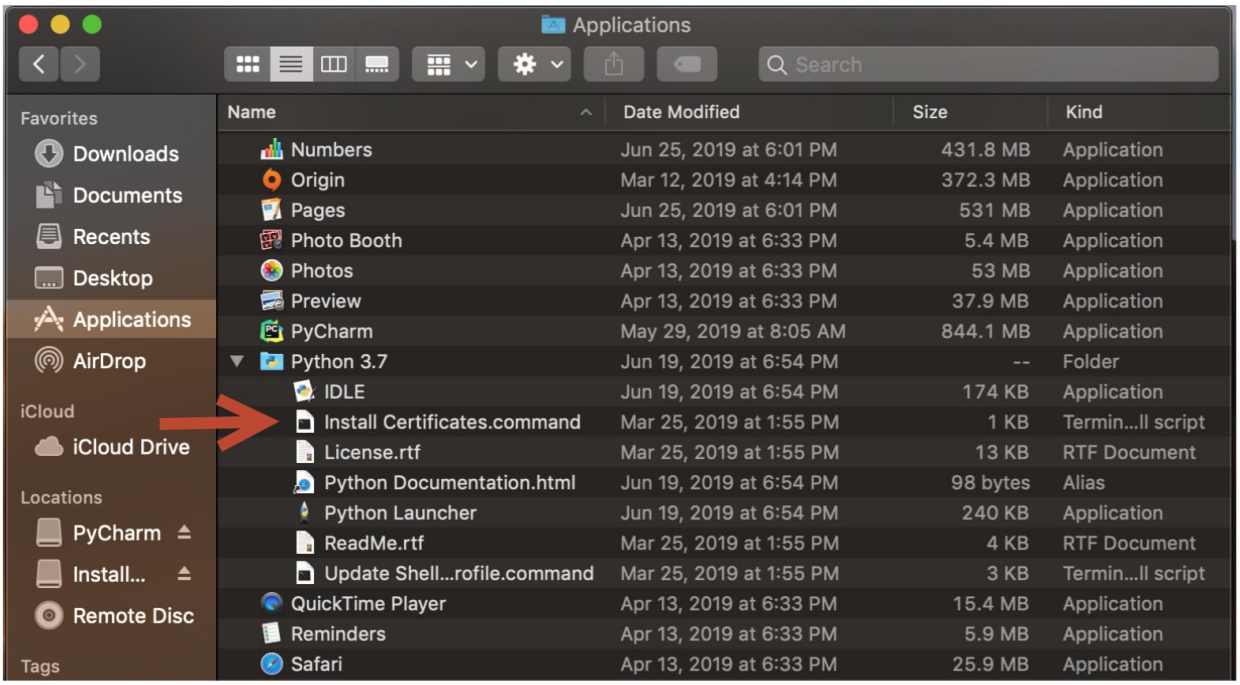
Assignment 16 – Databases

For Mac Book Users (SSL: CERTIFICATE_VERIFY_FAILED)

If you're using a mac book for this course and get an SSL: CERTIFICATE_VERIFY_FAILED error, you will need to update the certificate.

1. Navigate to: /Applications/Python/3.*x*/
2. Click on Install Certificates.command



DB Browser for SQLite

The book uses the SQLite Manager for Firefox which Firefox no longer supports. In it's place, you can use DB Browser for SQLite. Instruction for downloading and installing DB Browser for SQLite can be found in the db_browser_addendum.

You will not be required to download DB Browser for SQL for this assignment, but we will be covering it in class, and you have the link and can download and create and view databases and tables on your own if you wish.

Movie List Program

In this assignment, we'll create a movie database application using a sqlite database that is provided.

Set up the Project

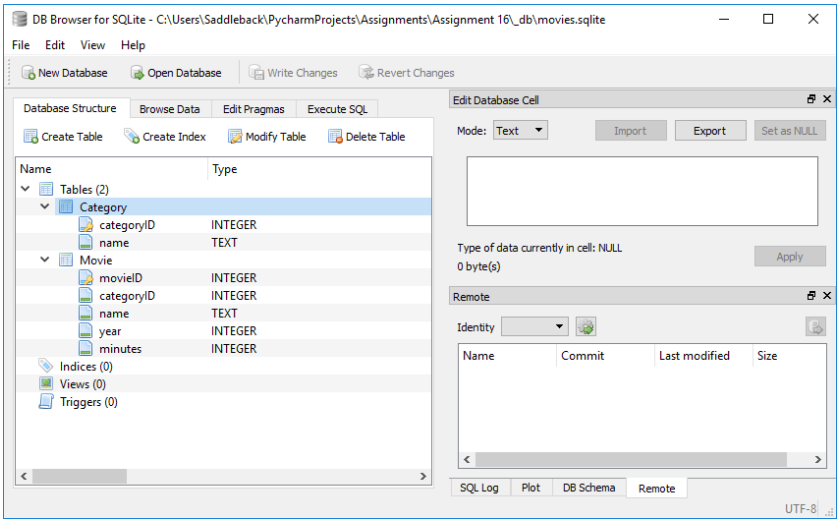
1. Create a project (or folder) named **movie**
   This is where you will keep all the files for the application

Set up the Database

2. Create a folder named **_db**
3. Download the **movies.sqlite** from Canvas into the **_db** folder.

movies.sqlite

The movies database consists of two pre-populated tables, the Category table and the Movie table. The Category table consists of an id and the category name and the movie table consists of an id, category id, movie name, year released and running time minutes.



Movies.sqlite records

The categories table is populated with the following records.

	categoryID	name
	Filter	Filter
1	1	Animation
2	2	Comedy
3	3	History

The movies table is populated with the following records.

	movieID	categoryID	name	year	minutes
	Filter	Filter	Filter	Filter	Filter
1	1	1	Spirit: Stallion of the Cimarron	2002	83
2	2	1	Spirited Away	2001	125
3	3	1	Aladdin	1992	90
4	4	1	Ice Age	2002	81
5	5	1	Toy Story	1995	81
6	6	2	Monty Python and the Holy Grail	1975	91
7	7	2	Monty Python's Life of Brian	1979	94
8	8	2	Monty Python's The Meaning of Life	1983	107
9	9	3	Gandhi	1982	191
10	10	3	Jinnah	1998	110
11	11	3	Lawrence of Arabia	1962	216
12	12	3	Hotel Rwanda	2004	121
13	13	3	Twelve Years a Slave	2013	134

Create the Objects class

We will need to create record objects to be able to work with each table. The first object we will create will be for the Movie class and it will include an id (primary key), movie name, release year, running time in minutes and a category id (foreign key).

4. Create a new python file **objects.py**
5. Add the Movie class as follows:

```
1   class Movie:
2       def __init__(self, id=0, name=None, year=0, minutes=0, category=None):
3           self.id = id
4           self.name = name
5           self.year = year
6           self.minutes = minutes
7           self.category = category
8
```

The second class we will create will be for the Category class and will just include an id (primary key) and the category name.

6. Now add a Category class as follows"

```
9
10  class Category:
11      def __init__(self, id=0, name=None):
12          self.id = id
13          self.name = name
14
```

Creating the db Class

The db class will have all the functionality required to work with the database. The first things we need will be the imports. Since our application will be using a sqlite database, we will need to import sqlite3. We'll also include the import for the closing function from the contextlib module to be able to use datasets (cursor) and of course the Movie and Category class we previously created.

7. Create a new python file **db.py**
8. Add the following imports and a global "conn" object as shown:

```
1  import sqlite3
2  from contextlib import closing
3
4  from objects import Category
5  from objects import Movie
6
7  conn = None
8
```

Next, we will add a method for connecting to the database. We will first need a global to store the connection. If we are not connected, connect to the database.

9. Crate the connect method as follows:

```
9
10 def connect():
11     global conn
12     if not conn:
13         db_file = "_db/movies.sqlite"
14
15         conn = sqlite3.connect(db_file)
16         conn.row_factory = sqlite3.Row
17
```

Now we will create the method to close the connection

10. Create close method as follows:

```
18
19 def close():
20     if conn:
21         conn.close()
22
```

We will want to create Movie and Category objects from fields returned from a query so that will be next.

11. Add the make_movie and make_category methods as follows:

```
23
24      def make_category(row):
25          return Category(row["categoryID"], row["categoryName"])
26
27
28      def make_movie(row):
29          return Movie(row["movieID"], row["name"], row["year"], row["minutes"],
30                       make_category(row))
31
```

We now have the db class. We'll return to add the functionality but let's first setup the user interface for our application.

Setting up the User Interface

The ui class will contain all the ui methods as well as the logic of the flow control for the application.

We'll first need to import our db class so we have access to all of the db functionality and we'll also need to add the Movie class since we'll be working with Movie objects.

12. Create a new python file **ui.py**
13. Add the following imports as shown:

```
1   #!/usr/bin/env/python3
2
3   import db
4   from objects import Movie
5
```

Next, we'll add functions to display the application title as well as the menu options.

14. Code as follows:

```
6
7   def display_title():
8       print("The Movie List program")
9       print()
10      display_menu()
11
12
13  def display_menu():
14      print("COMMAND MENU")
15      print("cat  - View movies by category")
16      print("year - View movies by year")
17      print("add  - Add a movie")
18      print("del  - Delete a movie")
19      print("exit - Exit program")
20      print()
21
```

Now let's create the main method with some of the basics so we can validate opening and closing the database work without errors.

15. Code the main as follows:

```
22
23  def main():
24      # Open the database connection
25      db.connect()
26
27      # display the application title and menu options
28      display_title()
29
30      # close the database connection
31      db.close()
32
```

Finally, we'll need to add the ability to run the application.

16. Code as follows:

```
33
34  if __name__ == "__main__":
35      main()
36
```

Test the Application

17. Run the application:

```
"C:\Users\Saddleback\PycharmProjects\
The Movie List program

COMMAND MENU
cat  - View movies by category
year - View movies by year
add  - Add a movie
del  - Delete a movie
exit - Exit program

Process finished with exit code 0
```

We got the menu to display so now let's start adding to the functionality!

Display the Category List

We'll first display the category list as part of the menu.

18. In the **db.py** file, add the following that get the category list from the database.

```
32
33  def get_categories():
34      # create the SQL
35      query = '''SELECT categoryID, name as categoryName
36                FROM Category'''
37
38      # retrieve the records from the category table
39      with closing(conn.cursor()) as c:
40          c.execute(query)
41          results = c.fetchall()
42
43      # build a categories tuple
44      categories = []
45      for row in results:
46          categories.append(make_category(row))
47
48      # return the category list
49      return categories
50
```

19. In the **ui.py** file, add the following to display the category list.

```
22
23  def display_categories():
24      print("CATEGORIES")
25      categories = db.get_categories()
26      for category in categories:
27          print(str(category.id) + ". " + category.name)
28      print()
29
```

20. And now from main, call the function to display the categories after the menu options.

```
30  def main():
31      # Open the database connection
32      db.connect()
33
34      # display the application title an
35      display_title()
36      display_categories()
37
38      # close the database connection
```

Test

21. Run the application

Screen Capture #1 (3 points)

```
"C:\Users\Saddleback\PycharmProjects
The Movie List program

COMMAND MENU
cat  - View movies by category
year - View movies by year
add  - Add a movie
del  - Delete a movie
exit - Exit program

CATEGORIES
1. Animation
2. Comedy
3. History

Process finished with exit code 0
```

Display Movies by Category

To display movies by a category, we'll first need to prompt the user to enter a category and then retrieve and display any movies for the selected category.

We'll start by adding a function to retrieve a category. We'll do this to determine if the user enter a valid category.

22. In **db.py**, code as follows:

```
51
52  def get_category(category_id):
53      # create the SQL
54      query = '''SELECT categoryID, name as categoryName
55                FROM Category WHERE categoryID = ?'''
56
57      # retrieve the record from the category table
58      with closing(conn.cursor()) as c:
59          c.execute(query, (category_id,))
60          row = c.fetchone()
61
62      # if found, return the category
63      if row:
64          return make_category(row)
65      else:
66          return None
67
```

Next, we'll add a function to retrieve a movie list for a selected category.

23. In **db.py**, code as follows:

```
68
69  def get_movies_by_category(category_id):
70      # create the SQL
71      query = '''SELECT movieID, Movie.name, year, minutes,
72                        Movie.categoryID as categoryId,
73                        Category.name as categoryName
74                FROM Movie JOIN Category
75                        ON Movie.categoryID = Category.categoryID
76                WHERE Movie.categoryID = ?'''
77
78      # retrieve the records from the movie and category tables
79      with closing(conn.cursor()) as c:
80          c.execute(query, (category_id,))
81          results = c.fetchall()
82
83      # build the movies list
84      movies = []
85      for row in results:
86          movies.append(make_movie(row))
87
88      # return movies list
89      return movies
90
```

For displaying, we'll crate a generic function to display a movie list.

24. In **ui.py** add the following:

```
30
31  def display_movies(movies, title_term):
32      print("MOVIES - " + title_term)
33      line_format = "{:3s} {:37s} {:6s} {:5s} {:10s}"
34      print(line_format.format("ID", "Name", "Year", "Mins", "Category"))
35      print("-" * 64)
36      for movie in movies:
37          print(line_format.format(str(movie.id), movie.name,
38                                   str(movie.year), str(movie.minutes),
39                                   movie.category.name))
40      print()
41
```

We'll also need to add the function to get and display the movie list by category.

25. In **ui.py** add the following:

```
42
43  def display_movies_by_category():
44      category_id = int(input("Category ID: "))
45      category = db.get_category(category_id)
46      if category is None:
47          print("There is no category with that ID.\n")
48      else:
49          print()
50          movies = db.get_movies_by_category(category_id)
51          display_movies(movies, category.name.upper())
52
```

And finally, we'll add the menu functionality to the main application flow.

```
60      display_categories()
61
62      # main application flow
63      while True:
64          command = input("Command: ")
65          if command == "cat":
66              # display movies by category
67              display_movies_by_category()
68
69          elif command == "exit":
70              # break out of the loop to exit the application
71              break
72          else:
73              print("Not a valid command. Please try again.\n")
74              display_menu()
75
76      # close the database connection
77      db.close()
```

Test

26. Run the application
27. Display the movies by the comedy category

Screen Capture #2 (5 points)

```
Command: cat
Category ID: 2

MOVIES - COMEDY
ID  Name                                    Year   Mins  Category
-----------------------------------------------------------------
6   Monty Python and the Holy Grail         1975   91    Comedy
7   Monty Python's Life of Brian            1979   94    Comedy
8   Monty Python's The Meaning of Life      1983   107   Comedy

Command:
```

Display Movies by Year

For displaying a movie by year, we're just going to repeat the process from displaying a movie by category with just some minor changes.

We'll start by adding a function to retrieve a movie list for a selected year.

28. In **db.py**, code as follows:

```
90
91  def get_movies_by_year(year):
92      query = '''SELECT movieID, Movie.name, year, minutes,
93                        Movie.categoryID as categoryId,
94                        Category.name as categoryName
95                 FROM Movie JOIN Category
96                     On Movie.categoryID = Category.categoryID
97                 WHERE Movie.year = ?'''
98
99      with closing(conn.cursor()) as c:
100         c.execute(query, (year,))
101         results = c.fetchall()
102
103     movies = []
104     for row in results:
105         movies.append(make_movie(row))
106
107     return movies
108
```

We'll also need to add the function to get and display the movie list by year.

29. In **ui.py** add the following:

```
53
54  def display_movies_by_year():
55      year = int(input("Year: "))
56      print()
57      movies = db.get_movies_by_year(year)
58      display_movies(movies, str(year))
59
```

And finally, we'll add the menu functionality to the main application flow.

```
73          # display movies by category
74          display_movies_by_category()
75
76      elif command == "year":
77          # display movies by year
78          display_movies_by_year()
79
80      elif command == "exit":
81          # break out of the loop to ex
```

Test

30. Display the movies by year 2002

Screen Capture #3 (5 points)

```
Command: year
Year: 2002

MOVIES - 2002
ID  Name                                    Year   Mins  Category
-------------------------------------------------------------------
1   Spirit: Stallion of the Cimarron        2002   83    Animation
4   Ice Age                                 2002   81    Animation

Command:
```

Add a Movies

We'll start by adding a function to add a movie to the movies table.

31. In **db.py**, code as follows:

```
114
115 def add_movie(movie):
116     # create the SQL
117     sql = '''INSERT INTO Movie (categoryID, name, year, minutes)
118              VALUES (?, ?, ?, ?)'''
119
120     # add the movie to the table
121     with closing(conn.cursor()) as c:
122         c.execute(sql, (movie.category.id, movie.name, movie.year,
123                         movie.minutes))
124         conn.commit()
125
```

We'll also need to add the function to get the movie information from the user.

32. In **ui.py** add the following:

```
60
61 def add_movie():
62     name = input("Name: ")
63     year = int(input("Year: "))
64     minutes = int(input("Minutes: "))
65     category_id = int(input("Category ID: "))
66
67     category = db.get_category(category_id)
68     if category is None:
69         print("There is no category with that ID. Movie NOT added.\n")
70     else:
71         movie = Movie(name=name, year=year, minutes=minutes,
72                       category=category)
73         db.add_movie(movie)
74         print(name + " was added to the database.\n")
75
```

And finally, we'll add the menu functionality to the main application flow.

```
94          display_movies_by_year(
95
96      elif command == "add":
97          # add a movie
98          add_movie()
99
100     elif command == "exit":
```

Test

33. Add a movie
    a. Make sure the view the category to prove it was added

Screen Capture #4 (5 points)

```
Command: add
Name: Ferris Bueller's Day Off
Year: 1986
Minutes: 103
Category ID: 2
Ferris Bueller's Day Off was added to the database.

Command: cat
Category ID: 2

MOVIES - COMEDY
ID  Name                                    Year   Mins  Category
-------------------------------------------------------------------
6   Monty Python and the Holy Grail         1975   91    Comedy
7   Monty Python's Life of Brian            1979   94    Comedy
8   Monty Python's The Meaning of Life      1983   107   Comedy
14  Ferris Bueller's Day Off                1986   103   Comedy
```

Delete a Movies

We'll start by adding a function to delete a movie from the movies table.

34. In **db.py**, code as follows:

```
126
127     def delete_movie(movie_id):
128         # create the SQL
129         sql = '''DELETE FROM Movie WHERE movieID = ?'''
130
131         # Delete the movie from the table
132         with closing(conn.cursor()) as c:
133             c.execute(sql, (movie_id,))
134             conn.commit()
135
```

We'll also need to add the function to get the movie information from the user.

35. In **ui.py** add the following:

```
76
77  def delete_movie():
78      movie_id = int(input("Enter Movie ID to delete: "))
79      db.delete_movie(movie_id)
80      print("Movie ID " + str(movie_id) + " was deleted from database.\n")
81
```

And finally, we'll add the menu functionality to the main application flow.

Test

36. Delete a movie
    b. Make sure the view the category before and after the delete to prove it was deleted.

Screen Capture #5 (5 points)

```
Command: cat
Category ID: 2

MOVIES - COMEDY
ID  Name                                    Year   Mins  Category
-----------------------------------------------------------------
6   Monty Python and the Holy Grail         1975   91    Comedy
7   Monty Python's Life of Brian            1979   94    Comedy
8   Monty Python's The Meaning of Life      1983   107   Comedy
14  Ferris Bueller's Day Off                1986   103   Comedy

Command: del
Enter Movie ID to delete: 7
Movie ID 7 was deleted from database.

Command: cat
Category ID: 2

MOVIES - COMEDY
ID  Name                                    Year   Mins  Category
-----------------------------------------------------------------
6   Monty Python and the Holy Grail         1975   91    Comedy
8   Monty Python's The Meaning of Life      1983   107   Comedy
14  Ferris Bueller's Day Off                1986   103   Comedy
```

Extra Credit

To get full points for each extra credit, you must include screen captures of the running output as well as the python (.py) code files.

Extra Credit #1 – Custom Data Importer (+2 Extra Credit)

Create a program that imports data from a CSV file into a database table.

```
Customer Data Importer

CSV file:    customers.csv
DB file:     customers.sqlite
Table name: Customer

All old rows deleted from Customer table.
500 row(s) inserted into Customer table.
```

Specifications:

- You can download the required CSV and database files (customer.csv and customers.sqlite) from the assignment 16 section in Canvas. The SQLite database file should contain a table named Customer.
- The program should begin by deleting any old data from the Customer table. Then, it should insert all data from the customers.csv file into the Customer table of the SQLite database.
- The CSV file should be in this format:
  first_name,last_name,company_name,address,city,state,zip
  James,Butler,,N Blue Gum St,New Orleans,LA,70116
  Josphine,Darakjy,,4 B Blue Ridge Blvd,Brighton,MI,48116
  Art,Venere,,8 W Cerritos Ave #54,Bridgeport,NJ,08014
  Lenna,Paprocki,Feltz Printing,639 Main St,Anchorage,AK,99501
- The Customer table should have the following columns and data types:

customerID	INTEGER PRIMARY KEY
firstName	TEXT
lastName	TEXT
companyName	TEXT
address	TEXT
city	TEXT
state	TEXT
zip	TEXT

- The program should conclude by displaying the screen from above.