Lecture 4: In-class activity - Update, delete

Let's setup the SQL environment

In [1]:
```
#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
```

Let's define some helper functions for running queries and printing results

In [2]:
```
dbname = "music_streaming4.db"

def printSqlResults(cursor, tblName):
  try:
    df = pd.DataFrame(cursor.fetchall(), columns=[i[0] for i in cursor
    display(HTML("<b><font color=Green> " + tblName + "</font></b>" +
  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 "pointe
  cursor.execute(query) # Execute the query
  printSqlResults(cursor, caption) # Print the results
  conn.close()

def runSqlWithCommit(caption, query):
  conn = sqlite3.connect(dbname) # Connect to the database
  cursor = conn.cursor() # Create a cursor (think: it's like a "pointe
  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 + ';')
```

Let's setup a Schema and insert some data

In [3]:
```
# Connect to database (creates the file if it doesn't exist)
"""
1. Connections: A connection represents a connection to a database thr
which we can execute SQL queries. The dbname here specifies the databa
In SQLlite, if the DB doesn't exist, it will be created.
2. Cursors: A cursor is an object associated with a database connectio
It allows you to execute SQL queries, fetch query results.
"""
conn = sqlite3.connect(dbname)
cursor = conn.cursor()

# Create the Users table
cursor.execute("""
CREATE TABLE IF NOT EXISTS Users (
    user_id INTEGER PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE
);
""")

# Create the Songs table
cursor.execute("""
CREATE TABLE IF NOT EXISTS Songs (
    song_id INTEGER PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    artist VARCHAR(100) NOT NULL,
    genre VARCHAR(100)
);
""")

# Create the Listens table
cursor.execute("""
CREATE TABLE IF NOT EXISTS Listens (
    listen_id INTEGER PRIMARY KEY,
    user_id INTEGER NOT NULL,
    song_id INTEGER NOT NULL,
    rating FLOAT,
    listen_time TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (song_id) REFERENCES Songs(song_id)
);
""")

# Create the recommendations table
cursor.execute("""
CREATE TABLE IF NOT EXISTS Recommendations (
    user_id INTEGER NOT NULL,
    song_id INTEGER NOT NULL,
    recommendation_id not NULL,
    recommendation_time TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (song_id) REFERENCES Songs(song_id)
);
""")


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

In [4]:

```
# Connect to database again and insert sample data
conn = sqlite3.connect(dbname)
sqlite3.enable_callback_tracebacks(True)

cursor = conn.cursor()
cursor.execute("delete from Songs;")
cursor.execute("delete from Users;")
cursor.execute("delete from Listens;")
cursor.execute("delete from Recommendations;")

# Insert sample users
cursor.execute("""
INSERT INTO Users (user_id, name, email)
VALUES
    (1, 'Mickey', 'mickey@example.com'),
    (2, 'Minnie', 'minnie@example.com'),
    (3, 'Daffy', 'daffy@example.com'),
    (4, 'Pluto', 'pluto@example.com');
""")

# Insert sample songs from Taylor Swift, Ed Sheeran, Beatles
cursor.execute("""
INSERT INTO Songs (song_id, title, artist, genre)
VALUES
    (1, 'Evermore', 'Taylor Swift', 'Pop'),
    (2, 'Willow', 'Taylor Swift', 'Pop'),
    (3, 'Shape of You', 'Ed Sheeran', 'Rock'),
    (4, 'Photograph', 'Ed Sheeran', 'Rock'),
    (5, 'Shivers', 'Ed Sheeran', 'Rock'),
    (6, 'Yesterday', 'Beatles', 'Classic'),
    (7, 'Yellow Submarine', 'Beatles', 'Classic'),
    (8, 'Hey Jude', 'Beatles', 'Classic'),
    (9, 'Bad Blood', 'Taylor Swift', 'Rock'),
    (10, 'DJ Mix', 'DJ', NULL);
""")

# Insert sample listens
cursor.execute("""
INSERT INTO Listens (listen_id, user_id, song_id, rating)
VALUES
    (1, 1, 1, 4.5),
    (2, 1, 2, 4.2),
    (3, 1, 6, 3.9),
    (4, 2, 2, 4.7),
    (5, 2, 7, 4.6),
    (6, 2, 8, 3.9),
    (7, 3, 1, 2.9),
    (8, 3, 2, 4.9),
    (9, 3, 6, NULL);
""")
# Commit changes and close the connection
conn.commit()
conn.close()

runSql('Users', "SELECT * FROM Users;")
runSql('Songs', "SELECT * FROM Songs;")
runSql('Listens', "SELECT * FROM Listens;")
runSql('filter users', "SELECT email FROM Users WHERE name='Mickey';")
```

Users

user_id	name	email
1	Mickey	mickey@example.com
2	Minnie	minnie@example.com
3	Daffy	daffy@example.com
4	Pluto	pluto@example.com

Songs

song_id	title	artist	genre
1	Evermore	Taylor Swift	Pop
2	Willow	Taylor Swift	Pop
3	Shape of You	Ed Sheeran	Rock
4	Photograph	Ed Sheeran	Rock
5	Shivers	Ed Sheeran	Rock
6	Yesterday	Beatles	Classic
7	Yellow Submarine	Beatles	Classic
8	Hey Jude	Beatles	Classic
9	Bad Blood	Taylor Swift	Rock
10	DJ Mix	DJ	None

Listens

listen_id	user_id	song_id	rating	listen_time
1	1	1	4.5	None
2	1	2	4.2	None
3	1	6	3.9	None
4	2	2	4.7	None
5	2	7	4.6	None
6	2	8	3.9	None
7	3	1	2.9	None
8	3	2	4.9	None
9	3	6	NaN	None

filter users

email
mickey@example.com

In-class exercise: Update the email address for user with the name 'Daffy' and delete all listening records for the same user.

```
In [6]: runSqlWithCommit('Update Daffy Email', "UPDATE Users SET email = 'daff
```

```
In [7]: runSql('Users', "SELECT * FROM Users;")
```

Users

user_id	name	email
1	Mickey	mickey@example.com
2	Minnie	minnie@example.com
3	Daffy	daffynewemail@example.com
4	Pluto	pluto@example.com

```
In [9]: runSqlWithCommit('Listens', "SELECT user_id FROM Users WHERE Name = 'D
```

Listens

user_id
3

```
In [10]: runSqlWithCommit('Delete Data for Daffy', "DELETE FROM Listens WHERE u
In [11]: runSql('Listens', "SELECT * FROM Listens;")
```

Listens

listen_id	user_id	song_id	rating	listen_time
1	1	1	4.5	None
2	1	2	4.2	None
3	1	6	3.9	None
4	2	2	4.7	None
5	2	7	4.6	None
6	2	8	3.9	None

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```