4. Python SQLite Game Shop POS - Sales

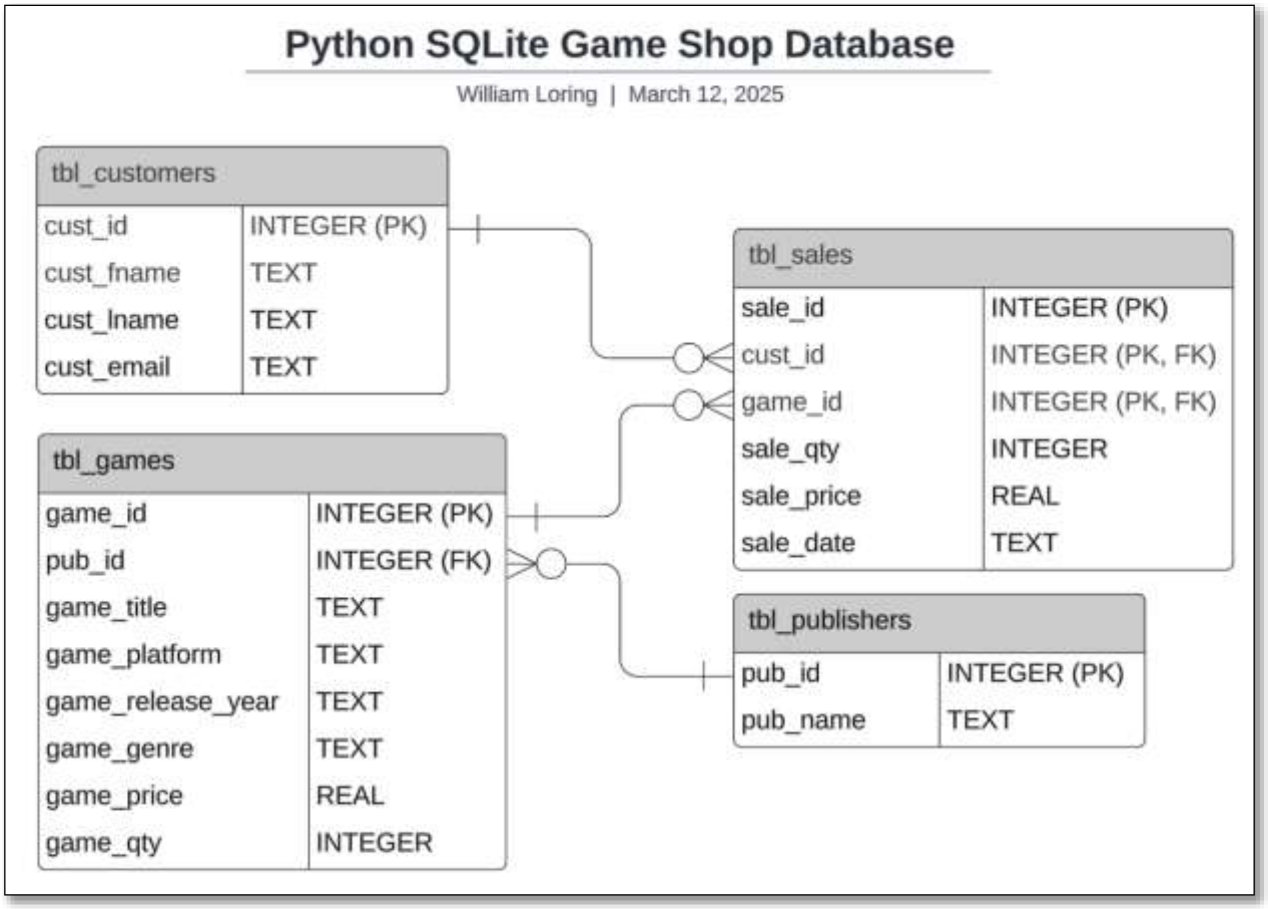
Contents

Time required: 60 minutes

- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Sales

Our program is not much of a Point of Sale (POS) program without sales. Let's take another look at our ERD.



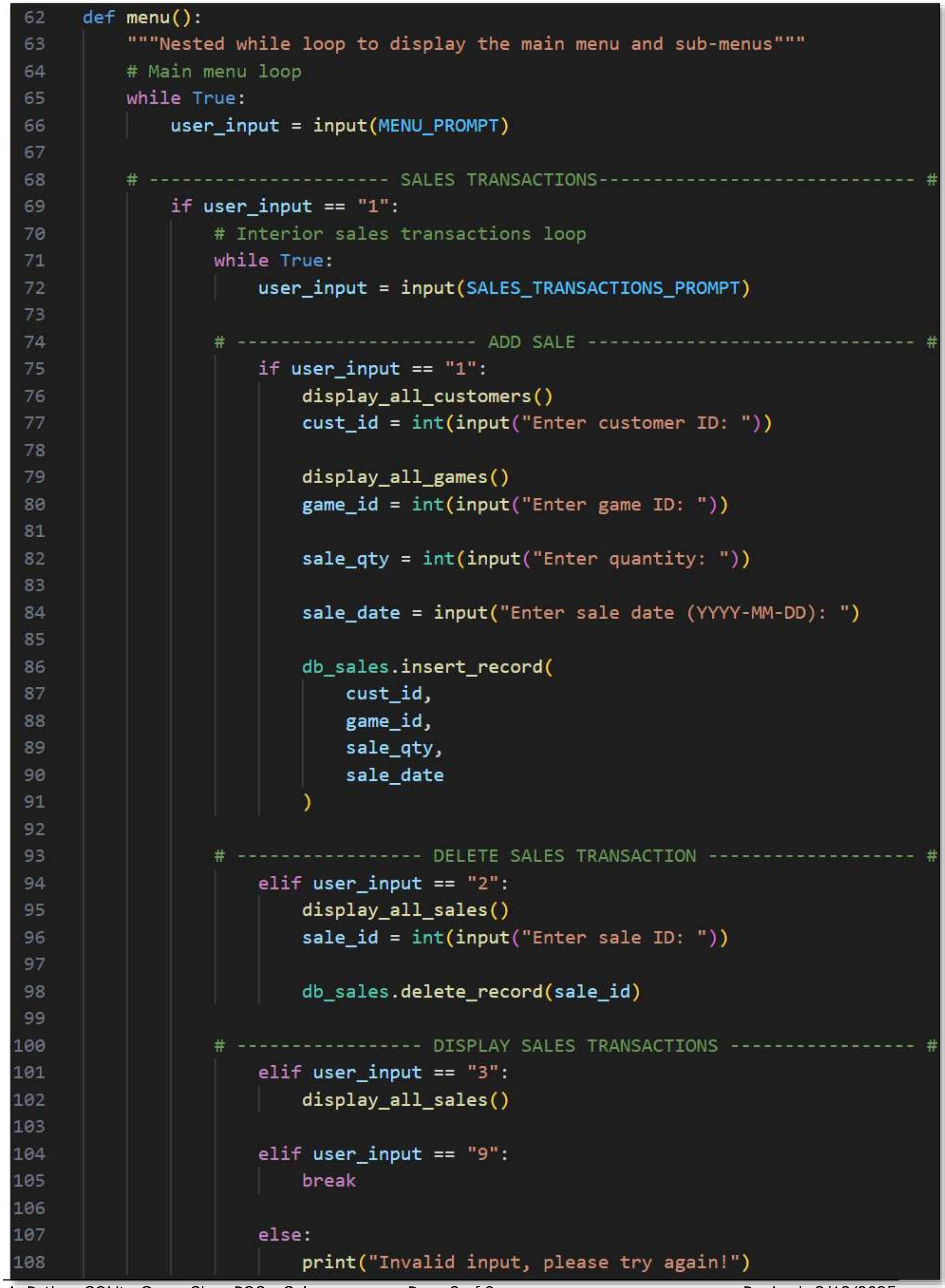
To add a sale, we must know 2 things: the two foreign keys for our sales table.

- cust_id
- game_id

We accomplish this by calling the display_all_customers() and display_all_games() functions before we enter the ID's. After that, we get the quantity and the date.

We will also add db_sales.delete_record() and display_all_sales()

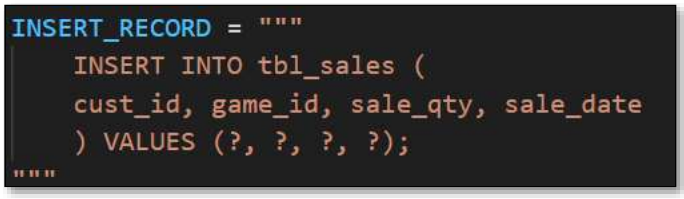
Add the following to your game_shop_pos.py file. This finishes out the Sales menu.


```python
def menu():
    """Nested while loop to display the main menu and sub-menus"""
    # Main menu loop
    while True:
        user_input = input(MENU_PROMPT)

    # ---------------------- SALES TRANSACTIONS----------------------------- #
        if user_input == "1":
            # Interior sales transactions loop
            while True:
                user_input = input(SALES_TRANSACTIONS_PROMPT)

            # ---------------------- ADD SALE ----------------------------- #
                if user_input == "1":
                    display_all_customers()
                    cust_id = int(input("Enter customer ID: "))

                    display_all_games()
                    game_id = int(input("Enter game ID: "))

                    sale_qty = int(input("Enter quantity: "))

                    sale_date = input("Enter sale date (YYYY-MM-DD): ")

                    db_sales.insert_record(
                        cust_id,
                        game_id,
                        sale_qty,
                        sale_date
                    )

            # ----------------- DELETE SALES TRANSACTION ------------------- #
                elif user_input == "2":
                    display_all_sales()
                    sale_id = int(input("Enter sale ID: "))

                    db_sales.delete_record(sale_id)

            # ----------------- DISPLAY SALES TRANSACTIONS ---------------- #
                elif user_input == "3":
                    display_all_sales()

                elif user_input == "9":
                    break

                else:
                    print("Invalid input, please try again!")
```

Insert Sales Record

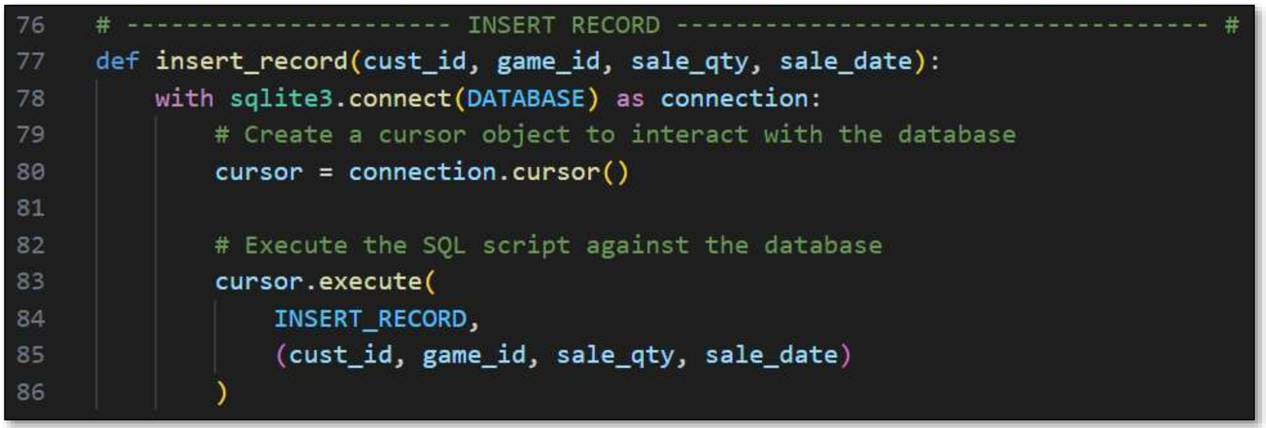
Let’s add the insert_record() function to our db_sales.py file.

The following SQL code goes at the top of the file.



```
INSERT_RECORD = """
    INSERT INTO tbl_sales (
    cust_id, game_id, sale_qty, sale_date
    ) VALUES (?, ?, ?, ?);
"""
```

Time to create the insert_record() function. This function inserts the sale record into the database.



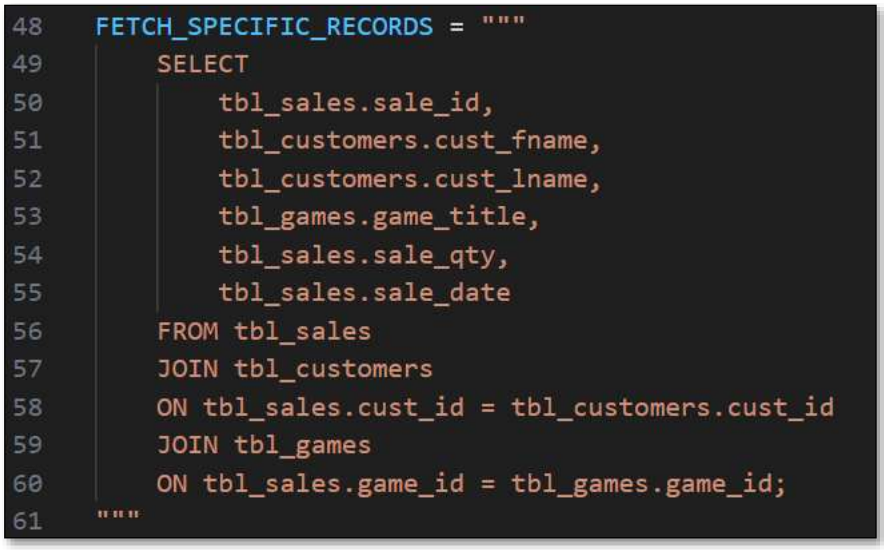
```
# --------------------- INSERT RECORD ----------------------------------- #
def insert_record(cust_id, game_id, sale_qty, sale_date):
    with sqlite3.connect(DATABASE) as connection:
        # Create a cursor object to interact with the database
        cursor = connection.cursor()

        # Execute the SQL script against the database
        cursor.execute(
            INSERT_RECORD,
            (cust_id, game_id, sale_qty, sale_date)
        )
```

Display All Sales

If we display the sales with only the id’s, we have no idea what we have sold. We need a relational query that will get the appropriate information from the games and the customers table associated with that particular sale.

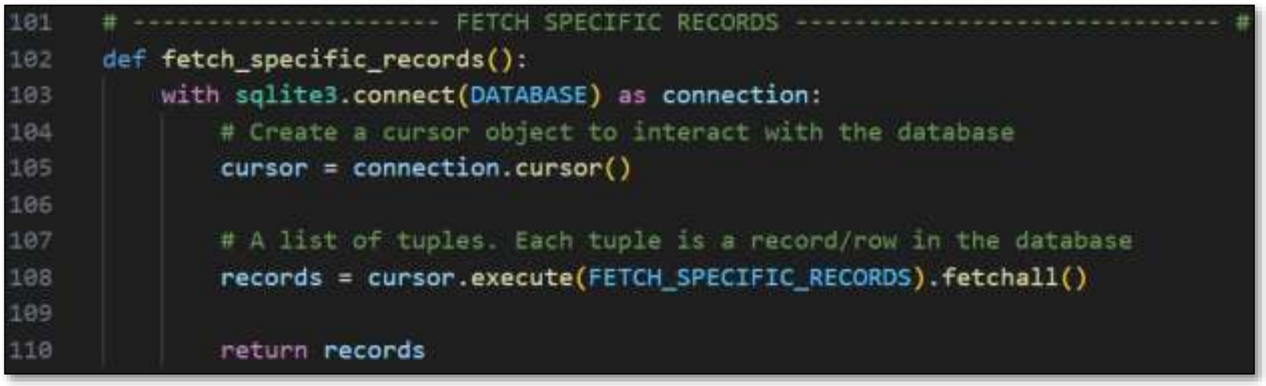
To do this, we join the customers table on the cust_id with the customers table and the games table with the game_id.

Add the following SQL to your db_sales.py file.



```python
FETCH_SPECIFIC_RECORDS = """
    SELECT
        tbl_sales.sale_id,
        tbl_customers.cust_fname,
        tbl_customers.cust_lname,
        tbl_games.game_title,
        tbl_sales.sale_qty,
        tbl_sales.sale_date
    FROM tbl_sales
    JOIN tbl_customers
    ON tbl_sales.cust_id = tbl_customers.cust_id
    JOIN tbl_games
    ON tbl_sales.game_id = tbl_games.game_id;
"""
```

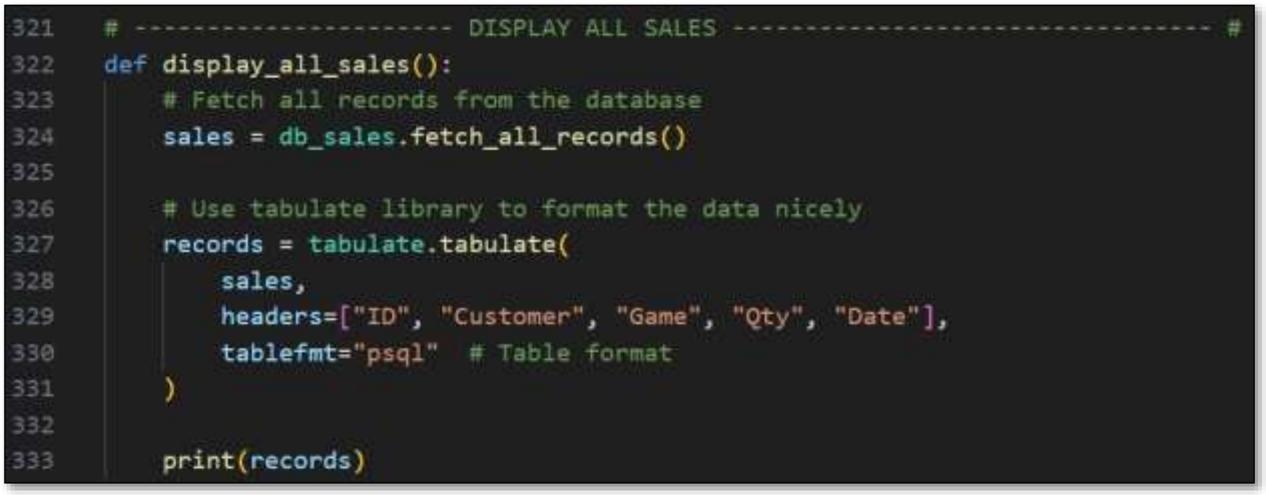
We also need the related function that will execute the SQL query against our database.



```python
# --------------------- FETCH SPECIFIC RECORDS ---------------------------- #
def fetch_specific_records():
    with sqlite3.connect(DATABASE) as connection:
        # Create a cursor object to interact with the database
        cursor = connection.cursor()

        # A list of tuples. Each tuple is a record/row in the database
        records = cursor.execute(FETCH_SPECIFIC_RECORDS).fetchall()

        return records
```

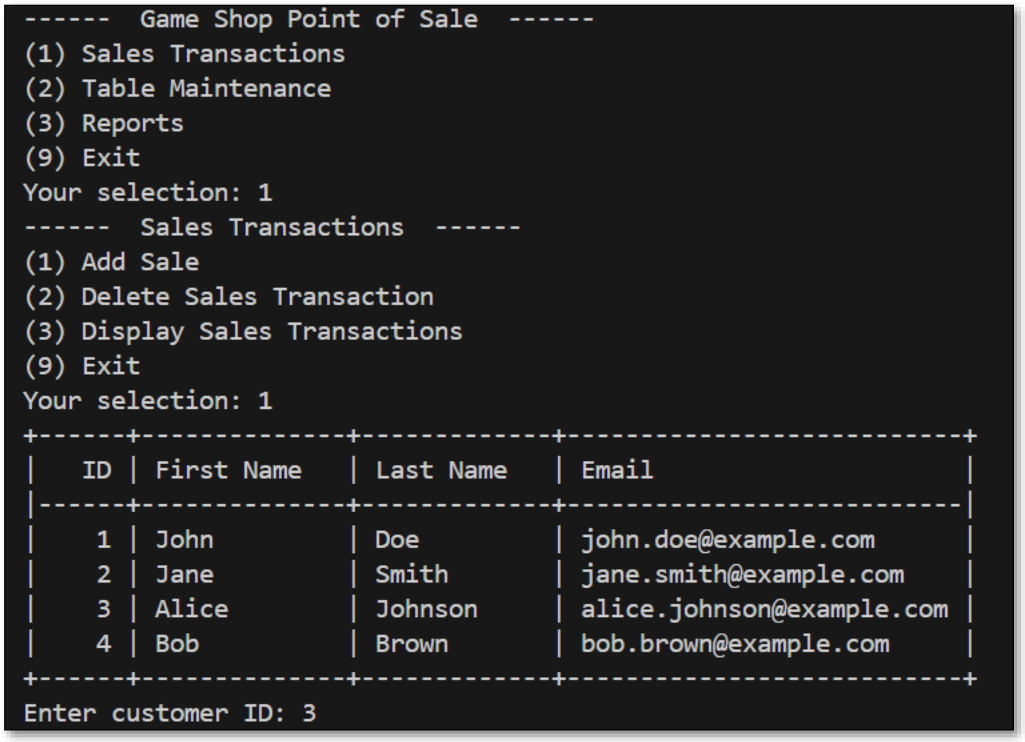
Sales Menu

Back to our game_shop_pos.py file. The following function will display all sales.

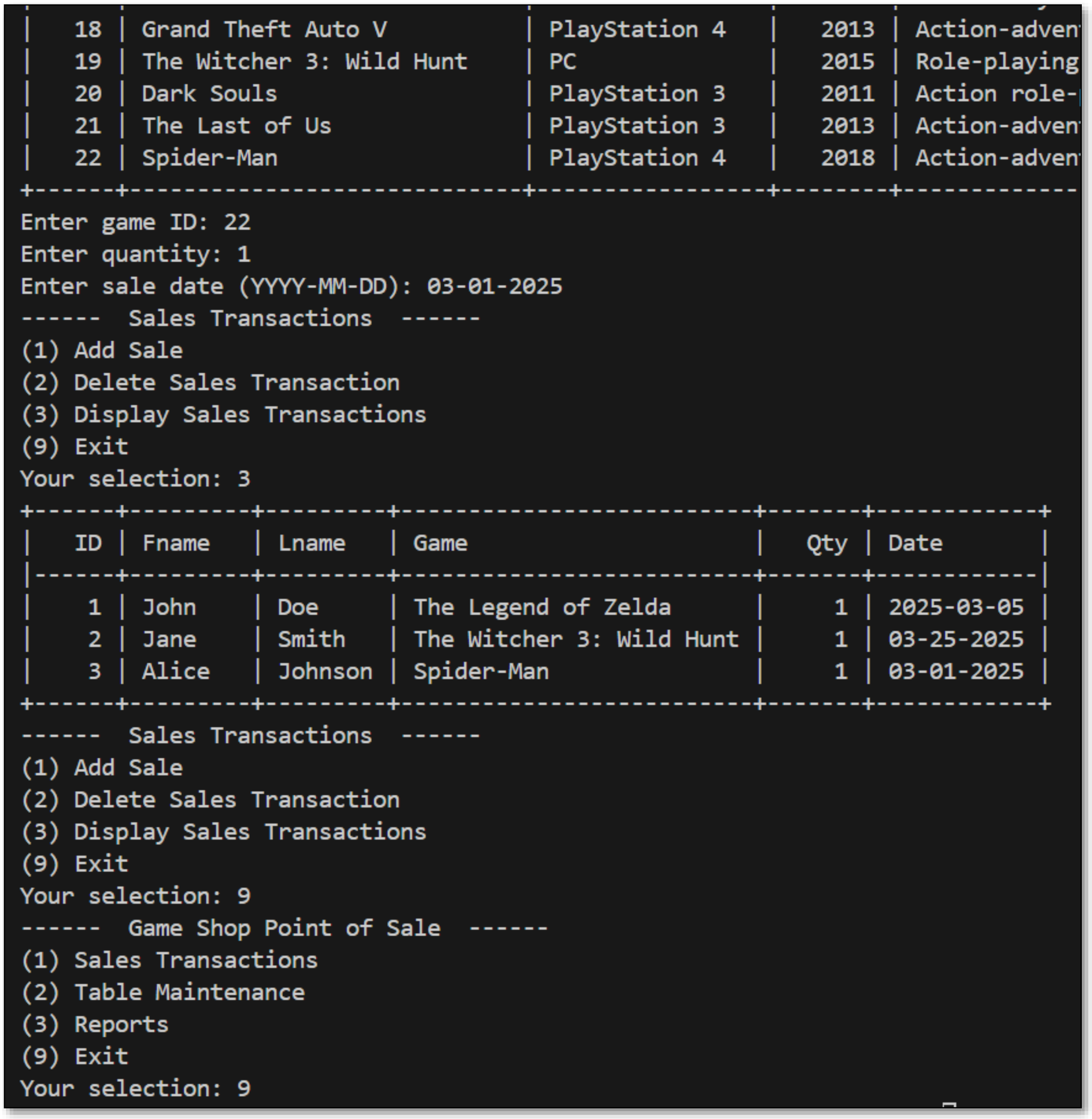


```python
# --------------------- DISPLAY ALL SALES -------------------------------- #
def display_all_sales():
    # Fetch all records from the database
    sales = db_sales.fetch_all_records()

    # Use tabulate library to format the data nicely
    records = tabulate.tabulate(
        sales,
        headers=["ID", "Customer", "Game", "Qty", "Date"],
        tablefmt="psql"  # Table format
    )

    print(records)
```

Let’s try out our program before we add the delete sale functionality.



```
------  Game Shop Point of Sale  ------
(1) Sales Transactions
(2) Table Maintenance
(3) Reports
(9) Exit
Your selection: 1
------  Sales Transactions  ------
(1) Add Sale
(2) Delete Sales Transaction
(3) Display Sales Transactions
(9) Exit
Your selection: 1
+------+-------------+------------+-------------------------+
|   ID | First Name  | Last Name  | Email                   |
|------+-------------+------------+-------------------------|
|    1 | John        | Doe        | john.doe@example.com    |
|    2 | Jane        | Smith      | jane.smith@example.com  |
|    3 | Alice       | Johnson    | alice.johnson@example.com |
|    4 | Bob         | Brown      | bob.brown@example.com   |
+------+-------------+------------+-------------------------+
Enter customer ID: 3
```


```
|   18 | Grand Theft Auto V          | PlayStation 4    |   2013 | Action-adven
|   19 | The Witcher 3: Wild Hunt    | PC               |   2015 | Role-playing
|   20 | Dark Souls                  | PlayStation 3    |   2011 | Action role-
|   21 | The Last of Us              | PlayStation 3    |   2013 | Action-adven
|   22 | Spider-Man                  | PlayStation 4    |   2018 | Action-adven
+------+-----------------------------+------------------+--------+------------
Enter game ID: 22
Enter quantity: 1
Enter sale date (YYYY-MM-DD): 03-01-2025
------  Sales Transactions  ------
(1) Add Sale
(2) Delete Sales Transaction
(3) Display Sales Transactions
(9) Exit
Your selection: 3
+------+---------+---------+--------------------------+-------+------------+
|   ID | Fname   | Lname   | Game                     |   Qty | Date       |
|------+---------+---------+--------------------------+-------+------------|
|    1 | John    | Doe     | The Legend of Zelda      |     1 | 2025-03-05 |
|    2 | Jane    | Smith   | The Witcher 3: Wild Hunt |     1 | 03-25-2025 |
|    3 | Alice   | Johnson | Spider-Man               |     1 | 03-01-2025 |
+------+---------+---------+--------------------------+-------+------------+
------  Sales Transactions  ------
(1) Add Sale
(2) Delete Sales Transaction
(3) Display Sales Transactions
(9) Exit
Your selection: 9
------  Game Shop Point of Sale  ------
(1) Sales Transactions
(2) Table Maintenance
(3) Reports
(9) Exit
Your selection: 9
```

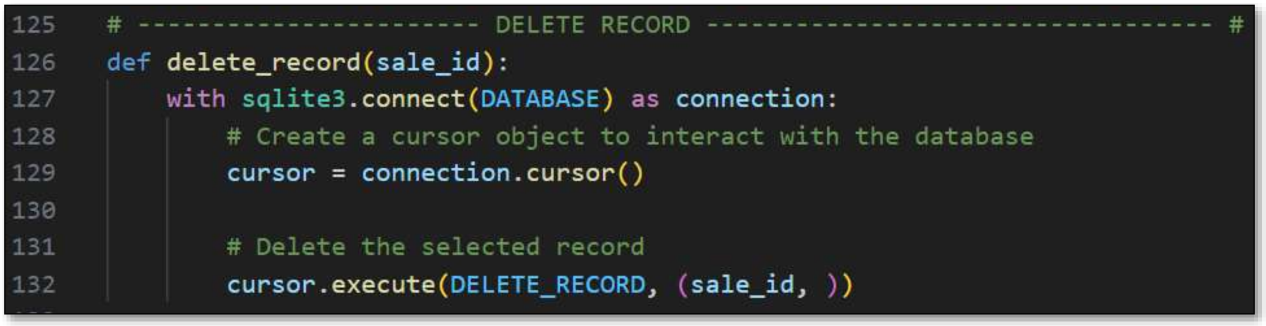
Delete Sale

Last step and our Sales menu is complete.

db_sale.py

```python
DELETE_RECORD = "DELETE FROM tbl_sales WHERE sale_id = ?"
```

delete_record function.



```
# ---------------------- DELETE RECORD -------------------------------- #
def delete_record(sale_id):
    with sqlite3.connect(DATABASE) as connection:
        # Create a cursor object to interact with the database
        cursor = connection.cursor()

        # Delete the selected record
        cursor.execute(DELETE_RECORD, (sale_id, ))
```

Test out the delete_record() function.

Assignment Submission

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