**Exercises: Advanced Querying**

This document defines the **exercise assignments** for the ["Databases Advanced – EF Core HYPERLINK "https://softuni.bg/trainings/1741/databases-advanced-entity-framework-october-2017"" course @ Software University](https://softuni.bg/trainings/1741/databases-advanced-entity-framework-october-2017).

**BookShop System**

For the following tasks, use the [Book HYPERLINK "http://svn.softuni.org/admin/svn/csharp-databases/2019-Jan/2.%20DB-Advanced-EF-Core/07.%20DB-Advanced-Advanced-Querying/BookShop.zip"S HYPERLINK "http://svn.softuni.org/admin/svn/csharp-databases/2019-Jan/2.%20DB-Advanced-EF-Core/07.%20DB-Advanced-Advanced-Querying/BookShop.zip"hop](http://svn.softuni.org/admin/svn/csharp-databases/2019-Jan/2.%20DB-Advanced-EF-Core/07.%20DB-Advanced-Advanced-Querying/BookShop.zip) database. You can download the complete project or create it yourself in **task 0**,but you should still use the pre-defined **Seed()** method from the project to have the same **sample** data.

* **Book Shop Database**

You must create a **database** for a **book** **shop** **system**. It should look like this:



**Constraints**

Your **namespaces** should be:

* **BookShop** – for your **StartUp** class
* **BookShop.Data** – for your DbContext
* **BookShop.Models** – for your models
* **BookShop.Models**.**Enums** – for your models

Your **models** should be:

* **BookShopContext** – your DbContext
* **Author**:
* AuthorId
* FirstName (up to 50 characters, unicode, not required)
* LastName (up to 50 characters, unicode) +ICOLLECTION
* **Book**:
* BookId
* Title (up to 50 characters, unicode)
* Description (up to 1000 characters, unicode)
* ReleaseDate (not required)
* Copies (an integer)
* Price
* EditionType – enum (Normal, Promo, Gold)
* AgeRestriction – enum (Minor, Teen, Adult)
* Author
* BookCategories - ICOLLECTION
* **Category**:
* CategoryId
* Name (up to 50 characters, unicode)
* CategoryBooks -ICOLLECTION
* **BookCategory** – mapping class

For the following tasks, you will be creating methods that accept a BookShopContext as a parameter and use it to run some queries. Create those methods inside your **StartUp** class and upload your whole solution to **Judge**.

* **Age Restriction**

**NOTE**: You will need method public static string **GetBooksByAgeRestriction**(BookShopContext context, string command) and public StartUp class.

Return in a **single** **string** allbook **titles**, each on a **new line,** that have **age** **restriction**, equal to the **given** **command**. Order the titles **alphabetically**.

Read **input** from the console in your **main** **method**, and call your **method** with the **necessary** **arguments**. Print the **returned** **string** to the console. **Ignore** casing of the input.

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| miNor | A Confederacy of Dunces  A Farewell to Arms  A Handful of Dust  … |
| teEN | A Passage to India  A Scanner Darkly  A Swiftly Tilting Planet  … |

* **Golden Books**

**NOTE**: You will need method public static string **GetGoldenBooks**(BookShopContext context) and public StartUp class.

Return in a **single** string **titles of the golden edition books** that have **less than 5000 copies**,each on a **new line**. Order them by **book** **id** ascending.

Call the **GetGoldenBooks**(BookShopContext context) method in your **Main()** and print the returned string to the console.

**Example**

|  |
| --- |
| **Output** |
| Lilies of the Field  Look Homeward  The Mirror Crack'd from Side to Side  … |

* **Books by Price**

**NOTE**: You will need method public static string **GetBooksByPrice**(BookShopContext context) and public StartUp class.

Return in a single string all **titles and prices** **of books** with **price higher than 40**, each on a **new** **row** in the **format** given below. Order them by **price** descending.

**Example**

|  |
| --- |
| **Output** |
| O Pioneers! - $49.90  That Hideous Strength - $48.63  A Handful of Dust - $48.63  … |

* **Not Released In**

**NOTE**: You will need method public static string **GetBooksNotReleasedIn**(BookShopContext context, int year) and public StartUp class.

Return in a **single** string all **titles of books** that are **NOT released** on a given year. Order them by **book** **id** ascending.

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2000 | Absalom  Nectar in a Sieve  Nine Coaches Waiting  … |
| 1998 | The Needle's Eye  No Country for Old Men  No Highway  … |

* **Book Titles by Category**

**NOTE**: You will need method public static string **GetBooksByCategory**(BookShopContext context, string input) and public StartUp class.

Returnin a single string the **titles of books** by a given **list of categories**. The list of **categories** will be given in a single line separated with one or more spaces. Ignore casing. Order by **title** alphabetically.

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| horror mystery drama | A Fanatic Heart  A Farewell to Arms  A Glass of Blessings  … |

* **Released Before Date**

**NOTE**: You will need method public static string **GetBooksReleasedBefore**(BookShopContext context, string date) and public StartUp class.

Return **the title, edition type and price** of all books that are **released before a given date**. The date will be a string **in format dd-MM-yyyy**.

Return all of the rows in a **single** string, ordered by **release** **date** **descending**.

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 12-04-1992 | If I Forget Thee Jerusalem - Gold - $33.21  Oh! To be in England - Normal - $46.67  The Monkey's Raincoat - Normal - $46.93  … |
| 30-12-1989 | A Fanatic Heart - Normal - $9.41  The Curious Incident of the Dog in the Night-Time - Normal - $23.41  The Other Side of Silence - Gold - $46.26  … |

* **Author Search**

**NOTE**: You will need method public static string **GetAuthorNamesEndingIn**(BookShopContext context, string input) and public StartUp class.

Return the **full** **names** of **authors**, whose **first** **name** ends with a **given** **string**.

Return all **names** in a **single** **string**, each on a **new** **row**, ordered alphabetically.

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| e | George Powell  Jane Ortiz |
| dy | Randy Morales |

* **Book Search**

**NOTE**: You will need method public static string **GetBookTitlesContaining**(BookShopContext context, string input) and public StartUp class.

Return the **titles** of **book**, which contain a **given** **string**. Ignore casing.

Return all **titles** in a **single** **string**, each on a **new** **row**, ordered alphabetically.

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| sK | A Catskill Eagle  The Daffodil Sky  The Skull Beneath the Skin |
| WOR | Great Work of Time  Terrible Swift Sword |

* **Book Search by Author**

**NOTE**: You will need method public static string **GetBooksByAuthor**(BookShopContext context, string input) and public StartUp class.

Return **all titles of books and their authors’ names** for books, which are written by authors whose last names **start with the given string**.

Return a single string with each title on a new row. **Ignore** casing. Order by **book** **id** ascending.

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| R | The Heart Is Deceitful Above All Things (Bozhidara Rysinova)  His Dark Materials (Bozhidara Rysinova)  The Heart Is a Lonely Hunter (Bozhidara Rysinova)  … |
| po | Postern of Fate (Stanko Popov)  Precious Bane (Stanko Popov)  The Proper Study (Stanko Popov)  … |

* **Count Books**

**NOTE**: You will need method public static int **CountBooks**(BookShopContext context, int lengthCheck) and public StartUp class.

Return **the number of books,** which have a **title longer than the number** given as an input.

**Example**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 12 | 169 | There are 169 books with longer title than 12 symbols |
| 40 | 2 | There are 2 books with longer title than 40 symbols |

* **Total Book Copies**

**NOTE**: You will need method public static string **CountCopiesByAuthor**(BookShopContext context) and public StartUp class.

Return the **total number of book copies** **for each author**. Order the results **descending by total book copies**.

Return all results in a **single** **string**, each on a **new** **line**.

**Example**

|  |
| --- |
| **Output** |
| Stanko Popov - 117778  Lyubov Ivanova - 107391  Jane Ortiz – 103673  … |

* **Profit by Category**

**NOTE**: You will need method public static string **GetTotalProfitByCategory**(BookShopContext context) and public StartUp class.

Return the **total profit of all books by category**. Profit for a book can be calculated by multiplying its **number of copies** by the **price per single book**. Order the results by **descending by total profit** for category and **ascending by category name**.

**Example**

|  |
| --- |
| **Output** |
| Art $6428917.79  Fantasy $5291439.71  Adventure $5153920.77  Children's $4809746.22  … |

* **Most Recent Books**

**NOTE**: You will need method public static string **GetMostRecentBooks**(BookShopContext context) and public StartUp class.

Get the most recent books by categories. The **categories** should be ordered by **name alphabetically**. Only take the **top 3** most recent books from each category - ordered by **release date** (descending). **Select** and **print** the **category name**, and for each **book** – its **title** and **release year**.

**Example**

|  |
| --- |
| **Output** |
| --Action  Brandy ofthe Damned (2015)  Bonjour Tristesse (2013)  By Grand Central Station I Sat Down and Wept (2010)  --Adventure  The Cricket on the Hearth (2013)  Dance Dance Dance (2002)  Cover Her Face (2000)  … |

* **Increase Prices**

**NOTE**: You will need method public static void **IncreasePrices**(BookShopContext context) and public StartUp class.

**Increase the prices of all books** **released before 2010 by 5**.

* **Remove Books**

**NOTE**: You will need method public static int **RemoveBooks**(BookShopContext context) and public StartUp class.

Removeall **books**, which have less than **4200 copies**. Return an **int** - the **number of books that were deleted** from the database.

**Example**

|  |
| --- |
| **Output** |
| 34 |