SAN JOSE STATE UNIVERSITY - CMPE 180B - Database Systems

Phuong Duy Lam, Nguyen

SJSU ID: 018229432

HOMEWORK 1

Due date: 03/07/2025 11:59PM

Problem 1 (6.10 in the text book)

In [5]:
```python
#Install pysqlite3 for python and import pandas to use later
#!pip install pysqlite3
from sqlite3 import dbapi2 as sqlite3
print(sqlite3.sqlite_version)
import pandas as pd
from IPython.display import display, HTML
```

```
3.45.3
```

In [6]:
```python
dbname = "homework1-1.db"

def printSqlResults(cursor, tblName):
  try:
    df = pd.DataFrame(cursor.fetchall(), columns=[i[0] for i in cursor.description])
    display(HTML("<b><font color=Green> " + tblName + "</font></b>" + df.to_html(index=False)))
  except:
    pass

def runSql(caption, query):
  conn = sqlite3.connect(dbname) # Connect to the database
  cursor = conn.cursor() # Create a cursor (think: it's like a "pointer")
  cursor.execute(query) # Execute the query
  printSqlResults(cursor, caption) # Print the results
  conn.close()

def runSql_withCommit(caption, query):
  conn = sqlite3.connect(dbname) # Connect to the database
  cursor = conn.cursor() # Create a cursor (think: it's like a "pointer")
  cursor.execute(query) # Execute the query
  printSqlResults(cursor, caption) # Print the results
  conn.commit()
  conn.close()

def runStepByStepSql(query, fromline):
  lines = query.strip().split('\n')
  for lineidx in range(fromline, len(lines)):
    partial_query = '\n'.join(lines[:lineidx])
    caption = 'Query till line:' +  partial_query
    runSql(caption, partial_query + ';')
```

In [7]:
```python
# Connect to SQLite database (or create it)
conn = sqlite3.connect("homework1-1.db")
cursor = conn.cursor()

# Drop tables if they exist (to prevent duplication)
cursor.executescript("""
DROP TABLE IF EXISTS EMPLOYEE;
DROP TABLE IF EXISTS DEPARTMENT;
DROP TABLE IF EXISTS WORKS_ON;
DROP TABLE IF EXISTS PROJECT;
DROP TABLE IF EXISTS DEPENDENT;
DROP TABLE IF EXISTS DEPT_LOCATIONS;
""")

# Create tables
cursor.executescript("""
CREATE TABLE EMPLOYEE (
    Fname TEXT,
    Minit TEXT,
    Lname TEXT,
    Ssn TEXT PRIMARY KEY,
    Bdate TEXT,
    Address TEXT,
    Sex TEXT,
    Salary INTEGER,
    Super_ssn TEXT,
    Dno INTEGER
);

CREATE TABLE DEPARTMENT (
    Dname TEXT,
    Dnumber INTEGER PRIMARY KEY,
    Mgr_ssn TEXT,
    Mgr_start_date TEXT
);

CREATE TABLE WORKS_ON (
    Essn TEXT,
    Pno INTEGER,
    Hours REAL,
    FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn)
);

CREATE TABLE PROJECT (
    Pname TEXT,
    Pnumber INTEGER PRIMARY KEY,
    Plocation TEXT,
    Dnum INTEGER,
    FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber)
);

CREATE TABLE DEPENDENT (
    Essn TEXT,
    Dependent_name TEXT,
    Sex TEXT,
    Bdate TEXT,
    Relationship TEXT,
    FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn)
);

CREATE TABLE DEPT_LOCATIONS (
    Dnumber INTEGER,
    Dlocation TEXT,
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
);
""")

conn.commit()
```

In [8]:
```python
# Insert Employee Data
cursor.executemany("""
INSERT INTO EMPLOYEE VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
""", [
    ("John", "B", "Smith", "123456789", "1965-01-09", "731 Fondren, Houston, TX", "M", 30000, "333445555", 5),
    ("Franklin", "T", "Wong", "333445555", "1955-12-08", "638 Voss, Houston, TX", "M", 40000, "888665555", 5),
    ("Alicia", "J", "Zelaya", "999887777", "1968-01-19", "3321 Castle, Spring, TX", "F", 25000, "987654321", 4),
    ("Jennifer", "S", "Wallace", "987654321", "1941-06-20", "291 Berry, Bellaire, TX", "F", 43000, "888665555", 4),
    ("Ramesh", "K", "Narayan", "666884444", "1962-09-15", "975 Fire Oak, Humble, TX", "M", 38000, "333445555", 5),
    ("Joyce", "A", "English", "453454353", "1972-07-31", "5631 Rice, Houston, TX", "F", 25000, "333445555", 5),
    ("Ahmad", "V", "Jabbar", "987987987", "1969-03-29", "980 Dallas, Houston, TX", "M", 25000, "987654321", 4),
    ("James", "E", "Borg", "888665555", "1937-11-10", "450 Stone, Houston, TX", "M", 55000, None, 1)
])

# Insert Department Data
cursor.executemany("""
INSERT INTO DEPARTMENT VALUES (?, ?, ?, ?);
""", [
    ("Research", 5, "333445555", "1988-05-22"),
    ("Administration", 4, "987654321", "1995-01-01"),
    ("Headquarters", 1, "888665555", "1981-06-19")
])

# Insert Works_On Data
cursor.executemany("""
INSERT INTO WORKS_ON VALUES (?, ?, ?);
""", [
    ("123456789", 1, 32.5),
    ("123456789", 2, 7.5),
    ("666884444", 3, 40.0),
    ("453454353", 1, 20.0),
    ("453454353", 2, 20.0),
    ("333445555", 10, 10.0),
    ("333445555", 20, 10.0),
    ("333445555", 30, 10.0),
    ("999887777", 30, 10.0),
    ("987987987", 10, 35.0),
    ("987654321", 30, 20.0),
    ("987654321", 20, 15.0),
    ("888665555", 20, None)
])

# Insert Project Data
cursor.executemany("""
INSERT INTO PROJECT VALUES (?, ?, ?, ?);
""", [
    ("ProductX", 1, "Bellaire", 5),
    ("ProductY", 2, "Sugarland", 5),
    ("ProductZ", 3, "Houston", 5),
    ("Computerization", 10, "Stafford", 4),
    ("Reorganization", 20, "Houston", 1),
    ("Newbenefits", 30, "Stafford", 4)
])

# Insert Dependent Data
cursor.executemany("""
INSERT INTO DEPENDENT VALUES (?, ?, ?, ?, ?);
""", [
    ("333445555", "Alice", "F", "1986-04-05", "Daughter"),
    ("333445555", "Theodore", "M", "1983-10-25", "Son"),
    ("333445555", "Joy", "F", "1958-05-03", "Spouse"),
    ("987654321", "Abner", "M", "1942-02-28", "Spouse"),
    ("123456789", "Michael", "M", "1988-01-04", "Son"),
    ("123456789", "Alice", "F", "1988-12-30", "Daughter"),
    ("123456789", "Elizabeth", "F", "1967-05-05", "Spouse")
])

# Insert Department Locations
cursor.executemany("""
INSERT INTO DEPT_LOCATIONS VALUES (?, ?);
""", [
    (1, "Houston"),
    (4, "Stafford"),
    (5, "Bellaire"),
    (5, "Sugarland"),
    (5, "Houston")
])

# Commit changes and close the connection
conn.commit()
conn.close()

print("Database successfully created and populated!")
runSql('EMPLOYEE', "select * from EMPLOYEE;")
runSql('DEPARTMENT', "select * from DEPARTMENT;")
runSql('WORKS_ON', "select * from WORKS_ON;")
runSql('PROJECT', "select * from PROJECT;")
runSql('DEPENDENT', "select * from DEPENDENT;")
runSql('DEPT_LOCATIONS', "select * from DEPT_LOCATIONS;")
```

```
Database successfully created and populated!
```

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453454353	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	None	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Headquarters	1	888665555	1981-06-19
Administration	4	987654321	1995-01-01
Research	5	333445555	1988-05-22

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453454353	1	20.0
453454353	2	20.0
333445555	10	10.0
333445555	20	10.0
333445555	30	10.0
999887777	30	10.0
987987987	10	35.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NaN

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

In [9]:
```python
qry_question1a="""
SELECT E.Fname, E.Minit, E.Lname, E.Dno, W.Hours, P.Pname
FROM EMPLOYEE E
JOIN WORKS_ON W ON E.Ssn = W.Essn
JOIN PROJECT P ON W.Pno = P.Pnumber
WHERE E.Dno = 5 AND W.Hours > 10 AND P.Pname = 'ProductX'
;
"""
runSql('Question 1a (6.10a)', qry_question1a)
```

Question 1a (6.10a)

Fname	Minit	Lname	Dno	Hours	Pname
John	B	Smith	5	32.5	ProductX
Joyce	A	English	5	20.0	ProductX

In [10]:
```python
qry_question1b="""
SELECT E.Fname, E.Minit, E.Lname, DE.Dependent_name, DE.Relationship
FROM EMPLOYEE E
JOIN DEPENDENT DE ON E.Ssn = DE.Essn
WHERE E.Fname = DE.Dependent_name
;
"""
runSql('Question 1b (6.10b)', qry_question1b)
```

Question 1b (6.10b)

Fname	Minit	Lname	Dependent_name	Relationship

In [11]:
```python
qry_question1c="""
SELECT E.Fname, E.Minit, E.Lname, E.Ssn
FROM EMPLOYEE E
WHERE E.Super_ssn = (SELECT E1.Ssn From EMPLOYEE E1 WHERE E1.Fname = 'Franklin' AND E1.Lname = 'Wong')
;
"""
runSql('Question 1c (6.10c)', qry_question1c)
```

Question 1c (6.10c)

Fname	Minit	Lname	Ssn
John	B	Smith	123456789
Ramesh	K	Narayan	666884444
Joyce	A	English	453454353

Problem 2

Specify the following query on the database in Figure 5.5 in SQL.

Show the query results if the query is applied to the database state in Figure 5.6. -For each project whose average employee salary is more than $27,000, retrieve the project name and the number of employees working on that project.

In [14]:
```python
qry_question2="""
SELECT P.Pname, COUNT(DISTINCT E.Ssn) AS Number_Employee_Working_Project, AVG(E.Salary) AS Average_Project_Salary
FROM Project P
JOIN WORKS_ON W ON P.Pnumber = W.Pno
JOIN EMPLOYEE E ON W.Essn = E.Ssn
GROUP BY P.Pname
HAVING AVG(E.Salary) > 27000
;
"""
runSql('Problem 2', qry_question2)
```

Problem 2

Pname	Number_Employee_Working_Project	Average_Project_Salary
Computerization	2	32500.0
Newbenefits	3	36000.0
ProductX	2	27500.0
ProductY	2	27500.0
ProductZ	1	38000.0
Reorganization	3	46000.0

Problem 3

In SQL, show the following queries on the database in Figure 5.5 using the concept of nested queries and other concepts described in chapter 7.

Additionally, list the results of these queries.

a. Retrieve the names of all employees who work in the department that has the employee with the highest salary among all employees. b. Retrieve the names of all employees whose supervisor's supervisor has '123456789' for Ssn. c. Retrieve the names of employees who make at least $10,000 more than the employee who is paid the least in the company.

In [17]:
```python
qry_question3a="""
SELECT E.Fname, E.Minit, E.Lname, DE.Dname, E.Salary
FROM EMPLOYEE E
JOIN DEPARTMENT DE ON DE.Dnumber = E.Dno
WHERE E.Dno = (SELECT Dno FROM EMPLOYEE WHERE Salary = (SELECT MAX(Salary) FROM EMPLOYEE))
;
"""
runSql('Problem 3a', qry_question3a)
```

Problem 3a

Fname	Minit	Lname	Dname	Salary
James	E	Borg	Headquarters	55000

In [18]:
```python
qry_question3b="""
SELECT E.Fname, E.Minit, E.Lname, E.Super_ssn
FROM EMPLOYEE E
WHERE E.Super_ssn IN (SELECT S.Ssn FROM EMPLOYEE S WHERE S.Super_ssn = 123456789)
;
"""
runSql('Problem 3b', qry_question3b)
```

Problem 3b

Fname	Minit	Lname	Super_ssn

In [19]:
```python
qry_question3c="""
SELECT S.Fname, S.Minit, S.Lname, S.Salary
FROM EMPLOYEE S
WHERE S.Salary >= (
SELECT MIN(E.Salary)
FROM EMPLOYEE E) + 10000
;
"""
runSql('Problem 3c', qry_question3c)
```

Problem 3c

Fname	Minit	Lname	Salary
Franklin	T	Wong	40000
Jennifer	S	Wallace	43000
Ramesh	K	Narayan	38000
James	E	Borg	55000

In [ ]: