In [1]:
```
import os
import sqlite3
import pandas as pd
from pathlib import PurePath
from collections import defaultdict
```

In [2]:
```
files = defaultdict(pd.DataFrame)
FILEPATH = PurePath("./data/")

for f in os.listdir(FILEPATH):
    filename, _ = f.split(".")
    files[filename] = pd.read_csv(FILEPATH / f)
```

In [3]:
```
for name, df in files.items():
    print(f"{name}: \n{df.head(3)}\n...\n")
```

```
salaries:
   emp_no  salary
0   10001   60117
1   10002   65828
2   10003   40006
...

dept_emp:
   emp_no dept_no
0   10001    d005
1   10002    d007
2   10003    d004
...

dept_manager:
  dept_no  emp_no
0    d001  110022
1    d001  110039
2    d002  110085
...

departments:
  dept_no        dept_name
0    d001        Marketing
1    d002          Finance
2    d003  Human Resources
...

titles:
  title_id              title
0    s0001              Staff
1    s0002        Senior Staff
2    e0001  Assistant Engineer
...

employees:
   emp_no emp_title_id  birth_date first_name  last_name sex  hire_date
0  473302        s0001   7/25/1953   Hideyuki   Zallocco   M  4/28/1990
1  475053        e0002  11/18/1954      Byong  Delgrande   F   9/7/1991
2   57444        e0002   1/30/1958      Berry       Babb   F  3/21/1992
...
```

In [4]:
```
conn = sqlite3.connect("SQL_Challenge.sqlite")
create_table_queries = {
    "departments": """
    CREATE TABLE IF NOT EXISTS departments (
        dept_name       VARCHAR(20) NOT NULL,
        dept_no         VARCHAR(4) PRIMARY KEY NOT NULL
    );
    """,
    "titles": """
    CREATE TABLE IF NOT EXISTS titles (
        title           TEXT NOT NULL,
        title_id        VARCHAR(5) NOT NULL
    );
    """,
    "employees": """
    CREATE TABLE IF NOT EXISTS employees (
        hire_date       TEXT NOT NULL,
        birth_date      TEXT NOT NULL,
        first_name      TEXT NOT NULL,
        sex             VARCHAR(1) NOT NULL,
        emp_title_id    VARCHAR(5) NOT NULL,
        emp_no          INTEGER PRIMARY KEY NOT NULL,

        FOREIGN KEY (emp_title_id) REFERENCES titles (title_id)
            ON DELETE CASCADE
            ON UPDATE CASCADE
    );
    """,
    "dept_emp": """
    CREATE TABLE IF NOT EXISTS dept_emp (
        emp_no          INTEGER NOT NULL,
        dept_no         VARCHAR(4) NOT NULL,

        FOREIGN KEY (emp_no) REFERENCES employees (emp_no)
            ON DELETE CASCADE
            ON UPDATE CASCADE,

        FOREIGN KEY (dept_no) REFERENCES departments (dept_no)
            ON DELETE CASCADE
            ON UPDATE CASCADE
    );
    """,
    "dept_manager": """
    CREATE TABLE IF NOT EXISTS dept_manager (
        emp_no          INTEGER NOT NULL,
        dept_no         VARCHAR(4) NOT NULL,
        FOREIGN KEY (emp_no) REFERENCES employees (emp_no)
            ON DELETE CASCADE
            ON UPDATE CASCADE,

            FOREIGN KEY (dept_no) REFERENCES departments (dept_no)
            ON DELETE CASCADE
            ON UPDATE CASCADE
    );
    """,
    "salaries": """
    CREATE TABLE IF NOT EXISTS salaries (
        emp_no          INTEGER NOT NULL,
        salary          INTEGER NOT NULL,
            FOREIGN KEY (emp_no) REFERENCES employees (emp_no)
                ON DELETE CASCADE
                ON UPDATE CASCADE
        );
        """,
}
```

In [5]:
```
for tablename, query in create_table_queries.items():
    conn.execute(query)
    files[tablename].to_sql(tablename, conn, if_exists="replace", index=False)
    conn.commit()
```

- List the employee number, last name, first name, sex, and salary of each employee (2 points)

In [6]:
```
pd.read_sql(
    """
    SELECT e.emp_no, e.last_name, e.first_name, e.sex, s.salary
    FROM employees e INNER JOIN salaries s ON e.emp_no = s.emp_no;
    """,
    conn,
)
```

Out[6]:

	emp_no	last_name	first_name	sex	salary
0	473302	Zallocco	Hideyuki	M	40000
1	475053	Delgrande	Byong	F	53422
2	57444	Babb	Berry	F	48973
3	421786	Verhoeff	Xiong	M	40000
4	282238	Baumann	Abdelkader	F	40000
...	...	...	...	...	...
300019	464231	Eastman	Constantino	M	69696
300020	255832	Dayang	Yuping	F	75355
300021	76671	Plessier	Ortrud	M	61886
300022	264920	Samarati	Percy	F	62772
300023	464503	Slobodova	Arvind	M	41708

300024 rows × 5 columns

• List the first name, last name, and hire date for the employees who were hired in 1986 (2 points)

In [7]:
```
pd.read_sql(
    """
    SELECT first_name, last_name, hire_date
    FROM employees
    WHERE hire_date LIKE "%/%/1986";
    """,
    conn,
)
```

Out[7]:

	first_name	last_name	hire_date
0	Eran	Cusworth	11/14/1986
1	Bojan	Zallocco	10/14/1986
2	Nevio	Demizu	5/18/1986
3	Ziva	Vecchi	7/3/1986
4	Mohit	Speek	1/14/1986
...	...	...	...
36145	Uriel	Heijenga	6/30/1986
36146	Ziyad	Constantine	2/28/1986
36147	Yishay	Maksimenko	1/27/1986
36148	Yannik	Ranai	4/6/1986
36149	Chenyi	Orlowska	12/25/1986

36150 rows × 3 columns

• List the manager of each department along with their department number, department name, employee number, last name, and first name (2 points)

In [8]:
```
# ds.dept_no, ds.dept_name, dm.emp_no,
pd.read_sql(
    """
    SELECT ds.dept_no, ds.dept_name, dm.emp_no, e.last_name, e.first_name
    FROM dept_manager dm INNER JOIN departments ds ON dm.dept_no = ds.dept_no
    INNER JOIN employees e ON e.emp_no = dm.emp_no;
    """,
    conn,
)
```

Out[8]:

	dept_no	dept_name	emp_no	last_name	first_name
0	d001	Marketing	110022	Markovitch	Margareta
1	d001	Marketing	110039	Minakawa	Vishwani
2	d002	Finance	110085	Alpin	Ebru
3	d002	Finance	110114	Legleitner	Isamu
4	d003	Human Resources	110183	Ossenbruggen	Shirish
5	d003	Human Resources	110228	Sigstam	Karsten
6	d004	Production	110303	Wegerle	Krassimir
7	d004	Production	110344	Cools	Rosine
8	d004	Production	110386	Kieras	Shem
9	d004	Production	110420	Ghazalie	Oscar
10	d005	Development	110511	Hagimont	DeForest
11	d005	Development	110567	DasSarma	Leon
12	d006	Quality Management	110725	Onuegbe	Peternela
13	d006	Quality Management	110765	Hofmeyr	Rutger
14	d006	Quality Management	110800	Quadeer	Sanjoy
15	d006	Quality Management	110854	Pesch	Dung
16	d007	Sales	111035	Kaelbling	Przemyslawa
17	d007	Sales	111133	Zhang	Hauke
18	d008	Research	111400	Staelin	Arie
19	d008	Research	111534	Kambil	Hilary
20	d009	Customer Service	111692	Butterworth	Tonny
21	d009	Customer Service	111784	Giarratana	Marjo
22	d009	Customer Service	111877	Spinelli	Xiaobin
23	d009	Customer Service	111939	Weedman	Yuchang

• List the department number for each employee along with that employee's employee number, last name, first name, and department name (2 points)

In [9]:
```
# de.dept_no, e.emp_no, e.last_name, e.first_name, d.dept_name
pd.read_sql(
    """
    SELECT de.dept_no, e.emp_no, e.last_name, e.first_name, d.dept_name
    FROM employees e
    INNER JOIN dept_emp de ON e.emp_no = de.emp_no
    INNER JOIN departments d ON d.dept_no = de.dept_no
    """,
    conn,
)
```

Out[9]:

	dept_no	emp_no	last_name	first_name	dept_name
0	d002	473302	Zallocco	Hideyuki	Finance
1	d004	475053	Delgrande	Byong	Production
2	d004	57444	Babb	Berry	Production
3	d003	421786	Verhoeff	Xiong	Human Resources
4	d006	282238	Baumann	Abdelkader	Quality Management
...	...	...	...	...	...
331598	d004	255832	Dayang	Yuping	Production
331599	d007	76671	Plessier	Ortrud	Sales
331600	d002	264920	Samarati	Percy	Finance
331601	d007	264920	Samarati	Percy	Sales
331602	d008	464503	Slobodova	Arvind	Research

331603 rows × 5 columns

• List first name, last name, and sex of each employee whose first name is Hercules and whose last name begins with the letter B (2 points)

In [10]:
```
# de.dept_no, e.emp_no, e.last_name, e.first_name, d.dept_name
pd.read_sql(
    """
    SELECT first_name, last_name, sex
    FROM employees
    WHERE first_name = "Hercules" AND last_name LIKE "B%";
    """,
    conn,
)
```

Out[10]:

	first_name	last_name	sex
0	Hercules	Baer	M
1	Hercules	Biron	F
2	Hercules	Birge	F
3	Hercules	Berstel	F
4	Hercules	Bernatsky	M
5	Hercules	Bail	F
6	Hercules	Bodoff	M
7	Hercules	Benantar	F
8	Hercules	Basagni	M
9	Hercules	Bernardinello	F
10	Hercules	Baranowski	M
11	Hercules	Bisiani	F
12	Hercules	Benzmuller	M
13	Hercules	Bahr	M
14	Hercules	Biran	F
15	Hercules	Bain	F
16	Hercules	Brendel	F
17	Hercules	Buchter	M
18	Hercules	Barreiro	M
19	Hercules	Baak	M

• List each employee in the Sales department, including their employee number, last name, and first name (2 points)

In [11]:
```
pd.read_sql(
    """
    SELECT e.emp_no, last_name, first_name
    FROM employees e
    INNER JOIN dept_emp de ON e.emp_no = de.emp_no
    WHERE dept_no = "d007";
    """,
    conn
)
```

Out[11]:

	emp_no	last_name	first_name
0	10002	Simmel	Bezalel
1	10016	Cappelletti	Kazuhito
2	10034	Swan	Bader
3	10041	Lenart	Uri
4	10050	Dredge	Yinghua
...	...	...	...
52240	499976	Felder	Guozhong
52241	499980	Usery	Gino
52242	499986	Ranta	Nathan
52243	499987	Dusink	Rimli
52244	499988	Kleiser	Bangqing

52245 rows × 3 columns

I could have done like the following, but the performance would be greatly affected

In [12]:
```
pd.read_sql(
        """
        SELECT e.emp_no, last_name, first_name
        FROM employees e
        INNER JOIN dept_emp de ON e.emp_no = de.emp_no
        INNER JOIN departments d ON d.dept_no = de.dept_no
        WHERE dept_name = "Finance";
        """,
        conn
)
```

Out[12]:

	emp_no	last_name	first_name
0	10042	Stamatiou	Magy
1	10050	Dredge	Yinghua
2	10059	McAlpine	Alejandro
3	10080	Baek	Premal
4	10132	Skrikant	Ayakannu
...	...	...	...
17341	499950	Gente	Weidon
17342	499975	Chorvat	Masali
17343	499977	Weisert	Martial
17344	499989	Lindqvist	Keiichiro
17345	499998	Breugel	Patricia

17346 rows × 3 columns

• List each employee in the Sales and Development departments, including their employee number, last name, first name, and department name (4 points)

In [13]:
```
pd.read_sql(
    """
    SELECT e.emp_no, first_name, last_name, dept_name
    FROM employees e
    INNER JOIN dept_emp de ON e.emp_no = de.emp_no
    INNER JOIN departments d ON de.dept_no = d.dept_no
    WHERE dept_name IN ("Development", "Sales");
    """,
    conn,
)
```

Out[13]:

	emp_no	first_name	last_name	dept_name
0	10001	Georgi	Facello	Development
1	10006	Anneke	Preusig	Development
2	10008	Saniya	Kalloufi	Development
3	10012	Patricio	Bridgland	Development
4	10014	Berni	Genin	Development
...	...	...	...	...
137947	499976	Guozhong	Felder	Sales
137948	499980	Gino	Usery	Sales
137949	499986	Nathan	Ranta	Sales
137950	499987	Rimli	Dusink	Sales
137951	499988	Bangqing	Kleiser	Sales

137952 rows × 4 columns

• List the frequency counts, in descending order, of all the employee last names (that is, how many employees share each last name) (4 points)

In [14]:
```
pd.read_sql(
    """
    SELECT last_name, COUNT(last_name) AS amount
    FROM employees
    GROUP BY last_name
    ORDER by (last_name) DESC;
    """,
    conn,
)
```

Out[14]:

	last_name	amount
0	dAstous	166
1	Zykh	148
2	Zyda	181
3	Zwicker	176
4	Zweizig	180
...	...	...
1633	Akaboshi	199
1634	Aingworth	172
1635	Adachi	221
1636	Acton	189
1637	Aamodt	205

1638 rows × 2 columns

In [15]:
```
conn.close()
os.remove("SQL_Challenge.sqlite")
```