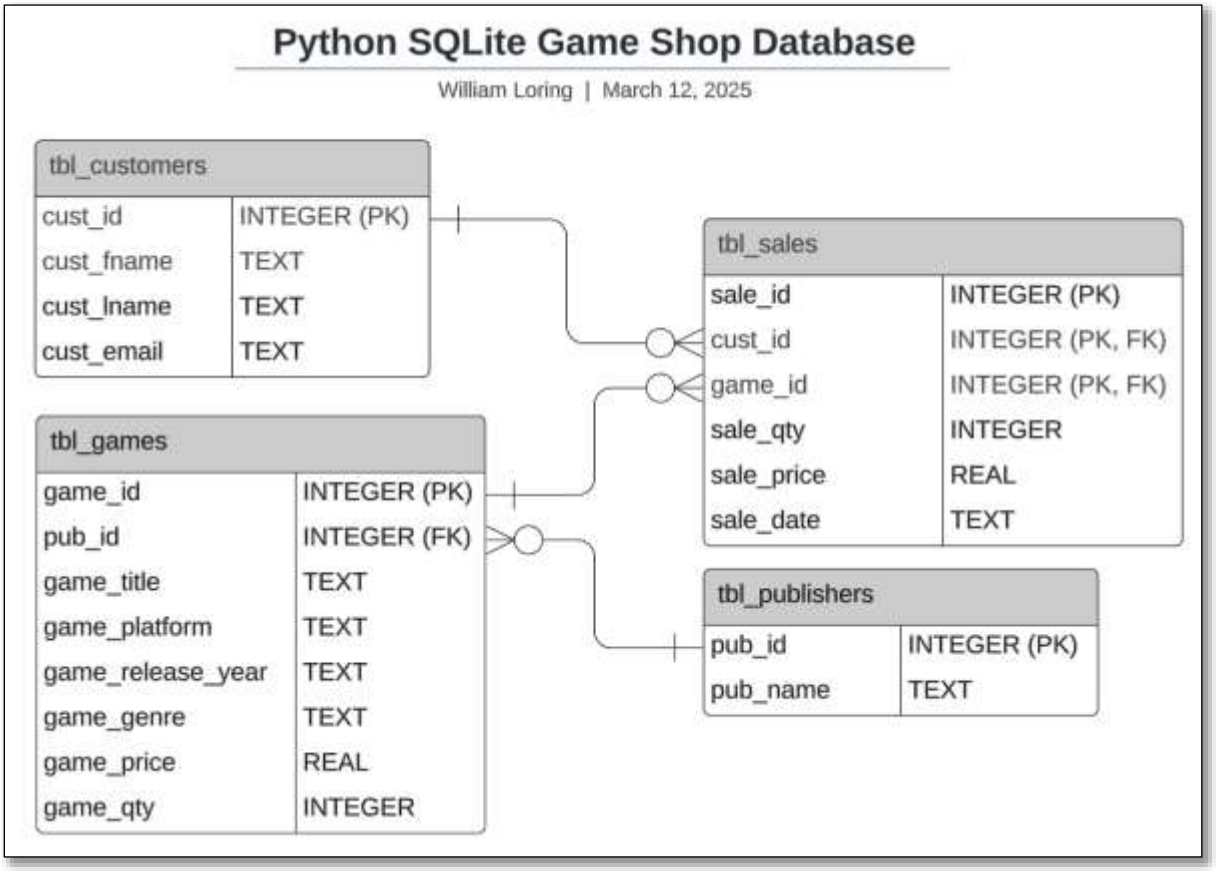
5. Python SQLite Game Shop POS Reports

Contents

Time required: 90 minutes



Reports Menu Prompt

This is the current Game Shop Reports Menu Prompt. Yours will be different as you may not have the publishers table to deal with.

```
REPORTS_PROMPT = """------  Reports  -------
(1) Display Sales Transactions
(2) Sales by Game
(3) Sales by Genre
(4) Sales by Platform
(5) Sales by Publisher
(6) Display Games
(7) Display Customers
(8) Display Publishers
(9) Display Publishers & Games
(99) Exit
Your selection: """
```

Display Sales Transactions

We created this report earlier for our Sales Transactions menu. Add this to the Reports menu.

display_all_sales()

Sales by Game

I have a db_app.py file for queries that are not specific to a particular table.

```
"""
    Name: db_app.py
    Author: William Loring
    Created: 10/07/24
    This module contains functions that interact with more than one table
"""

# Import sqlite3 database library
import sqlite3

DATABASE = "game_shop.db"
```

SQL for Fetch Sales by Games. This SQL will give us our pattern for the other reports from the games table to the sales table.

```
FETCH_SALES_BY_GAME = """
SELECT tbl_games.game_title,
tbl_sales.sale_qty,
tbl_sales.sale_price,
SUM(tbl_sales.sale_price * tbl_sales.sale_qty) AS total_sales
FROM tbl_sales
JOIN tbl_games ON tbl_sales.game_id = tbl_games.game_id
GROUP BY tbl_games.game_title
ORDER BY total_sales DESC
"""
```

SQL Explanation

- **SELECT clause**: Specifies columns to retrieve:
    - tbl_games.game_title - The title of each game
    - tbl_sales.sale_qty - How many of each game was sold
    - tbl_sales.sale_price - Price of each game
    - SUM(tbl_sales.sale_price * tbl_sales.sale_qty) AS total_sales - Calculates the total sales amount for each game and labels it as "total_sales"
- **FROM clause**: Starts with the tbl_sales table as the base table
- **JOIN clause**:
    - JOIN tbl_games ON tbl_sales.game_id = tbl_games.game_id: Links sales records to their corresponding games using the game_id as the common identifier
- **GROUP BY clause**:
    - GROUP BY tbl_games.game_title: Combines all sales records for each unique game title into a single row
- **ORDER BY clause**:
    - ORDER BY total_sales DESC: Sorts the results in descending order of total sales, so the best-selling games appear at the top

Purpose:

This query generates a sales report showing how much revenue each game has generated. It's useful for:

- Identifying the most profitable games
- Comparing sales performance across different titles
- Making inventory and marketing decisions based on sales data
- Reporting on the financial performance of different products

The query transforms individual sales transactions into actionable business intelligence by summarizing the data at the product level.

In **db_app.py**, create the following **fetch_sales_by_game()** function.

```
65  # --------------------- FETCH SALES BY GAME ------------------------------- #
66  def fetch_sales_by_game():
67      with sqlite3.connect(DATABASE) as connection:
68          # Create a cursor object to interact with the database
69          cursor = connection.cursor()
70
71          # A list of tuples. Each tuple is a record/row in the database.
72          records = cursor.execute(FETCH_SALES_BY_GAMES).fetchall()
73
74          return records
```

Add to Point of Sale Program

In the game_shop_pos.py program, add the display_game_sales() function.

```
343 # ---------------------- DISPLAY SALES BY GAME ---------------------------- #
344 def display_game_sales():
345     # Fetch all records from the database
346     sales = db_app.fetch_sales_by_game()
347
348     # Use tabulate library to format the data
349     records = tabulate.tabulate(
350         sales,
351         headers=["Game", "Total Sales"],
352         tablefmt="psql"  # Table format
353     )
354
355     print(records)
```

Add the function to the corresponding menu item.

Example run:

```
------  Game Shop Point of Sale  ------
(1) Sales Transactions
(2) Table Maintenance
(3) Reports
(9) Exit
Your selection: 3
------  Reports  -------
(1) Display Sales Transactions
(2) Sales by Game
(3) Sales by Genre
(4) Sales by Platform
(5) Sales by Publisher
(6) Display Games
(7) Display Customers
(8) Display Publishers
(9) Display Publishers & Games
(99) Exit
Your selection: 2
+--------------------+-------+---------+---------------+
| Game               |   Qty |   Price |   Total Sales |
|--------------------+-------+---------+---------------|
| Sonic the Hedgehog |     2 |   39.98 |         79.96 |
| The Legend of Zelda |     1 |   59.99 |         59.99 |
+--------------------+-------+---------+---------------+
------  Reports  -------
(1) Display Sales Transactions
(2) Sales by Game
(3) Sales by Genre
(4) Sales by Platform
(5) Sales by Publisher
(6) Display Games
(7) Display Customers
(8) Display Publishers
(9) Display Publishers & Games
(99) Exit
Your selection:
```

Assignment 1: Create Your Own Report

Create another report similar to the Sales by Game report. Choose another attribute to report by. In my example above, I added Sales by Genre and Platform.

You would change the following lines of the SQL for the Sales by Game reports.

Change the game_title to the attribute you wish to report on.

- tbl_games.game_title: The title of each game

Change game_title to the same attribute you chose above.

- GROUP BY tbl_games.game_title: Combines all sales records for each unique game title into a single row

Everything else follows the same pattern.

Final Submission

Every menu in your program should now be functional. You may not have all the reports that the Game Shop has, as you may not have a publishers table.

Test your entire program before submission.

Assignment Submission

1. Attach the program files.
2. Attach screenshots showing the successful operation of the program.
3. Submit in Blackboard.