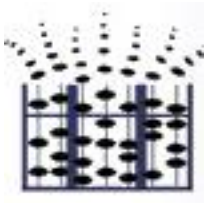

CIS 211
Spring 2014

Project 6: SQLite Databases

Due Wednesday May 21
Upload via Blackboard by 11:00 P.M.

Goals

This project will give you some experience writing database queries in SQL and connecting to a SQLite database from Python.

Reading

Introduction to Computing Using Python, Sec 12.1 -- 12.2, plus resources available on the class web site.

Queries

Write a query that will answer each of the following questions based on the data in a SQLite database named `sakila.db`. Test your queries using the `sqlite3` command line client. Then ***copy and paste the queries and the output they produce*** into a document called `queries.txt`.

NOTE: some queries (like the first one) will produce hundreds of lines of output. In these cases add "limit 5" to the end of your query.

***Note**: Names, titles, and other strings in this database are in all caps, and sqlite3 string comparisons are case sensitive.*

1. What are the first and last names of all the customers?
2. What are the e-mail addresses of customers whose first name is "Willard"?
3. How many customers are there?
4. How many customers shop at store number 1?
5. How much does it cost to rent the film named "Virtual Spoilers"?
6. Do any films have the word "Princess" in the title?
7. What are the titles of the films that are longer than 180 minutes?
8. How many films have a "G" rating and are less than 60 minutes long?
9. What is the maximum replacement cost for any film?
10. Print a table that lists the different types of ratings and the number of films that have that rating.

The following queries all require a join of two or more tables. As a hint for how to create the query the table names are shown to the left of a question.

film, film_actor

11. How many actors starred in the film named “Scalawag Duck”?

actor, film_actor

12. What are the film IDs of movies starring Jude Cruise?

film, language

13. What language was “Ace Goldfinger” filmed in?

customer, rental

14. Produce a table that shows how many films each customer has rented.

customer, rental

15. How many films did the customer named Smith rent?

customer, rental

16. Which customers have not returned films (i.e. the `return_date` field in the rental table is null)?

film, actor, film_actor

17. Which actors starred in “Splash Gump”?

film, actor, film_actor

18. Which films has Penelope Guiness starred in?

customer, rental, inventory, film

19. What are the names of the films rented by the customer named Knight?

Programming Project

Write a Python program named `summary.py` that will print a report showing the monthly bill for a customer at the Sakila DVD Store. The program will be called from the command line; arguments will specify the customer's last name, a year, and a month. The program should fetch the necessary information from the database and print a monthly bill for that customer.

Here is an example that shows the bill for the customer named "Ebert" for the month of June, 2005:

```
$ python3 summary.py ebert 2005 6

--- Sakila DVD Rentals ---

Monthly report for Leo Ebert

Lonely Elephant            06/16/2005    $2.99
              **late fee   06/24/2005    $2.99
Enemy Odds                 06/18/2005    $4.99
              **late fee   06/27/2005    $4.99
Song Hedwig                06/20/2005    $0.99
              **late fee   06/26/2005    $0.99
Wash Heavenly              06/21/2005    $4.99
Hurricane Affair           06/21/2005    $2.99

Monthly total:  $25.92
```

Your program needs to do two queries to make a report like the one shown above:

- fetch the customer's first and last name so you can print the header ("Monthly report for....")
- fetch all of the rental information for the specified month and year so you can print a list of films the customer rented

All of the information you need for the films, dates, and fees can be retrieved with a single query that joins the rental, inventory, film, and customer tables. Note that this is the same query used to answer Question 19 above, except with different column names.

To compute the total bill, add all the rental fees, and for any film returned late charge an additional rental. In the example shown above, Leo rented *Lonely Elephant* on June 16 but didn't return it until June 24. It was a 3-day rental, so he was charged an additional $2.99 (it was 5 days late, so we could have charged him $4.98 in late fees, but we'll keep it simple and just charge the basic fee as the late fee).

Strategy

We ***strongly*** recommend you follow the incremental design strategy for this project. Start with a very simple version and gradually add parts until the final program is working.

1. Get the arguments from the command line, *i.e.* the first version should do nothing more than print "preparing report for customer *x* for *z y*" where *x*, *y*, and *z* are the three values from the command line.
2. Fetch the first and last name from the database. Now you have enough information to print the actual header lines of the output.
3. Execute the query that gets all the rental information. This version should just print the films without figuring out the total.
4. Add up the basic rental fees, but don't worry about late fees yet. This version can print the monthly total.
5. Write the code that figures out if a film was returned late, and if so, add an additional charge.

Computing Late Fees

To decide whether a film was returned late you need to subtract the rental date from the return date and see if the difference is greater than the rental duration. The easiest way to do this is to use Python's `datetime` library.

Here is the code that will create a `datetime` object from a date string in the Sakila database. If `rent_date` is a string you have fetched from the rental table, use this statement to make a `datetime` object named `rented`:

```
date_format = "%Y-%m-%d %H:%M:%S.%f"
rented = datetime.strptime(rent_date, date_format)
```

The method name `strptime` stands for "string parse time" -- it parses a string to figure out the year, month, *etc.* using the format specification given in the second argument.

Do the same thing for the return date, making an object named `returned`. Then subtract the rental date from the return date to get a `timedelta` object:

```
diff = returned - rented
```

Now you can get attributes of the `diff` object to tell you how far apart the dates were, for example `diff.days` will tell you the difference in days.

Extra Credit Ideas

- Extend the command line API to allow the user to specify either the customer ID or customer name, e.g.

```
$ python3 summary.py -id 466 2005 6
```

- Make the month parameter optional. If no month is given, print a summary for each month of the specified year.
- Fetch the customer's address and other information, print that along with the customer name at the top of the report.
- Determine which store the customer shops at and print the store address at the top of the page.
- [Many extra points] Figure out how to use the `locale` library, and print the rental dates and currency using the locale settings on your computer.

What to Turn In

Documentation Write a short description (two or three paragraphs total) of what you did for your `summary.py` program. This is your chance to tell the graders about anything you think you did well. The documentation should be in a file named `writeup` with an extension that identifies the file format (.doc for Microsoft Word, .pdf for Adobe PDF, .txt for plain text, .rtf for rich text format).

Code Your Python program should be in a single file named `summary.py`.

Create a package (tar or zip format) that includes

1. ***your queries.txt file***
2. ***your summary.py program***
3. ***your writeup***

Upload the package via Blackboard.