Ryan Waterman

Homework 4 SQL Questions

Exercise 3-1

Retrieve the employee ID, first name, and last name for all bank employees. Sort by last name and then by first name.

In [3]:
```
!pip install dotenv psycopg2
```

```
Requirement already satisfied: dotenv in c:\users\water\anaconda3\envs\dse5002\lib\s
ite-packages (0.9.9)
Requirement already satisfied: psycopg2 in c:\users\water\anaconda3\envs\dse5002\lib
\site-packages (2.9.10)
Requirement already satisfied: python-dotenv in c:\users\water\anaconda3\envs\dse500
2\lib\site-packages (from dotenv) (1.1.0)
```

In [4]:
```
import os
import psycopg2
import pandas as pd
from dotenv import find_dotenv, dotenv_values
```

In [5]:
```
keys = list(dotenv_values(find_dotenv('.env')).items())
os.environ['POSTGRES_PASS'] = keys[1][1]
print(os.getenv('POSTGRES_PASS'))
```

```
password
```

In [6]:
```
conn = psycopg2.connect(
        host="localhost",
        database="bank",
        user="Lab_03",
        password=os.getenv('POSTGRES_PASS'),
        port="5432"
    )

conn.autocommit=True
```

In [7]:
```
# Instantiate the cursor
cursor = conn.cursor()
```

In [8]:
```
#Execute the query
cursor.execute("SELECT emp_id, fname, lname FROM employee")

#Get the response
response = cursor.fetchall()
```

In [9]:
```
#Here is one approach to parsing the response into a df...
df = pd.DataFrame(
    {
        'emp_id':[item[0] for item in response],
        'fname':[item[1]for item in response],
        'lname':[item[2]for item in response]
    }
    )
df.head(5)
```

Out[9]:

	emp_id	fname	lname
0	1	Michael	Smith
1	2	Susan	Barker
2	3	Robert	Tyler
3	4	Susan	Hawthorne
4	5	John	Gooding

In [10]:
```
# I think a better way to do this is to read in the
# whole response and assign the columns in the df construction
cursor.execute("SELECT emp_id, fname, lname FROM employee")

df1 = pd.DataFrame(data=cursor.fetchall(), columns=['emp_id', 'fname', 'lname'])
df1.head(5)
```

Out[10]:

	emp_id	fname	lname
0	1	Michael	Smith
1	2	Susan	Barker
2	3	Robert	Tyler
3	4	Susan	Hawthorne
4	5	John	Gooding

In [11]:
```
#Anywho... let's finish the problem. This should sort by last, then first
df.sort_values(by='lname').sort_values(by='fname')
```

Out[11]:

	emp_id	fname	lname
16	17	Beth	Fowler
6	7	Chris	Tucker
13	14	Cindy	Mason
14	15	Frank	Portman
5	6	Helen	Fleming
8	9	Jane	Grossman
12	13	John	Blake
4	5	John	Gooding
0	1	Michael	Smith
9	10	Paula	Roberts
17	18	Rick	Tulman
2	3	Robert	Tyler
11	12	Samantha	Jameson
7	8	Sarah	Parker
1	2	Susan	Barker
3	4	Susan	Hawthorne
15	16	Theresa	Markham
10	11	Thomas	Ziegler

As expected, the sort function can be chained together... Very convenient.

Now, try this with just a SQL query

In [38]:
```
cursor.execute("""SELECT emp_id, fname, lname
                FROM employee
                ORDER BY fname ASC, lname ASC;""")
cursor.fetchall()
```

```
Out[38]: [(17, 'Beth', 'Fowler'),
          (7, 'Chris', 'Tucker'),
          (14, 'Cindy', 'Mason'),
          (15, 'Frank', 'Portman'),
          (6, 'Helen', 'Fleming'),
          (9, 'Jane', 'Grossman'),
          (13, 'John', 'Blake'),
          (5, 'John', 'Gooding'),
          (1, 'Michael', 'Smith'),
          (10, 'Paula', 'Roberts'),
          (18, 'Rick', 'Tulman'),
          (3, 'Robert', 'Tyler'),
          (12, 'Samantha', 'Jameson'),
          (8, 'Sarah', 'Parker'),
          (2, 'Susan', 'Barker'),
          (4, 'Susan', 'Hawthorne'),
          (16, 'Theresa', 'Markham'),
          (11, 'Thomas', 'Ziegler')]
```

Okay, so this can be done within in the SQL query, I just needed to swap the sort order.

Exercise 3-2

Retrieve the account ID, customer ID, and available balance for all accounts whose status equals 'ACTIVE' and whose available balance is greater than $2,500.

```
In [17]: #execute the query... this looks like its well suited for just a SQL command
         cursor.execute("""SELECT account_id, cust_id, avail_balance
                         FROM account
                         WHERE status = 'ACTIVE' and avail_balance > '2500';""")
         cursor.fetchall()
```

```
Out[17]: [(3, 1, 3000.0),
          (12, 4, 5487.09),
          (15, 6, 10000.0),
          (17, 7, 5000.0),
          (18, 8, 3487.19),
          (22, 9, 9345.55),
          (24, 10, 23575.12),
          (27, 11, 9345.55),
          (28, 12, 38552.05),
          (29, 13, 50000.0)]
```

Exercise 3-3

Write a query against the account table that returns the IDs of the employees who opened the accounts (use the account.open_emp_id column). Include a single row for each distinct employee.

```
In [19]: cursor.execute("SELECT DISTINCT open_emp_id FROM account")
         cursor.fetchall()
```

```
Out[19]: [(13,), (10,), (1,), (16,)]
```

Exercise 4-3

Construct a query that retrieves all accounts opened in 2002.

```
In [21]: #It looks like account id is the primary key... Do an inner join on this column

cursor.execute("""SELECT * FROM account
                INNER JOIN transaction
                On account.account_id=transaction.account_id
                WHERE open_date > '2002-12-31';""")
pd.DataFrame(cursor.fetchall())
```

Out[21]:

	0	1	2	3	4	5	6	7	8	9	...	11	12	13	14	15
0	3	CD	1	2004-06-30	None	2004-06-30	ACTIVE	2	10	3000.00	...	T	3	2004-06-30	3	CDT
1	10	CHK	4	2003-09-12	None	2005-01-03	ACTIVE	1	1	534.12	...	T	8	2003-09-12	10	CDT
2	12	MM	4	2004-09-30	None	2004-11-11	ACTIVE	1	1	5487.09	...	T	10	2004-09-30	12	CDT
3	13	CHK	5	2004-01-27	None	2005-01-05	ACTIVE	4	16	2237.97	...	T	11	2004-01-27	13	CDT
4	15	CD	6	2004-12-28	None	2004-12-28	ACTIVE	1	1	10000.00	...	T	13	2004-12-28	15	CDT
5	17	CD	7	2004-01-12	None	2004-01-12	ACTIVE	2	10	5000.00	...	T	14	2004-01-12	17	CDT
6	21	CHK	9	2003-07-30	None	2004-12-15	ACTIVE	1	1	125.67	...	T	17	2003-07-30	21	CDT
7	22	MM	9	2004-10-28	None	2004-10-28	ACTIVE	1	1	9345.55	...	T	18	2004-10-28	22	CDT
8	23	CD	9	2004-06-30	None	2004-06-30	ACTIVE	1	1	1500.00	...	T	19	2004-06-30	23	CDT
9	28	CHK	12	2003-07-30	None	2004-12-15	ACTIVE	4	16	38552.05	...	T	21	2003-07-30	28	CDT

10 rows × 21 columns

I realized I misread the exercise intro and transaction data was only meant to be used for 4-1 and 4-2...

Here is 4-3 done the correct way:

In [23]:
```
cursor.execute("SELECT * FROM account WHERE open_date > '2002-12-31';")
pd.DataFrame(cursor.fetchall())
```

Out[23]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	3	CD	1	2004-06-30	None	2004-06-30	ACTIVE	2	10	3000.00	3000.00	T
1	10	CHK	4	2003-09-12	None	2005-01-03	ACTIVE	1	1	534.12	534.12	T
2	12	MM	4	2004-09-30	None	2004-11-11	ACTIVE	1	1	5487.09	5487.09	T
3	13	CHK	5	2004-01-27	None	2005-01-05	ACTIVE	4	16	2237.97	2897.97	T
4	15	CD	6	2004-12-28	None	2004-12-28	ACTIVE	1	1	10000.00	10000.00	T
5	17	CD	7	2004-01-12	None	2004-01-12	ACTIVE	2	10	5000.00	5000.00	T
6	21	CHK	9	2003-07-30	None	2004-12-15	ACTIVE	1	1	125.67	125.67	T
7	22	MM	9	2004-10-28	None	2004-10-28	ACTIVE	1	1	9345.55	9845.55	T
8	23	CD	9	2004-06-30	None	2004-06-30	ACTIVE	1	1	1500.00	1500.00	T
9	27	BUS	11	2004-03-22	None	2004-11-14	ACTIVE	2	10	9345.55	9345.55	T
10	28	CHK	12	2003-07-30	None	2004-12-15	ACTIVE	4	16	38552.05	38552.05	T
11	29	SBL	13	2004-02-22	None	2004-12-17	ACTIVE	3	13	50000.00	50000.00	T