ASSIGNMENT 2

1DV503

Database Application Programming

Contact persons: Alisa Lincke (alisa.lincke@lnu.se) and TAs. For questions, use the forum on the Moodle course page or Slack.

Description
This assignment can be done in **groups** (consisting of a maximum of 2 students) or **individually**. In this assignment, you will develop a client application (console application) in Python using the Visual Studio Code IDE. The application will interact with the database (MySQL Server) using an API (Python MySQL Connector).

Prerequisites: Installed MySQL Server and MySQL Connector/Python.

Study materials: Lectures 5 and 6, and short videos

Minimum score to pass: 60

Submission:
Your submission should include solutions to all of the tasks above. Submit all of your Python files to Moodle. No .sql files should be sent, because we use our own database to test the application. Therefore, it is important that you follow the instructions in Task 1 (configuring the bookstore database) exactly.

Task 1 Use MySQL Workbench to create and configure the bookstore database (20 points)

1.1 Open MySQL Workbench/DBeaver and connect to MySQL Server
1.2 Create a named schema (**boardgame_shop**) for the boardgame shop database according to the relationship diagram shown in Figure 1.

The database consists of five tables:

Games: This table records information about the boardgames sold in the boardgame shop. Each boardgame is classified under a "genre" to enable genre searches. Each game has a *game_id* as the identification, like “BG000001”.

Users: This table records information about users of the application. Each user chooses their unique email address (unique) and password upon registration. UserId is automatically incremented and is automatically created by the database. The email address is unique in the member’s table.

Order: This table records information about orders placed by users. Orders can contain one or more games, and the order details are in a separate table. The database generates a unique order number.

Order Items: This table records information about each order, including *game_id* and the number of games in the order. It also includes a *line_total*, which represents the total price for this game.

Cart: This table contains the *game_id* and quantity of each game placed in a user’s shopping cart. Once a user has checked out, the cart is emptied and an order is created.

Of MySQLWorkbench-symboler will be useful for this task

1.3 Use the supplied ***boardgames.sql*** *(which is available in moodle)* script to populate the *book* table. The ***boardgames.sql*** file contains data for this database.

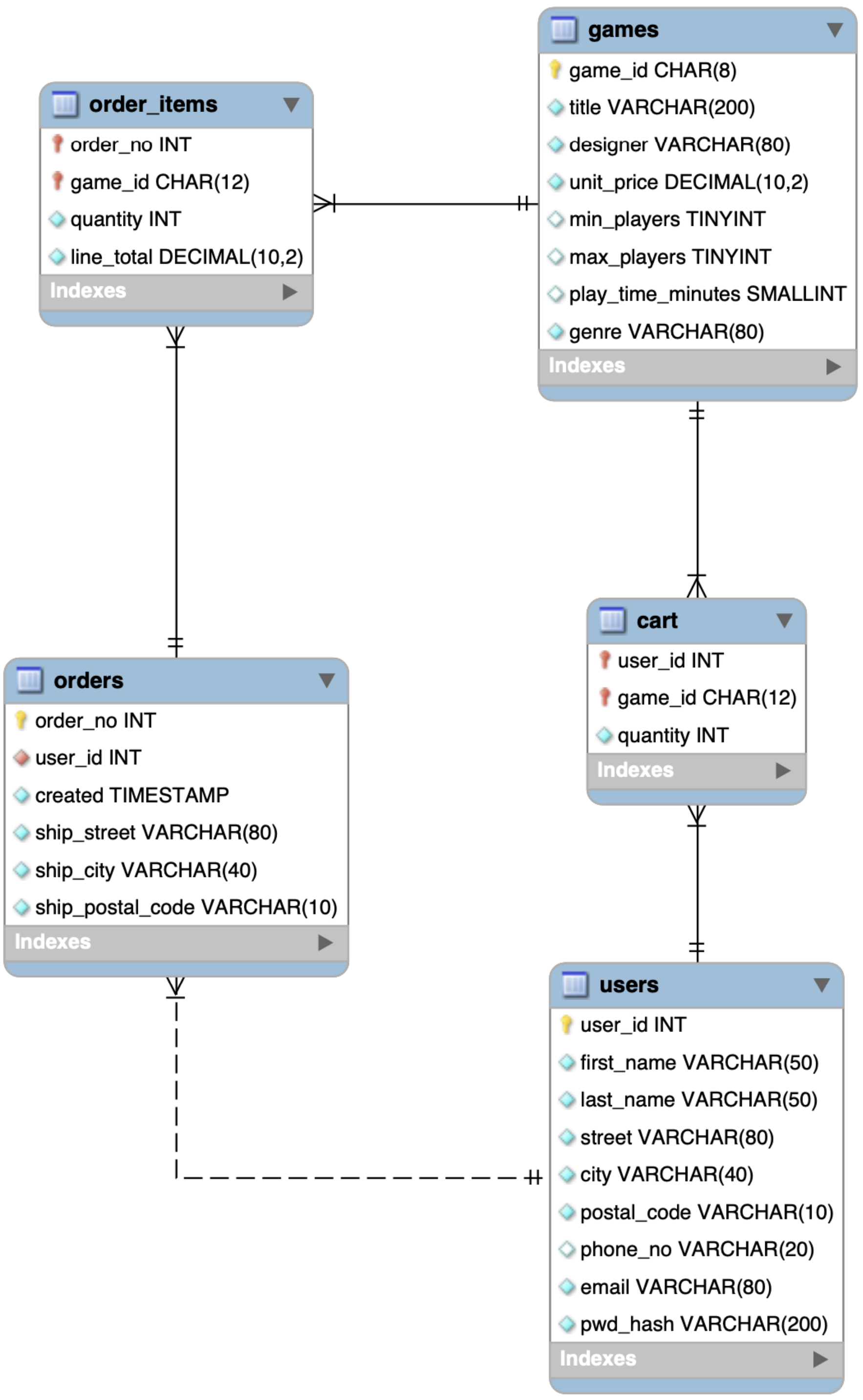


Figure 1. Relational model diagram for the boardgame shop database

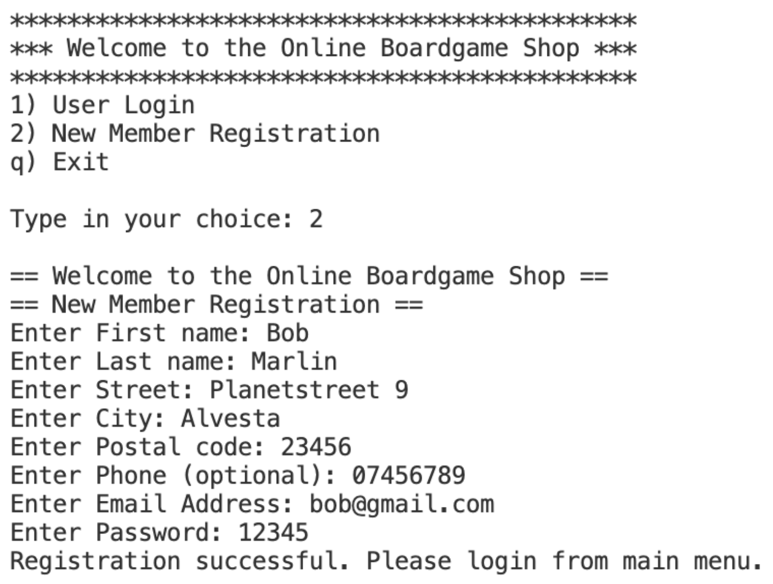
NOTE: Use the same name for all tables and attributes because we will be using our database to test your application.

Task 2 The BookStore application (80 points):

2.0 Connecting to the database

The program should prompt the user for the username and password required to connect to the database. The connection should be established only once and in a single location within the program.

2.1 Implement the member registration and member login functions, which look like the image below (20 points)



```
********************************************
*** Welcome to the Online Boardgame Shop ***
********************************************
1) User Login
2) New Member Registration
q) Exit

Type in your choice: 2

== Welcome to the Online Boardgame Shop ==
== New Member Registration ==
Enter First name: Bob
Enter Last name: Marlin
Enter Street: Planetstreet 9
Enter City: Alvesta
Enter Postal code: 23456
Enter Phone (optional): 07456789
Enter Email Address: bob@gmail.com
Enter Password: 12345
Registration successful. Please login from main menu.
```

Sensitive information stored in the database should generally be encrypted. In this task, you will focus on: encrypting only the password, using a Python library, for example `hashlib`. Ensure that all data sent to the database is valid. Input validation must be implemented directly in the Python program and not delegated to the database.

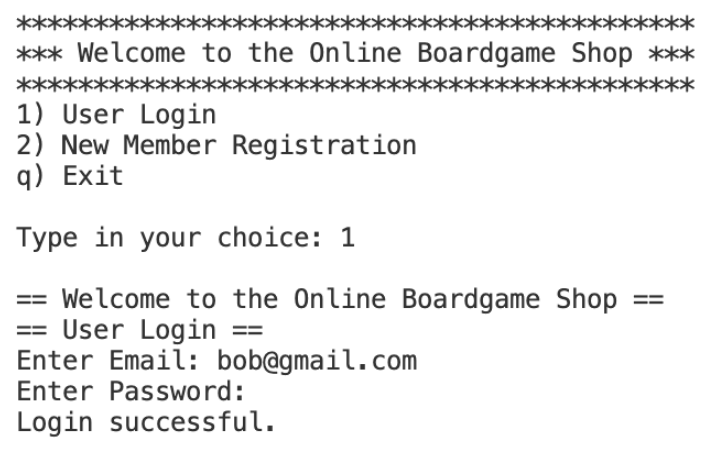
More details:

- Check that the first name, last name, email, and password fields are not empty and provide the user with an appropriate message about which field is invalid.

- Ensure that the email address is unique. If a user tries to register with an email address that already exists, display a message stating that the email address is already in use.

If registration was unsuccessful, show a message to the user and ask again or return to the main menu.

The user returns to the main menu and selects option 1.

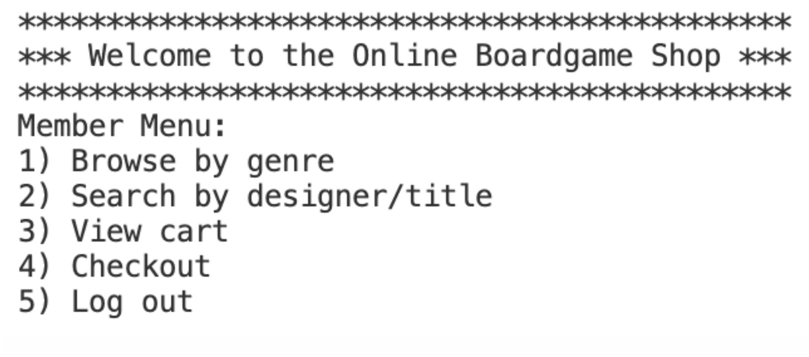


```
********************************************
*** Welcome to the Online Boardgame Shop ***
********************************************
1) User Login
2) New Member Registration
q) Exit

Type in your choice: 1

== Welcome to the Online Boardgame Shop ==
== User Login ==
Enter Email: bob@gmail.com
Enter Password:
Login successful.
```

The user should log in to the database once, and his/her user ID should be stored in a global variable or used globally in the application.

Validate the user's input for email address and password before sending it to the database. If the login is successful, display a success message. If the login fails, display an appropriate error message explaining the reason, such as "Email address does not exist," "Incorrect password," or "Email/Password required" if the user leaves any field blank.

If the login is successful, the program will display the following member menu options below.



```
********************************************
*** Welcome to the Online Boardgame Shop ***
********************************************
Member Menu:
1) Browse by genre
2) Search by designer/title
3) View cart
4) Checkout
5) Log out
```

2.2 Browse by genre (20 points)

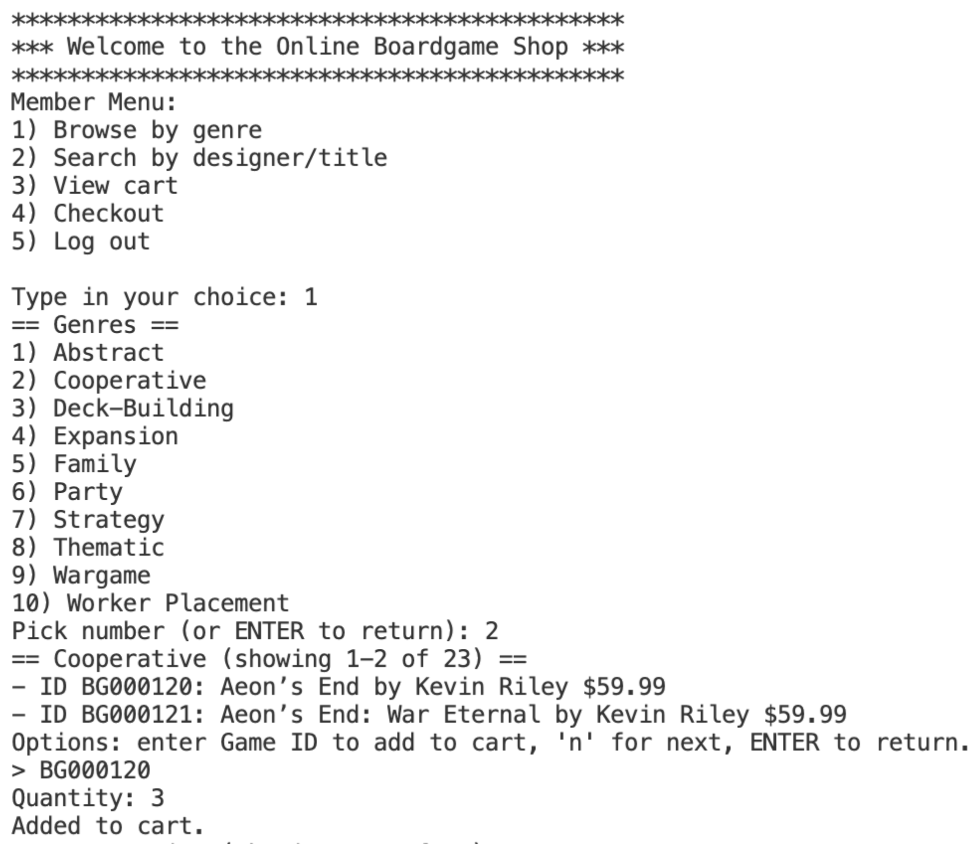
This option should first list all unique genres in alphabetical order. It then allows the user to select a genre. When selecting a genre, the program displays game details, the number of games available in the selected genre, and displays 2 games at a time on one screen. You should use OFFSET and LIMIT in the SQL query, and do not fetch all games at once, even if there are not many games in this database. Use COUNT to take the number of games for a specific genre.

The input for the genre number should be validated in the Python program. After selecting a genre, two games are displayed with their details, including the designer's name, title, game id and price. The user has three choices:

1. Enter the game id for a game
2. Press 'n Enter' to continue browsing and see the next two games.
3. Press Enter to return to the member menu.

You need to implement pagination/navigation for the books using OFFSET and LIMIT.

After the user enters the game_id, the program should ask for the number. This number should be validated (a positive number and it must be a number) by the Python program, and if invalid input is provided, display the appropriate message. The user can add many games to the shopping cart. The information about games added to the shopping cart should be saved in the "cart" table. The program should allow adding the same game that is already in the shopping cart and updating its quantity appropriately.



```
*******************************************
*** Welcome to the Online Boardgame Shop ***
*******************************************
Member Menu:
1) Browse by genre
2) Search by designer/title
3) View cart
4) Checkout
5) Log out

Type in your choice: 1
== Genres ==
1) Abstract
2) Cooperative
3) Deck-Building
4) Expansion
5) Family
6) Party
7) Strategy
8) Thematic
9) Wargame
10) Worker Placement
Pick number (or ENTER to return): 2
== Cooperative (showing 1-2 of 23) ==
- ID BG000120: Aeon’s End by Kevin Riley $59.99
- ID BG000121: Aeon’s End: War Eternal by Kevin Riley $59.99
Options: enter Game ID to add to cart, 'n' for next, ENTER to return.
> BG000120
Quantity: 3
Added to cart.
```

2.3 Search by designer/title (20 points)

This option should provide three sub-options:

1. Search for designers
2. Title search
3. Go back to the main menu

Search for designer: The user enters part of a designer's name, and the program finds all authors whose names begin with the specified string. For example, if the user enters "Ali" or "ali", the program displays all designers whose names begin with "Ali" or "ali", such as Alison, Ali, Alim, Alik, etc.

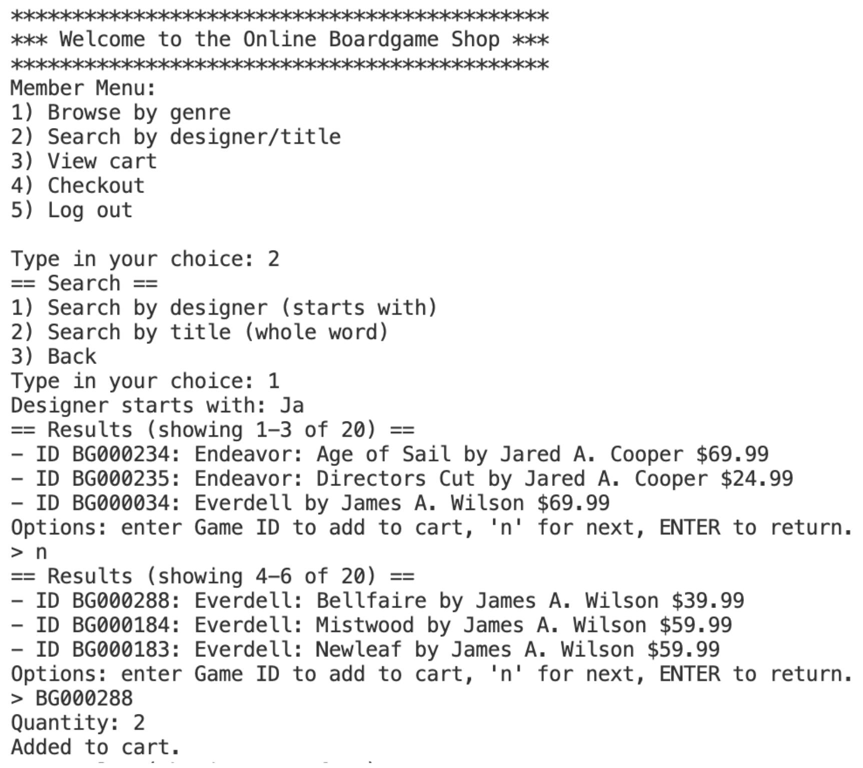
Search by title: The user enters a word, and the program finds all titles that contain exactly that word, not part of it. For example, if the word is "Sport" or “sport”, the program displays all titles that contain "Sport" (regardless of the uppercase or lowercase letter S).

The display should show 3 books at a time on the screen.

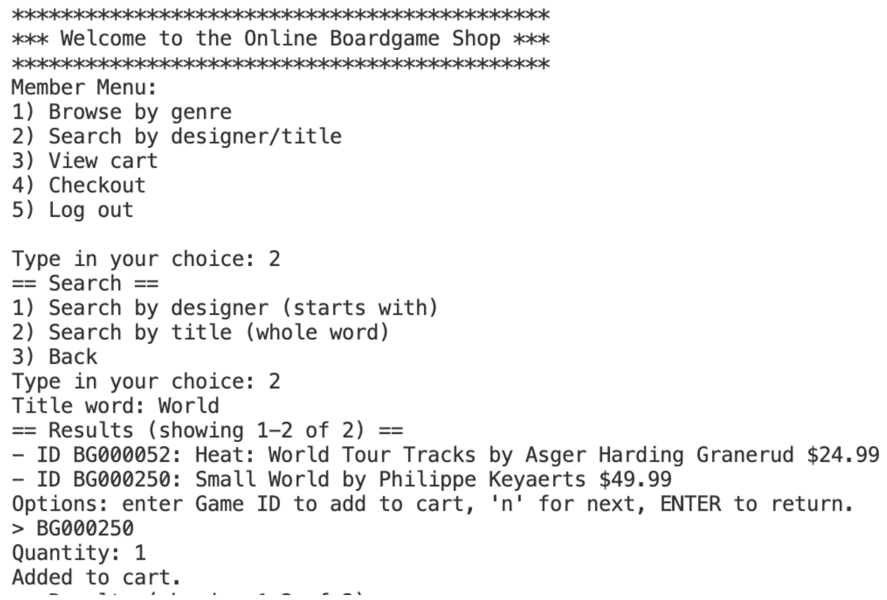
The system should also allow the user to:

enter the game_id to add the book to the cart;
to press ENTER to return to the main menu;
to press n ENTER to continue searching

~~ISBN and~~ the number of games should be validated by a Python program.



```
********************************************
*** Welcome to the Online Boardgame Shop ***
********************************************
Member Menu:
1) Browse by genre
2) Search by designer/title
3) View cart
4) Checkout
5) Log out

Type in your choice: 2
== Search ==
1) Search by designer (starts with)
2) Search by title (whole word)
3) Back
Type in your choice: 1
Designer starts with: Ja
== Results (showing 1–3 of 20) ==
- ID BG000234: Endeavor: Age of Sail by Jared A. Cooper $69.99
- ID BG000235: Endeavor: Directors Cut by Jared A. Cooper $24.99
- ID BG000034: Everdell by James A. Wilson $69.99
Options: enter Game ID to add to cart, 'n' for next, ENTER to return.
> n
== Results (showing 4–6 of 20) ==
- ID BG000288: Everdell: Bellfaire by James A. Wilson $39.99
- ID BG000184: Everdell: Mistwood by James A. Wilson $59.99
- ID BG000183: Everdell: Newleaf by James A. Wilson $59.99
Options: enter Game ID to add to cart, 'n' for next, ENTER to return.
> BG000288
Quantity: 2
Added to cart.
```



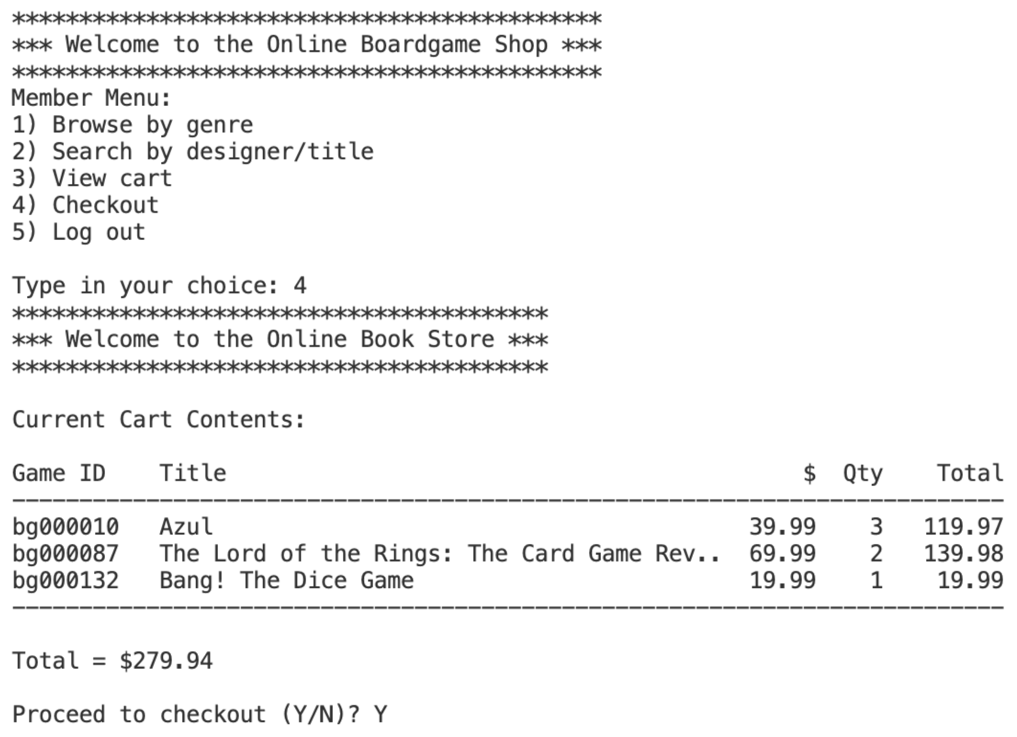
```
********************************************
*** Welcome to the Online Boardgame Shop ***
********************************************
Member Menu:
1) Browse by genre
2) Search by designer/title
3) View cart
4) Checkout
5) Log out

Type in your choice: 2
== Search ==
1) Search by designer (starts with)
2) Search by title (whole word)
3) Back
Type in your choice: 2
Title word: World
== Results (showing 1–2 of 2) ==
- ID BG000052: Heat: World Tour Tracks by Asger Harding Granerud $24.99
- ID BG000250: Small World by Philippe Keyaerts $49.99
Options: enter Game ID to add to cart, 'n' for next, ENTER to return.
> BG000250
Quantity: 1
Added to cart.
```

<ins>You should use the same implementation of pagination/navigation for the games using OFFSET and LIMIT as in task 2.2.</ins>

2.4 View Cart & To Checkout (20 points)

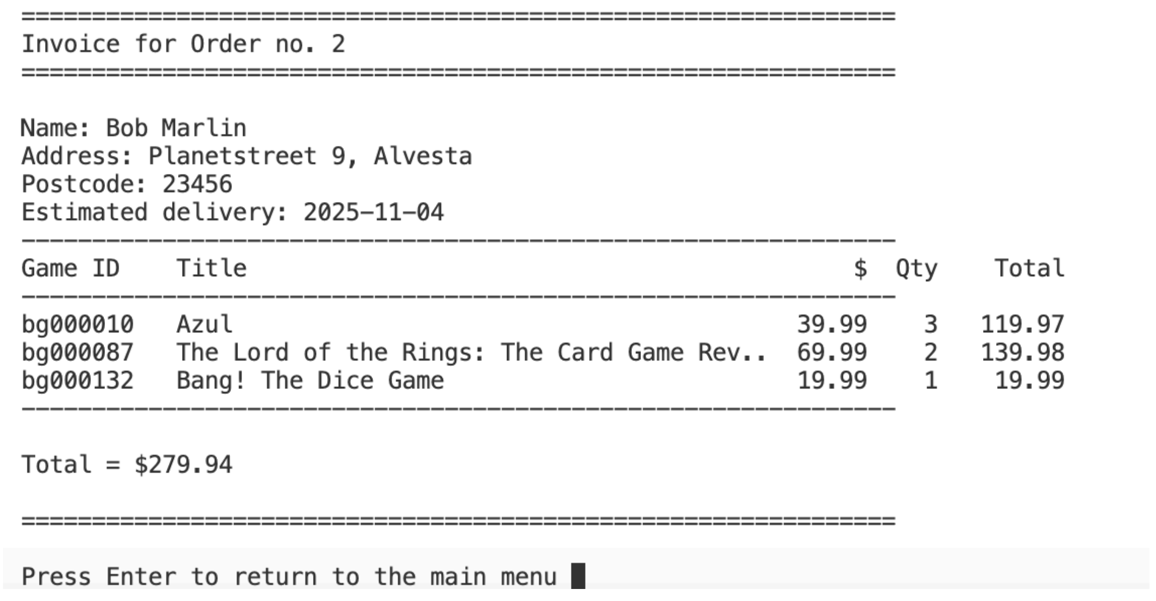
Option 3 should display a list of games in carts with game's information, including game_id, title, game price and total price.

Option 4 should display an invoice (game information (game_id, Title), game price, quantity, and total price) and a question if the user is ready to check out. The order can have many games, not just one game.



```
********************************************
*** Welcome to the Online Boardgame Shop ***
********************************************
Member Menu:
1) Browse by genre
2) Search by designer/title
3) View cart
4) Checkout
5) Log out

Type in your choice: 4
****************************************
*** Welcome to the Online Book Store ***
****************************************

Current Cart Contents:

Game ID    Title                                           $  Qty    Total
--------------------------------------------------------------------------
bg000010   Azul                                        39.99    3   119.97
bg000087   The Lord of the Rings: The Card Game Rev..  69.99    2   139.98
bg000132   Bang! The Dice Game                         19.99    1    19.99
--------------------------------------------------------------------------

Total = $279.94

Proceed to checkout (Y/N)? Y
```

When the user enters "Y" or "y" then an invoice should be printed as shown below.



```
==============================================================
Invoice for Order no. 2
==============================================================

Name: Bob Marlin
Address: Planetstreet 9, Alvesta
Postcode: 23456
Estimated delivery: 2025-11-04
--------------------------------------------------------------
Game ID    Title                                           $  Qty    Total
--------------------------------------------------------------
bg000010   Azul                                        39.99    3   119.97
bg000087   The Lord of the Rings: The Card Game Rev..  69.99    2   139.98
bg000132   Bang! The Dice Game                         19.99    1    19.99
--------------------------------------------------------------

Total = $279.94

==============================================================

Press Enter to return to the main menu █
```

Use the user's current address for delivery. The order is saved in "*orders*" the table with a recipient date ("created" date), delivery date (generated date one week in advance that is not stored in the database, only displayed on the screen), with delivery address that corresponds to the user's address at registration. And to *"order_items"* table, store all the games (game_id), their quantity (qty) and "line_total" (as quantity * game price).

If the user enters "*N*" at checkout, display the member's menu where the user can continue browsing and searching for books.

4 Log out
Back to the first menu (login and registration menu)
5 Exit
Exit the program

Note. Upon receiving incorrect input, the program should provide the user with an appropriate message to correct the input. The code should be well-structured and clean, with appropriate comments. Please **make no changes** to the specified database schema. The program interface does not have to be exactly as shown in the task; it can have a different appearance but contain the same information.

Assessment criteria

Criterion A
The database is configured according to the given instructions.
Only valid data is sent to the database
SQL queries are protected against SQL injection attacks by using parameterized queries
The database connection is established only once, and the user logs in only once during the program's execution.
Clear and appropriate messages are displayed to the user in case of invalid input.
The code is well-structured and clean, with appropriate comments
The program works as described in the task.
Criterion B
The database is configured according to the given instructions.

Only valid data is sent to the database
Not all SQL queries are protected against SQL injection attacks by using parameterized queries
The database connection is established only once, and the user logs in only once during the program's execution.
Clear and appropriate messages are displayed to the user in case of invalid input.
The code is well-structured and clean, with appropriate comments
The program works as described in the task.
Criterion C
The database is partially configured according to the given instructions.
Partially valid data is sent to the database
SQL queries are not protected against SQL injection attacks
The database connection is established only once, and the user logs in only once during the program's execution.
Not all messages are clear and appropriate in case of invalid input.
The code is partly well-structured and clean, with appropriate comments
The program partially works as described in the assignment.
Criterion D
The database is partially configured according to the given instructions.
Data validation is handled by the database instead of being performed in Python before the data is sent to the database.
SQL queries are not protected against SQL injection attacks
Not all messages are clear and appropriate in case of invalid input.
The code is partly well-structured and clean, with appropriate comments
The program partially works as described in the assignment.
The program crashes when the user input is invalid.
Criteria E
The database is partially configured according to the given instructions.

Data validation is handled by the database instead of being performed in Python before the data is sent to the database.
SQL queries are not protected against SQL injection attacks
No appropriate messages for invalid input.
The code is unstructured and without appropriate comments
The program partially works as described in the assignment.
The program crashes when the user input is invalid or for some other reason
Criterion F
Some tasks are not implemented.
The program crashes and cannot be run because the database is completely misconfigured. You cannot run and test the program.