TODO: Exercise 6-2, 6-3, 8-1, 8-2, 10-1

Import Pandas

In [2]:
```
import pandas as pd
```

Set up connection to DB

In [1]:
```
import os
from dotenv import find_dotenv, dotenv_values

keys = list(dotenv_values(find_dotenv('.env')).items())
os.environ['POSTGRES_PASS'] = keys[1][1]
os.environ['POSTGRES_USER'] = keys[2][1]
host = 'localhost'
port = '5432'
db = 'bank'

engine=f'postgresql://{os.getenv('POSTGRES_USER')}:{os.getenv('POSTGRES_PASS')}@{ho
```

Exercise 6-2

Write a compound query that finds the first and last names of all individual customers along with the first and last names of all employees.

In [5]:
```
pd.read_sql_query("""
                    SELECT fname, lname
                    FROM individual
                    UNION
                    SELECT fname, lname
                    FROM employee
                """,
                engine
                )
```

Out[5]:

	fname	lname
0	Paula	Roberts
1	Susan	Tingley
2	Samantha	Jameson
3	Cindy	Mason
4	Chris	Tucker
5	Richard	Farley
6	Frank	Portman
7	Rick	Tulman
8	Beth	Fowler
9	Sarah	Parker
10	John	Gooding
11	John	Hayward
12	Robert	Tyler
13	Margaret	Young
14	Charles	Frasier
15	Thomas	Ziegler
16	Theresa	Markham
17	Susan	Barker
18	Louis	Blake
19	John	Blake
20	James	Hadley
21	Susan	Hawthorne
22	John	Spencer
23	Jane	Grossman
24	Frank	Tucker
25	Helen	Fleming
26	Michael	Smith

Exercise 6-3

Sort the results from Exercise 6-2 by the lname column.

In [6]:
```
pd.read_sql_query("""
                    SELECT fname, lname
                    FROM individual
                    UNION
                    SELECT fname, lname
                    FROM employee
                    ORDER BY lname ASC
                """,
                engine
                )
```

Out[6]:

	fname	lname
0	Susan	Barker
1	Louis	Blake
2	John	Blake
3	Richard	Farley
4	Helen	Fleming
5	Beth	Fowler
6	Charles	Frasier
7	John	Gooding
8	Jane	Grossman
9	James	Hadley
10	Susan	Hawthorne
11	John	Hayward
12	Samantha	Jameson
13	Theresa	Markham
14	Cindy	Mason
15	Sarah	Parker
16	Frank	Portman
17	Paula	Roberts
18	Michael	Smith
19	John	Spencer
20	Susan	Tingley
21	Frank	Tucker
22	Chris	Tucker
23	Rick	Tulman
24	Robert	Tyler
25	Margaret	Young
26	Thomas	Ziegler

Exercise 8-1

Construct a query that counts the number of rows in the account table.

In [10]:
```
pd.read_sql_query("""
                    SELECT COUNT(*) rows
                    FROM account
                """,
                engine
                )
```

Out[10]:

	rows
0	24

Exercise 8-2

Modify your query from Exercise 8-1 to count the number of accounts held by each customer. Show the customer ID and the number of accounts for each customer.

In [22]:
```
pd.read_sql_query("""
                    SELECT cust_id,
                    COUNT(*) num_accounts
                    FROM account
                    GROUP BY cust_id
                    ORDER BY cust_id ASC
                """,
                engine
                )
```

Out[22]:

	cust_id	num_accounts
0	1	3
1	2	2
2	3	2
3	4	3
4	5	1
5	6	2
6	7	1
7	8	2
8	9	3
9	10	2
10	11	1
11	12	1
12	13	1

Exercise 10-1

Write a query that returns all product names along with the accounts based on that product (use the product_cd column in the account table to link to the product table). Include all products, even if no accounts have been opened for that product.

In [44]:
```
"""
This will require the following steps:
1. Select account id and product name
2. Full outer join on product_cd... this is required to return product names that a

After completing this, I am noticing that the type of account ID has changed. I thi
Maybe they are represented as floats? Thus, the column is cast to a float to have a
I don't think this matters for this application, but I will fix it anyway. I should

I will need some AI assistance on this, so I will output the data to a csv.
"""

pd.read_sql_query("""
                    SELECT a.account_id, p.name
                    FROM account a FULL OUTER JOIN product p
                    ON a.product_cd = p.product_cd
                """,
                engine
                ).to_csv("data/10-1.csv")
```

I consulted with Gemini on this problem, conversation link, below:

https://g.co/gemini/share/726570e92dc5

I am going to give the SQL statement that Gemini provided a shot.

In [49]:
```
pd.read_sql_query("""
                    SELECT COALESCE(a.account_id, -1) AS account_id, -- Replace NUL
                    p.name
                    FROM account a FULL OUTER JOIN product p
                    ON a.product_cd = p.product_cd
                """,
                engine
                )
```

Out[49]:

	account_id	name
0	1	checking account
1	2	savings account
2	3	certificate of deposit
3	4	checking account
4	5	savings account
5	7	checking account
6	8	money market account
7	10	checking account
8	11	savings account
9	12	money market account
10	13	checking account
11	14	checking account
12	15	certificate of deposit
13	17	certificate of deposit
14	18	checking account
15	19	savings account
16	21	checking account
17	22	money market account
18	23	certificate of deposit
19	24	checking account
20	25	business line of credit
21	27	business line of credit
22	28	checking account
23	29	small business loan
24	-1	home mortgage
25	-1	auto loan

Okay, this is better. I wish I could set -1 to NULL though.

I followed up with Gemini and now I have a better understanding of why this occurs, conversation, below:

https://g.co/gemini/share/2a47f4b5a45a

So, Pandas is at fault for the float casting, not SQL. The previous solution just modified the SQL output prior to Pandas interpreting it, so it appeared to do what I want.

Turns out, Gemini was right to suggest I need to modify the dataframe the first time around. Who would have guessed...

I guess I will oblige.

In [50]:
```
df = pd.read_sql_query("""
                    SELECT a.account_id, p.name
                    FROM account a FULL OUTER JOIN product p
                    ON a.product_cd = p.product_cd
                """,
                engine
                )
df['account_id'] = df['account_id'].astype('Int64')
df
```

Out[50]:

	account_id	name
0	1	checking account
1	2	savings account
2	3	certificate of deposit
3	4	checking account
4	5	savings account
5	7	checking account
6	8	money market account
7	10	checking account
8	11	savings account
9	12	money market account
10	13	checking account
11	14	checking account
12	15	certificate of deposit
13	17	certificate of deposit
14	18	checking account
15	19	savings account
16	21	checking account
17	22	money market account
18	23	certificate of deposit
19	24	checking account
20	25	business line of credit
21	27	business line of credit
22	28	checking account
23	29	small business loan
24	<NA>	home mortgage
25	<NA>	auto loan

Well, that did it. I apologize for the lengthy side quest. I should have recognized that I was looking at a pandas dataframe representation of the SQL query, therefore it would be a pandas issue.