Machine Learning Foundation

Section 1, Part a: Reading Data

Learning Objective(s)

- Create a SQL database connection to a sample SQL database, and read records from that database
- Explore common input parameters

Packages

- Pandas
- Pandas.read_sql
- SQLite3

Simple data reads

Structured Query Language (SQL) is an ANSI specification, implemented by various databases. SQL is a powerful format for interacting with large databases efficiently, and SQL allows for a consistent experience across a large market of databases. We'll be using sqlite, a lightweight and somewhat restricted version of sql for this example. sqlite uses a slightly modified version of SQL, which may be different than what you're used to.

In [1]:

```
# Imports
import sqlite3 as sq3
import pandas.io.sql as pds
import pandas as pd
```

Database connections

Our first step will be to create a connection to our SQL database. A few common SQL databases used with Python include:

- Microsoft SQL Server
- Postgres
- MySQL
- AWS Redshift
- AWS Aurora
- Oracle DB
- Terradata
- Db2 Family
- Many, many others

Each of these databases will require a slightly different setup, and may require credentials (username & password), tokens, or other access requirements. We'll be using `sqlite3` to connect to our database, but other connection packages include:

- `SQLAlchemy` (most common)

- psycopg2
- MySQLdb

In [2]:

```
# Initialize path to SQLite database
path = 'data/classic_rock.db'
con = sq3.Connection(path)

# We now have a live connection to our SQL database
```

Reading data

Now that we've got a connection to our database, we can perform queries, and load their results in as Pandas DataFrames

In [3]:

```
# Write the query
query = '''
SELECT *
FROM rock_songs;
'''

# Execute the query
observations = pds.read_sql(query, con)

observations.head()
```

Out[3]:

	Song	Artist	Release_Year	PlayCount
0	Caught Up in You	.38 Special	1982.0	82
1	Hold On Loosely	.38 Special	1981.0	85
2	Rockin' Into the Night	.38 Special	1980.0	18
3	Art For Arts Sake	10cc	1975.0	1
4	Kryptonite	3 Doors Down	2000.0	13

In [4]:

```
# We can also run any supported SQL query
# Write the query
query = '''
SELECT Artist, Release_Year, COUNT(*) AS num_songs, AVG(PlayCount) AS avg_plays
    FROM rock_songs
    GROUP BY Artist, Release_Year
    ORDER BY num_songs desc;
'''

# Execute the query
observations = pds.read_sql(query, con)

observations.head()
```

Out[4]:

	Artist	Release_Year	num_songs	avg_plays
0	The Beatles	1967.0	23	6.565217
1	Led Zeppelin	1969.0	18	21.000000
2	The Beatles	1965.0	15	3.800000
3	The Beatles	1968.0	13	13.000000
4	The Beatles	1969.0	13	15.000000

Common parameters

There are a number of common paramters that can be used to read in SQL data with formatting:

- coerce_float: Attempt to force numbers into floats
- parse_dates: List of columns to parse as dates
- chunksize: Number of rows to include in each chunk

Let's have a look at using some of these parameters

In [5]:

```
query='''
SELECT Artist, Release_Year, COUNT(*) AS num_songs, AVG(PlayCount) AS avg_plays
    FROM rock_songs
    GROUP BY Artist, Release_Year
    ORDER BY num_songs desc;
'''

# Execute the query
observations_generator = pds.read_sql(query,
                            con,
                            coerce_float=True, # Doesn't efefct this dataset, because floa
                            parse_dates=['Release_Year'], # Parse `Release_Year` as a date
                            chunksize=5 # Allows for streaming results as a series of shor
                           )

for index, observations in enumerate(observations_generator):
    if index < 5:
        print(f'Observations index: {index}'.format(index))
        display(observations)
```

```
Observations index: 0
```

	Artist	Release_Year	num_songs	avg_plays
0	The Beatles	1970-01-01 00:32:47	23	6.565217
1	Led Zeppelin	1970-01-01 00:32:49	18	21.000000
2	The Beatles	1970-01-01 00:32:45	15	3.800000
3	The Beatles	1970-01-01 00:32:48	13	13.000000
4	The Beatles	1970-01-01 00:32:49	13	15.000000

```
Observations index: 1
```

	Artist	Release_Year	num_songs	avg_plays
0	Led Zeppelin	1970-01-01 00:32:50	12	13.166667
1	Led Zeppelin	1970-01-01 00:32:55	12	14.166667
2	Pink Floyd	1970-01-01 00:32:59	11	41.454545
3	Pink Floyd	1970-01-01 00:32:53	10	29.100000
4	The Doors	1970-01-01 00:32:47	10	28.900000

```
Observations index: 2
```

	Artist	Release_Year	num_songs	avg_plays
0	Fleetwood Mac	1970-01-01 00:32:57	9	35.666667
1	Jimi Hendrix	1970-01-01 00:32:47	9	24.888889

	Artist	Release_Year	num_songs	avg_plays
2	The Beatles	1970-01-01 00:32:43	9	2.444444
3	The Beatles	1970-01-01 00:32:44	9	3.111111
4	Elton John	1970-01-01 00:32:53	8	18.500000

```
Observations index: 3
```

	Artist	Release_Year	num_songs	avg_plays
0	Led Zeppelin	1970-01-01 00:32:51	8	47.750000
1	Led Zeppelin	1970-01-01 00:32:53	8	34.125000
2	Boston	1970-01-01 00:32:56	7	69.285714
3	Rolling Stones	1970-01-01 00:32:49	7	36.142857
4	Van Halen	1970-01-01 00:32:58	7	51.142857

```
Observations index: 4
```

	Artist	Release_Year	num_songs	avg_plays
0	Bruce Springsteen	1970-01-01 00:32:55	6	7.666667
1	Bruce Springsteen	1970-01-01 00:33:04	6	11.500000
2	Creedence Clearwater Revival	1970-01-01 00:32:49	6	23.833333
3	Creedence Clearwater Revival	1970-01-01 00:32:50	6	18.833333
4	Def Leppard	1970-01-01 00:33:07	6	32.000000