was fixed.

What I Took from this Course and Project

This project made me feel more comfortable with using Python. It's also a very usefull skill to have to be able to connect the SQL database to IDLE in order to use Python as a means of creating the database. This course provided a different insight for me as to what databases are. Although there are guidelines on how to make the best and most efficient database, I like how you are still able to add a personal touch to it and because of this, no two database designers will design a database in the same way.

ER Diagram


Bank
id
name
Supervises
Employee
emp_ssn
name
address
street
city
zip
DOB
date
year
Salary
{phone_number}
Works For
Branch
branch_no
city
Loan-
Branch
Loan
loan_num
amount
Manages
Borrower
opened at
Customer
cust_ssn
name
address
street
city
zip
DOB
date
year
Account
account_num
balance
Depositor
credit_card
card_number
expiration
security_code
attached_to

Schema Diagram

bank(id, bank_name)

branch(id, branch_no, city)

customer(cust_ssn, cust_name, street, city, zip, date, year)

loan(loan_num, amount, cust_ssn, id, branch_no)

account(account_num, balance, cust_ssn, id, branch_no)

credit_card(card_number, expiration_month, expiration_year, security_code, account_num)

employee(emp_ssn, emp_name, street, city, zip, date, year, salary, id, branch_no)

emp_phone(emp_ssn, phone_number)

manages(id, branch_no, emp_ssn)

Complete Code (Python)

```
import mysql.connector

con = mysql.connector.connect(user='root',password='root',host='127.0.0.1',port='3306')


cursor = con.cursor()


#create database

cursor.execute("drop database bank;")    #carry out every run after initial run of program

cursor.execute("create database bank;")

cursor.execute("use bank;")


#create tables

cursor.execute("create table bank(id int, bank_name varchar(30), primary key (id));")

cursor.execute("create table branch(id int, branch_no int, city varchar (20), primary key(id, branch_no), foreign key (id) references bank(id) on delete cascade);")

cursor.execute("create table customer(cust_ssn int, cust_name varchar(40), street varchar(60), city varchar(20), zip int, date varchar(15), year int, primary key (cust_ssn));")

cursor.execute("create table loan(loan_num int, amount float, cust_ssn int, id int, branch_no int, primary key (loan_num), foreign key (cust_ssn) references customer (cust_ssn) on delete cascade, foreign key (id, branch_no) references branch(id, branch_no) on delete cascade);")

cursor.execute("create table account(account_num int, balance float, cust_ssn int, id int, branch_no int, primary key(account_num), foreign key (cust_ssn) references customer(cust_ssn) on delete cascade, foreign key (id, branch_no) references branch(id, branch_no) on delete cascade);")

cursor.execute("create table credit_card(card_number int, expiration_month varchar(10), expiration_year int, security_code int, account_num int, primary key (card_number), foreign key (account_num) references account (account_num) on delete cascade);")

cursor.execute("create table employee(emp_ssn int, emp_name varchar(40), street varchar(60), city varchar(20), zip int, date varchar(15), year int, salary int, id int, branch_no int, primary key (emp_ssn), foreign key (id, branch_no) references branch(id, branch_no) on delete cascade);")

cursor.execute("create table emp_phone(emp_ssn int, phone_number varchar(20), primary key (emp_ssn, phone_number), foreign key (emp_ssn) references employee (emp_ssn) on delete cascade);")

cursor.execute("create table manages(id int, branch_no int, emp_ssn int, primary key (id, branch_no, emp_ssn), foreign key (id, branch_no) references branch(id, branch_no) on delete cascade, foreign key (emp_ssn) references employee(emp_ssn) on delete cascade);")


#department insert statements

cursor.execute("insert into bank values(1, 'Chase');")

cursor.execute("insert into bank values(2, 'Bank of America');")

cursor.execute("insert into bank values(3, 'TD Bank');")
```

```
cursor.execute("insert into bank values(4, 'Wells Fargo');")

cursor.execute("insert into bank values(5, 'Capital One');")

cursor.execute("insert into bank values(6, 'Citibank');")

cursor.execute("insert into bank values(7, 'PNC');")

cursor.execute("insert into bank values(8, 'HSBC');")

cursor.execute("insert into bank values(9, 'Santander');")

cursor.execute("insert into bank values(10, 'Citizens');")


cursor.execute("insert into branch values(1, 3157, 'New York');")

cursor.execute("insert into branch values(2, 2546, 'Charlotte');")

cursor.execute("insert into branch values(3, 575, 'Boston');")

cursor.execute("insert into branch values(4, 2876, 'San Francisco');")

cursor.execute("insert into branch values(5, 378, 'San Francisco');")

cursor.execute("insert into branch values(6, 2300, 'New York');")

cursor.execute("insert into branch values(7, 1300, 'Pittsburgh');")

cursor.execute("insert into branch values(8, 1950, 'New York');")

cursor.execute("insert into branch values(9, 505, 'Boston');")

cursor.execute("insert into branch values(10, 600, 'Providence');")


cursor.execute("insert into customer values(215705871, 'Makayla Schulist', '8067 Eladio Street', 'New York, NY', 10015,
'November 26', 1940);")

cursor.execute("insert into customer values(635369784, 'Stella Hilpert', '2234 Kelly Street', 'Charlotte, NC ', 28263, 'August 18',
1945);")

cursor.execute("insert into customer values(047401985, 'Elouise Koch ', '2285 Briarwood Drive', 'Boston, MA', 02124,
'December 17', 1961);")

cursor.execute("insert into customer values(004668628, 'Lily Herzog ', '3342 Wayside Lane', 'San Francisco, CA', 94107, 'July
12', 1983);")

cursor.execute("insert into customer values(578942952, 'Barrett Kilback ', '3367 Golf Course Drive', 'San Francisco, CA', 94110,
'April 19', 1992);")

cursor.execute("insert into customer values(516270077, 'Edyth Cronin  ', '4170 Church Street', 'New York, NY', 10017, 'June 6',
1996);")

cursor.execute("insert into customer values(125368031, 'Lucinda Donnelly ', '1437 Stuart Street', 'Pittsburgh, PA', 15222, 'May
4', 1985);")

cursor.execute("insert into customer values(461231641, 'Tyshawn Hackett ', '3412 Geneva Street', 'New York, NY', 10014,
'September 24', 1978);")

cursor.execute("insert into customer values(047140993, 'Harmony Witting ', '772 Romano Street', 'Boston, MA', 02110,
'December 7', 1956);")
```

```
cursor.execute("insert into customer values(435623460, 'Althea Robel ', '1833 Winding Way', 'Providence, RI', 02908, 'July 8', 1943);")


cursor.execute("insert into loan values(101, 87650, 215705871, 1, 3157);")
cursor.execute("insert into loan values(201, 45000, 635369784, 2, 2546);")
cursor.execute("insert into loan values(301, 67500, 047401985, 3, 575);")
cursor.execute("insert into loan values(401, 97000, 004668628, 4, 2876);")
cursor.execute("insert into loan values(501, 155000, 578942952, 5, 378);")
cursor.execute("insert into loan values(601, 250000, 516270077, 6, 2300);")
cursor.execute("insert into loan values(701, 15000, 125368031, 7, 1300);")
cursor.execute("insert into loan values(801, 16000, 461231641, 8, 1950);")
cursor.execute("insert into loan values(901, 38000, 047140993, 9, 505);")
cursor.execute("insert into loan values(1001, 105000, 435623460, 10, 600);")


cursor.execute("insert into account values(691, 600000, 215705871, 1, 3157);")
cursor.execute("insert into account values(955, 300000, 635369784, 2, 2546);")
cursor.execute("insert into account values(459, 431000, 047401985, 3, 575);")
cursor.execute("insert into account values(542, 128000, 004668628, 4, 2876);")
cursor.execute("insert into account values(310, 98000, 578942952, 5, 378);")
cursor.execute("insert into account values(845, 14000, 516270077, 6, 2300);")
cursor.execute("insert into account values(255, 94123, 125368031, 7, 1300);")
cursor.execute("insert into account values(509, 224345, 461231641, 8, 1950);")
cursor.execute("insert into account values(178, 845678, 047140993, 9, 505);")
cursor.execute("insert into account values(703, 212000, 435623460, 10, 600);")


cursor.execute("insert into credit_card values(91301199, 'July', 2019, 214, 691);")
cursor.execute("insert into credit_card values(43077267, 'October', 2019, 657, 955);")
cursor.execute("insert into credit_card values(95087971, 'March', 2020, 915, 459);")
cursor.execute("insert into credit_card values(9026458, 'May', 2021, 903, 542);")
cursor.execute("insert into credit_card values(58241800, 'June', 2022, 241, 310);")
cursor.execute("insert into credit_card values(87678261, 'August', 2023, 809, 845);")
cursor.execute("insert into credit_card values(69414484, 'January', 2022, 921, 255);")
cursor.execute("insert into credit_card values(74638464, 'May', 2020, 452, 509);")
cursor.execute("insert into credit_card values(49731468, 'February', 2020, 829, 178);")
```

```
cursor.execute("insert into credit_card values(23715065, 'June', 2019, 623, 703);")


cursor.execute("insert into employee values(406385189, 'Sonja Chambers', '4264 Bell Street', 'New York, NY', 10013, 'January 26', 1981, 63996, 1, 3157);")

cursor.execute("insert into employee values(653264418, 'Elbert Gill', '4510 Concord Street', 'Charlotte, NC', 28202, 'February 12', 1982, 63000, 2, 2546);")

cursor.execute("insert into employee values(510544279, 'Lindsey Fitzgerald', '3818 Valley Street', 'Boston, MA', 02124, 'January 12', 1985, 95000, 3, 575);")

cursor.execute("insert into employee values(765195556, 'Ian Wallace', '4191 Delaware Avenue', 'San Francisco, CA', 94108, 'August 5', 1985, 64377, 4, 2876);")

cursor.execute("insert into employee values(032364984, 'Kayla Reeves', '1668 Perine Street', 'San Francisco, CA', 94109, 'January 2', 1986, 64339, 5, 378);")

cursor.execute("insert into employee values(569715045, 'Bertha Kelley', '1766 Bicetown Road', 'New York, NY', 10013, 'August 5', 1976, 76000, 6, 2300);")

cursor.execute("insert into employee values(609309730, 'Clara Medina', '1315 Platinum Drive', 'Pittsburgh, PA', 15219, 'April 8', 1986, 62814, 7, 1300);")

cursor.execute("insert into employee values(049263820, 'Yvette Watkins', '1782 Anmoore Road', 'New York, NY', 10014, 'May 16', 1985, 82581, 8, 1950);")

cursor.execute("insert into employee values(530040008, 'Abel Shaw', '2632 Pineview Drive', 'Boston, MA', 02110, 'March 3', 1984, 67015, 9, 505);")

cursor.execute("insert into employee values(502116336, 'Sandra Romero', '4092 Bond Street', 'Providence, RI', 02908, 'October 24', 1981, 59942, 10, 600);")

cursor.execute("insert into employee values(562304589, 'Pam Fox', '1168 Elm Drive', 'New York, NY', 10011, 'August 5', 1965, 66000, 1, 3157);")

cursor.execute("insert into employee values(678183161, 'Ernesto Hill', '1792 Lee Avenue', 'Boston, MA', 02124, 'May 21', 1976, 52814, 3, 575);")

cursor.execute("insert into employee values(003145219, 'Lonnie Willis', '3754 Forest Drive', 'San Francisco, CA', 94109, 'March 14', 1987, 62581, 5, 378);")

cursor.execute("insert into employee values(601496301, 'Maryanne Wolfe', '2476 Stiles Street', 'Pittsburgh, PA', 15219, 'January 5', 1981, 47015, 7, 1300);")

cursor.execute("insert into employee values(014563453, 'Jerome Warren', '4335 C Street', 'Boston, MA', 02110, 'May 11', 1970, 49942, 9, 505);")


cursor.execute("insert into emp_phone values(406385189, '631-514-7309');")

cursor.execute("insert into emp_phone values(653264418, '317-964-3863');")

cursor.execute("insert into emp_phone values(510544279, '570-948-2099');")

cursor.execute("insert into emp_phone values(765195556, '509-489-6303');")

cursor.execute("insert into emp_phone values(032364984, '231-303-2641');")

cursor.execute("insert into emp_phone values(569715045, '228-918-8478');")

cursor.execute("insert into emp_phone values(609309730, '636-745-3701');")
```

```
cursor.execute("insert into emp_phone values(049263820, '304-263-5489');")
cursor.execute("insert into emp_phone values(530040008, '352-726-5339');")
cursor.execute("insert into emp_phone values(502116336, '619-377-9156');")
cursor.execute("insert into emp_phone values(562304589, '646-459-8777');")
cursor.execute("insert into emp_phone values(562304589, '917-285-6981');")
cursor.execute("insert into emp_phone values(678183161, '856-321-7130');")
cursor.execute("insert into emp_phone values(003145219, '703-873-1151');")
cursor.execute("insert into emp_phone values(601496301, '412-527-9900');")
cursor.execute("insert into emp_phone values(014563453, '508-409-3915');")
cursor.execute("insert into emp_phone values(014563453, '617-697-1507');")

cursor.execute("insert into manages values(1, 3157, 406385189);")
cursor.execute("insert into manages values(2, 2546, 653264418);")
cursor.execute("insert into manages values(3, 575, 510544279);")
cursor.execute("insert into manages values(4, 2876, 765195556);")
cursor.execute("insert into manages values(5, 378, 032364984);")
cursor.execute("insert into manages values(6, 2300, 569715045);")
cursor.execute("insert into manages values(7, 1300, 609309730);")
cursor.execute("insert into manages values(8, 1950, 049263820);")
cursor.execute("insert into manages values(9, 505, 530040008);")
cursor.execute("insert into manages values(10, 600, 502116336);")
con.commit()



#*********************************ERROR-CHECK FUNCTIONS*********************************#
#checks if account number entered exists in the database
def existing_account(account_num):
    cursor.execute("select account_num from account where account_num = "+account_num+";")
    if cursor.fetchone() is None:   #if account number DOES NOT exist, return false
        return False;
    return True;    #if account number entered DOES exist in the database

#checks if withdrawal amount is within the balance in the bank account
def enough_money(account_num, amount):
```

```
    cursor.execute("select balance from account where account_num = "+account_num+";")
    for n in cursor:
        balance = n[0]

    if balance < int(amount):  #if the requested amount is LARGER THAN the balance, return false
        return False;
    return True;    #if the requested amount is SMALLER THAN the balance, return true

#checks if given ssn exists in the employee table
def existing_ssn(ssn):
    cursor.execute("select emp_ssn from employee where emp_ssn = "+ssn+";")
    if cursor.fetchone() is None:   #if ssn WAS NOT found in employee table
        return False;
    return True;    #if ssn WAS found in employee table

#checks if card number entered is a duplicate
def existing_card(number):
    cursor.execute("select card_number from credit_card where card_number = "+number+";")
    if cursor.fetchone() is None: #if card number WAS NOT found in credit_card table
        return False;
    return True;    #if card number WAS found = DUPLICATE
#*********************************************************************************#



#*************************************CHOICE 1*************************************#
#second menu item, checks requirements on deposit and performs deposit if allowed
def deposit(account_num, amount):
    if existing_account(account_num) != True:   #if account does not exist
        print("ERROR** The account number cannot be found. **ERROR")
        return;

    if int(amount) < 0: #if amount to be withdrawn is negative
        print("ERROR** Amount needs to be a positive value. **ERROR")
        return;
```

```
    cursor.execute("select balance from account where account_num = "+account_num+";")  #prints balance of existing account
    for n in cursor:
        originalBalance = n[0]
    print("Original Balance = $", str(originalBalance))

    #if account exists and request is within balance
    cursor.execute("update account set balance = balance + "+amount+" where account_num = "+account_num+";")

    #display overview of transaction
    cursor.execute("select balance from account where account_num = "+account_num+";")
    for n in cursor:
        newBalance = n[0]
    print("New Account Balance: $"+str(newBalance))
    return;
#*********************************************************************************#


#*************************************CHOICE 2*************************************#
#first menu item, checks requirements of withdrawal and performs withdrawal if allowed
def withdraw(account_num, amount):
    if existing_account(account_num) != True:   #if account does not exist
        print("ERROR** The account number cannot be found. **ERROR")
        return;

    if int(amount) < 0: #if amount to be withdrawn is negative
        print("ERROR** Amount needs to be a positive value. **ERROR")
        return;

    cursor.execute("select balance from account where account_num = "+account_num+";")  #prints balance of existing account
    for n in cursor:
        originalBalance = n[0]
    print("Current balance = $", str(originalBalance))
```

```
    if enough_money(account_num, amount) != True:   #if request exceeds balance
        print("ERROR** Request is greater than the account balance. **ERROR")
        return;

    #if account exists and request is within balance
    cursor.execute("update account set balance = balance - "+amount+" where account_num = "+account_num+";")

    #display overview of transaction
    cursor.execute("select balance from account where account_num = "+account_num+";")
    for n in cursor:
        newBalance = n[0]
    print("New Account Balance: $"+str(newBalance))
    return;
#*********************************************************************************#


#*************************************CHOICE 3*************************************#
#third menu item, searches input ssn in employee list and removes employee from database
def remove_emp(ssn):
    #checks if ssn exists in database
    if existing_ssn(ssn) != True:
        print("ERROR** SSN does not exist in the database. **ERROR")
        return;

    #if ssn does exist, remove employee
    cursor.execute("select emp_name from employee where emp_ssn = "+ssn+";")    #store name of employee first
    for n in cursor:
        emp_name = n[0]

    cursor.execute("delete from employee where emp_ssn = "+ssn+";")
    print(emp_name+" has been removed from the database.")
    return;
#*********************************************************************************#
```

```
#*************************************CHOICE 4*****************************************#
#fourth menu item, searches customer using the given account number, displays name of bank and number of credit cards
def search_cust(account_num):
    if existing_account(account_num) != True:   #if account does not exist
        print("ERROR** The account number cannot be found. **ERROR")
        return;

    #query to get customer info
    query = "select cust_name, bank_name, count(card_number) as Num_Cards from (account natural join customer), credit_card,
bank where credit_card.account_num = account.account_num and bank.id = account.id and account.account_num =
"+account_num+" group by cust_name;"

    cursor.execute(query)
    for n in cursor:
        print("CUSTOMER: ", n[0])
        print("BANK: ", n[1])
        print("NUMBER OF CREDIT CARDS: ", n[2])
    return;
#*******************************************************************************************#


#*************************************CHOICE 5*****************************************#
#fifth menu item, inserts new credit card given it follows all requirements
def add_card(account_num, number, month, year, sec_code):

    if existing_account(account_num) != True:   #if account does not exist
        print("ERROR** The account number cannot be found. **ERROR")
        return;

    if existing_card(number) == True:   #checks if number entered is a duplicate
        print("ERROR** Card number is a duplicate. **ERROR")
        return;
    if len(number) != 8:
        print("ERROR** Card number needs 8 digits. **ERROR")
```

```
        return;

    if int(year) < 2019:    #checks if year of expiration has not already occured
        print("ERROR** Invalid expiration year. **ERROR")
        return;
    if len(year) != 4:    #checks if year was entered in correct syntax
        print("ERROR** Invalid expiration year (use: YYYY). **ERROR")
        return;




    #if all requirements are met, a new credit card will be added
    cursor.execute("insert into credit_card values("+number+", '"+month+"', "+year+", "+sec_code+", "+account_num+");")
    cursor.nextset()

    print("Card registered successfully!")
    return;
#*********************************************************************************************#



#*************************************MAIN MENU*****************************************#
print("=================================================================")
print("\t  WELCOME TO THE U.S. BANKS DATABASE \t")
print("=================================================================")
def main_interface():
    another = True  #allows user to choose multiple menu items
    while another:
        print()
        print("MENU")
        print("1. DEPOSIT")
        print ("2. WITHDRAW")
        print ("3. REMOVE EMPLOYEE")
        print ("4. CUSTOMER BANKING INFO")
        print ("5. REGISTER NEW CREDIT CARD")
```

```
print ("6. EXIT")


choice = input("Please select an item: ")
if choice == '1':
    account_num = input('Enter Account Number: ')
    amount = input('Enter whole amount to be deposited: ')
    deposit(account_num, amount)   #calls deposit function w/ user input
elif choice == '2':
    account_num = input('Enter Account Number: ')
    amount = input('Enter whole amount to be withdrawn: ')
    withdraw(account_num, amount)   #calls withdraw function w/ user input
elif choice == '3':
    ssn = input('Enter SSN of employee you would like to remove: ')
    remove_emp(ssn);    #calls remove_emp function with given SSN
elif choice == '4':
    account_num = input('Enter Account Number: ')
    search_cust(account_num);
elif choice == '5':
    account_num = input('Enter Account Number: ')
    number = input('New Credit Card Number (8 numbers): ')
    month = input('Month of Expiration (ex: July): ')
    year = input('Year of expiration (YYYY): ')
    sec_code = input('Security Code: ')
    add_card(account_num, number, month, year, sec_code);
else:
    print("MENU CLOSED")
    break


a = input("Another choice?(y/n): ") #if user wants to choose another menu item
if (a == 'y'):
    another = True
elif(a == 'n'):
    another = False
```

```
main_interface()



cursor.close()
con.close()
```