Django ORM Task 2

Note - avoid N+1 query problems.

1. Using the shell plus and the django project you created before, run these queries to:

-- fetch all the objects of authors with only id, first name and last name

```
File  Edit  Selection  View  Go  Run  ...                ORMproject
models.py ×
Library > book > models.py > Book > __str__
1   from django.db import models
OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   SQL CONSOLE                                   python
from django.db.models import Avg, Case, Count, F, Max, Min, Prefetch, Q, Sum, When
from django.utils import timezone
from django.urls import reverse
from django.db.models import Exists, OuterRef, Subquery
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
>>> from book.models import Author, Book
>>> authors = list(Author.objects.values('id', 'first_name', 'last_name'))
>>> print(authors)
[{'id': 1, 'first_name': 'Haruki', 'last_name': 'Murakami'}, {'id': 2, 'first_name': 'J.K', 'last_name': 'Rowling'}, {'id': 3, 'first_name': 'Stephen', 'last_name': 'King'}, {'id': 4
, 'first_name': 'Toni', 'last_name': 'Morrison'}, {'id': 5, 'first_name': 'Gabriel García', 'last_name': 'Márquez'}]
>>> 
```

-- fetch all the objects of books as a list that contains name

```
>>> books = list(Book.objects.values_list('name', flat=True))
>>> print(books)
['Norwegian Wood', 'Kafka on the Shore', "Harry Potter and the Philosopher's Stone", 'The Shining', 'Beloved', 'One Hundred Years of Solitude', 'Song of Solomon', 'Love in the Time o
f Cholera']
>>> 
```

-- fetch all the objects of books as a tuple that contains name and count

```
>>> books = list(Book.objects.values_list('name', 'count'))
>>> print(books)
[('Norwegian Wood', 5), ('Kafka on the Shore', 15), ("Harry Potter and the Philosopher's Stone", 2), ('The Shining', 7), ('Beloved', 5), ('One Hundred Years of Solitude', 2), ('Song
of Solomon', 4), ('Love in the Time of Cholera', 5)]
>>> 
```

-- fetch an object from books table using id and update the count.

```
>>> book_id = 1
>>> book = Book.objects.get(id=book_id)
>>> new_count = 25
>>> book.count = new_count
>>> book.save()
>>> print(f"The count of {book.name} has been updated to {book.count}.")
The count of Norwegian Wood has been updated to 25.
>>> 
```

-- fetch an object from the books using name and update the author of that book

```
>>> book_name = "Norwegian Wood"
>>> try:
...     book = Book.objects.get(name=book_name)
...     new_author_id = 2
...     book.author_id = new_author_id
...     book.save()
...     print(f"The author of '{book.name}' has been updated to author with id {new_author_id}.")
... except Book.DoesNotExist:
...     print(f"The book with name '{book_name}' does not exist.")
...
The author of 'Norwegian Wood' has been updated to author with id 2.
>>> 
```

-- fetch all books from the table and show the name of the books and full name of authors as

a list of dictionaries

```
>>> books_with_authors = Book.objects.values('name', 'author__first_name', 'author__last_name')
>>>
>>> books_list = list(books_with_authors)
>>>
>>> print(books_list)
[{'name': 'Norwegian Wood', 'author__first_name': 'J.K', 'author__last_name': 'Rowling'}, {'name': 'Kafka on the Shore', 'author__first_name': 'Haruki', 'author__last_name': 'Murakam
i'}, {'name': "Harry Potter and the Philosopher's Stone", 'author__first_name': 'J.K', 'author__last_name': 'Rowling'}, {'name': 'The Shining', 'author__first_name': 'Stephen', 'auth
or__last_name': 'King'}, {'name': 'Beloved', 'author__first_name': 'Toni', 'author__last_name': 'Morrison'}, {'name': 'One Hundred Years of Solitude', 'author__first_name': 'Gabriel
García', 'author__last_name': 'Márquez'}, {'name': 'Song of Solomon', 'author__first_name': 'Toni', 'author__last_name': 'Morrison'}, {'name': 'Love in the Time of Cholera', 'author_
_first_name': 'Gabriel García', 'author__last_name': 'Márquez'}]
>>> 
```

-- fetch all the authors from the table and show the name of authors with the list of names of

the books of that author in a list of dictionaries format

```
>>> authors = Author.objects.all()
>>> authors_list = []
>>> for author in authors:
...     books = author.book_set.all()
...     book_names = [book.name for book in books]
...     authors_list.append({
...         'first_name': author.first_name,
...         'last_name': author.last_name,
...         'books': book_names
...     })
...
>>> print(authors_list)
[{'first_name': 'Haruki', 'last_name': 'Murakami', 'books': ['Kafka on the Shore']}, {'first_name': 'J.K', 'last_name': 'Rowling', 'books': ['Norwegian Wood', "Harry Potter and the P
hilosopher's Stone"]}, {'first_name': 'Stephen', 'last_name': 'King', 'books': ['The Shining']}, {'first_name': 'Toni', 'last_name': 'Morrison', 'books': ['Beloved', 'Song of Solomon
']}, {'first_name': 'Gabriel García', 'last_name': 'Márquez', 'books': ['One Hundred Years of Solitude', 'Love in the Time of Cholera']}]
>>>
```

-- fetch all the books from the table and show the name, price and count of the books in a list of dictionary format inside the queryset output

```
>>> books_with_details = Book.objects.values('name', 'price', 'count')
>>> print(list(books_with_details))
[{'name': 'Norwegian Wood', 'price': Decimal('500.00'), 'count': 25}, {'name': 'Kafka on the Shore', 'price': Decimal('200.00'), 'count': 15}, {'name': "Harry Potter and the Philosop
her's Stone", 'price': Decimal('300.00'), 'count': 2}, {'name': 'The Shining', 'price': Decimal('500.00'), 'count': 7}, {'name': 'Beloved', 'price': Decimal('700.00'), 'count': 5}, {
'name': 'One Hundred Years of Solitude', 'price': Decimal('800.00'), 'count': 2}, {'name': 'Song of Solomon', 'price': Decimal('900.00'), 'count': 4}, {'name': 'Love in the Time of C
holera', 'price': Decimal('1000.00'), 'count': 5}]
>>>
```

-- fetch an author from the table and increase the price with 100 for all the books of that Author

```
>>> author = Author.objects.get(id=author_id)
>>> books = author.book_set.all()
>>> for book in books:
...     book.price += 100
...     book.save()
...
>>> author_id = 1
>>> author = Author.objects.get(id=author_id)
>>> books = author.book_set.all()
>>> for book in books:
...     book.price += 100
...     book.save()
...     print(f"The price of '{book.name}' by {author.first_name} {author.last_name} has been increased by 100. New price: {book.price}")
...
The price of 'Kafka on the Shore' by Haruki Murakami has been increased by 100. New price: 500.00
>>>
```

-- fetch all the authors from the table and show the name of authors with the list of names of the books of that author that have an average rating greater than 3 in a list of dictionaries format

```
>>> authors = Author.objects.all()
>>> authors_list = []
>>> for author in authors:
...     books_with_high_rating = author.book_set.filter(average_rating__gt=3)
...     book_names = [book.name for book in books_with_high_rating]
...     authors_list.append({
...         'first_name': author.first_name,
...         'last_name': author.last_name,
...         'average_rating': author.average_rating,
...         'books_with_high_rating': book_names
...     })
...
>>> print(authors_list)
[{'first_name': 'Haruki', 'last_name': 'Murakami', 'average_rating': 8.5, 'books_with_high_rating': ['Kafka on the Shore']}, {'first_name': 'J.K', 'last_name': 'Rowling', 'average_ra
ting': 9.0, 'books_with_high_rating': ['Norwegian Wood', "Harry Potter and the Philosopher's Stone"]}, {'first_name': 'Stephen', 'last_name': 'King', 'average_rating': 6.0, 'books_wi
th_high_rating': ['The Shining']}, {'first_name': 'Toni', 'last_name': 'Morrison', 'average_rating': 9.5, 'books_with_high_rating': ['Beloved', 'Song of Solomon']}, {'first_name': 'G
abriel García', 'last_name': 'Márquez', 'average_rating': 6.7, 'books_with_high_rating': ['One Hundred Years of Solitude', 'Love in the Time of Cholera']}]
>>>
```

-- fetch all the authors from the table and show the name of authors with the list of names of the books of that author that have an average rating less than or equal to 3 in a list of dictionaries format

```
>>> authors = Author.objects.all()
>>>
>>> authors_list = []
>>>
>>> for author in authors:
...     low_rating_books = author.book_set.filter(average_rating__lte=3)
...     if low_rating_books.exists():
...         low_rating_book_names = [book.name for book in low_rating_books]
...         authors_list.append({
...             'first_name': author.first_name,
...             'last_name': author.last_name,
...             'books_with_low_rating': low_rating_book_names
...         })
...
>>> print(authors_list)
[]
>>>
```

-- fetch all the books from the table that has an average rating as 3

```
>>> books_with_rating_3 = Book.objects.filter(average_rating=3)
>>> print(books_with_rating_3)
<QuerySet []>
>>>
```

-- fetch all the authors from the table that have books_count less than 10

```
>>> authors_with_few_books = Author.objects.filter(books_count__lt=10)
>>> print(authors_with_few_books)
<QuerySet [<Author: J.K Rowling>, <Author: Stephen King>, <Author: Toni Morrison>, <Author: Gabriel García Márquez>]>
>>>
```

-- fetch all the authors from the table and update the books_count value with respect to the exact number of books connected to that author

```
>>> authors = Author.objects.all()
>>> for author in authors:
...     book_count = author.book_set.count()
...     author.books_count = book_count
...     author.save()
...     print(f"Updated books_count for {author}: {book_count}")
...
Updated books_count for Haruki Murakami: 1
Updated books_count for J.K Rowling: 2
Updated books_count for Stephen King: 1
Updated books_count for Toni Morrison: 2
Updated books_count for Gabriel García Márquez: 2
>>>
```

-- fetch all the books from the table and avoid count field while fetching the objects

```
>>> books_without_count = Book.objects.exclude()
>>> print(books_without_count)
<QuerySet [<Book: Norwegian Wood>, <Book: Kafka on the Shore>, <Book: Harry Potter and the Philosopher's Stone>, <Book: The Shining>, <Book: Beloved>, <Book: One Hundred Years of Sol
itude>, <Book: Song of Solomon>, <Book: Love in the Time of Cholera>]>
>>> books_without_count = Book.objects.values('id', 'name', 'price', 'average_rating', 'author_id')
>>> print(list(books_without_count))
[{'id': 1, 'name': 'Norwegian Wood', 'price': Decimal('500.00'), 'average_rating': 8.5, 'author_id': 2}, {'id': 2, 'name': 'Kafka on the Shore', 'price': Decimal('500.00'), 'average_
rating': 9.0, 'author_id': 1}, {'id': 3, 'name': "Harry Potter and the Philosopher's Stone", 'price': Decimal('300.00'), 'average_rating': 5.0, 'author_id': 2}, {'id': 5, 'name': 'Th
e Shining', 'price': Decimal('500.00'), 'average_rating': 4.5, 'author_id': 3}, {'id': 7, 'name': 'Beloved', 'price': Decimal('700.00'), 'average_rating': 8.9, 'author_id': 4}, {'id'
: 8, 'name': 'One Hundred Years of Solitude', 'price': Decimal('900.00'), 'average_rating': 7.7, 'author_id': 5}, {'id': 9, 'name': 'Song of Solomon', 'price': Decimal('900.00'), 'av
erage_rating': 9.0, 'author_id': 4}, {'id': 10, 'name': 'Love in the Time of Cholera', 'price': Decimal('1100.00'), 'average_rating': 5.6, 'author_id': 5}]
>>>
```

-- featch all the authors in a queryset showing list of values that contains first name and last name of that author

```
>>> authors_query = Author.objects.values_list('first_name', 'last_name')
>>> print(list(authors_query))
[('Haruki', 'Murakami'), ('J.K', 'Rowling'), ('Stephen', 'King'), ('Toni', 'Morrison'), ('Gabriel García', 'Márquez')]
>>>
```