Django ORM Task 3

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 from the book table and use annotate to add book_name to show name of the book. Show only book_name and price in list of dictionaries

```
>>> from django.db.models import F
>>> books = Book.objects.annotate(book_name=F('name')).values('book_name', 'price')
>>> books_list = list(books)
>>> print(books_list)
[{'price': Decimal('500.00'), 'book_name': 'Norwegian Wood'}, {'price': Decimal('500.00'), 'book_name': 'Kafka on the Shore'}, {'price': Decimal('300.00'), 'book_nam
e': "Harry Potter and the Philosopher's Stone"}, {'price': Decimal('500.00'), 'book_name': 'The Shining'}, {'price': Decimal('700.00'), 'book_name': 'Beloved'}, {'pr
ice': Decimal('900.00'), 'book_name': 'One Hundred Years of Solitude'}, {'price': Decimal('900.00'), 'book_name': 'Song of Solomon'}, {'price': Decimal('1100.00'), '
book_name': 'Love in the Time of Cholera'}]
>>>
```

-- fetch all the authors and use annotate to set full name and show the list of full name of the Authors

```
>>> from django.db.models import CharField, F, Value
>>> from django.db.models.functions import Concat
>>> authors_with_full_names = Author.objects.annotate(full_name=Concat('first_name', Value(' '), 'last_name', output_field=CharField()))
>>> full_names = [author.full_name for author in authors_with_full_names]
>>> print(full_names)
['Haruki Murakami', 'J.K Rowling', 'Stephen King', 'Toni Morrison', 'Gabriel García Márquez']
```

-- show the count of authors that has an average rating greater than or equal to 4 using Aggregate

```
>>> from django.db.models import Count
>>> authors_count = Author.objects.filter(average_rating__gte=4).aggregate(author_count=Count('id'))
>>> count_of_authors = authors_count['author_count']
>>> print(count_of_authors)
5
```

-- fetch all the books and add a field to store discount using annotate, the book with an average rating less than or equal to 3 should have a discount of 20% and all others. should have 0% discount. Add another variable to store the actual price of that book after discount by using annotate. Show the books name, discount, actual price in a list of dictionaries

```
>>> from django.db.models import Case, When, F, FloatField, Value, DecimalField
>>> from django.db.models.functions import Coalesce
>>>
>>> books_with_discount = Book.objects.annotate(
...     discount=Case(
...         When(average_rating__lte=3, then=Value(0.2)),
...         default=Value(0),
...         output_field=FloatField()
...     ),
...     actual_price=Coalesce(F('price') - F('price') * F('discount'), Value(0), output_field=DecimalField())
... ).values('name', 'discount', 'actual_price')
>>>
>>> books_list = list(books_with_discount)
>>> print(books_list)
[{'name': 'Norwegian Wood', 'discount': 0.0, 'actual_price': Decimal('500')}, {'name': 'Kafka on the Shore', 'discount': 0.0, 'actual_price': Decimal('500')}, {'name
': "Harry Potter and the Philosopher's Stone", 'discount': 0.0, 'actual_price': Decimal('300')}, {'name': 'The Shining', 'discount': 0.0, 'actual_price': Decimal('50
0')}, {'name': 'Beloved', 'discount': 0.0, 'actual_price': Decimal('700')}, {'name': 'One Hundred Years of Solitude', 'discount': 0.0, 'actual_price': Decimal('900')
}, {'name': 'Song of Solomon', 'discount': 0.0, 'actual_price': Decimal('900')}, {'name': 'Love in the Time of Cholera', 'discount': 0.0, 'actual_price': Decimal('11
00')}]
```

-- fetch all the authors and their books count that has average rating greater than or equal to

3 using subquery and annotate. Show the first name of author and books count as a list of dictionaries

```
>>> from django.db.models import Count
>>> filtered_authors = Author.objects.filter(book__average_rating__gte=3).distinct()
>>> authors_with_book_count = filtered_authors.annotate(
...     book_count=Count('book')
... ).values('first_name', 'book_count')
>>> authors_data = list(authors_with_book_count)
>>> for author in authors_data:
...     print(author)
...
{'first_name': 'Haruki', 'book_count': 1}
{'first_name': 'J.K', 'book_count': 2}
{'first_name': 'Stephen', 'book_count': 1}
{'first_name': 'Toni', 'book_count': 2}
{'first_name': 'Gabriel García', 'book_count': 2}
>>>
```

-- fetch all the books which are from one of the authors, filter using first name

```
>>> from book.models import Book
>>> books = Book.objects.filter(author__first_name='Toni')
>>> books_list = list(books.values())
>>> print(books_list)
[{'id': 7, 'name': 'Beloved', 'price': Decimal('700.00'), 'average_rating': 8.9, 'count': 5, 'author_id': 4}, {'id': 9, 'name': 'Song of Solomon', 'price': Decimal('
900.00'), 'average_rating': 9.0, 'count': 4, 'author_id': 4}]
>>>
```

-- fetch all the authors along with their books

```
>>> from book.models import Book
>>> books = Book.objects.filter(author__first_name='Toni')
>>> books_list = list(books.values())
>>> print(books_list)
[{'id': 7, 'name': 'Beloved', 'price': Decimal('700.00'), 'average_rating': 8.9, 'count': 5, 'author_id': 4}, {'id': 9, 'name': 'Song of Solomon', 'price': Decimal('
900.00'), 'average_rating': 9.0, 'count': 4, 'author_id': 4}]
>>> authors = Author.objects.all().prefetch_related('book_set')
>>> for author in authors:
...     print(author.first_name, author.last_name)
...     for book in author.book_set.all():
...         print(book.name)
...
Haruki Murakami
Kafka on the Shore
J.K Rowling
Norwegian Wood
Harry Potter and the Philosopher's Stone
Stephen King
The Shining
Toni Morrison
Beloved
Song of Solomon
Gabriel García Márquez
One Hundred Years of Solitude
Love in the Time of Cholera
>>>
```

-- fetch all the authors and use subquery to fetch count of the books, show the full name and count of the books in a list of dictionary format

```
>>> from django.db.models import Subquery, Count, OuterRef, IntegerField
>>> from django.db.models.functions import Concat
>>>
>>> book_count_subquery = Book.objects.filter(author=OuterRef('pk')).values('author').annotate(
...     book_count=Count('id')
... ).values('book_count')
>>>
>>> authors_with_book_count = Author.objects.annotate(
...     full_name=Concat('first_name', Value(' '), 'last_name'),
...     book_count=Subquery(book_count_subquery, output_field=IntegerField())
... ).values('full_name', 'book_count')
>>>
>>> authors_with_book_count_list = list(authors_with_book_count)
>>>
>>> print(authors_with_book_count_list)
[{'full_name': 'Haruki Murakami', 'book_count': 1}, {'full_name': 'J.K Rowling', 'book_count': 2}, {'full_name': 'Stephen King', 'book_count': 1}, {'full_name': 'Ton
i Morrison', 'book_count': 2}, {'full_name': 'Gabriel García Márquez', 'book_count': 2}]
>>>
```

-- fetch an object from books table and update its count and save it

```
>>> book = Book.objects.get(id=1)
>>> book.count = 23
>>> book.save()
>>> print("Book count updated successfully:", book.count)
Book count updated successfully: 23
>>>
```

-- fetch an object from authors table with select for update and atomic transaction to update the average rating and save it.

```
>>> from django.db import transaction
>>> from book.models import Author
>>>
>>> with transaction.atomic():
...     author = Author.objects.select_for_update().get(id=3)
...     author.average_rating = 6
...     author.save()
...
>>> print("Author's average rating updated successfully:", author.average_rating)
Author's average rating updated successfully: 6
```

-- fetch all the objects from book table and show the name, average rating, full name of author of the books in a list of dictionary format

```
>>> books = Book.objects.all()
>>>
>>> books_info = [
...     {
...         'name': book.name,
...         'average_rating': book.average_rating,
...         'author_full_name': f"{book.author.first_name} {book.author.last_name}"
...     }
...     for book in books
... ]
>>>
>>> print(books_info)
[{'name': 'Norwegian Wood', 'average_rating': 8.5, 'author_full_name': 'J.K Rowling'}, {'name': 'Kafka on the Shore', 'average_rating': 9.0, 'author_full_name': 'Haruki Murakami'}, {'name': "Harry Potter and the Philosopher's Stone", 'average_rating': 5.0, 'author_full_name': 'J.K Rowling'}, {'name': 'The Shining', 'average_rating': 4.5, 'author_full_name': 'Stephen King'}, {'name': 'Beloved', 'average_rating': 8.9, 'author_full_name': 'Toni Morrison'}, {'name': 'One Hundred Years of Solitude', 'average_rating': 7.7, 'author_full_name': 'Gabriel García Márquez'}, {'name': 'Song of Solomon', 'average_rating': 9.0, 'author_full_name': 'Toni Morrison'}, {'name': 'Love in the Time of Cholera', 'average_rating': 5.6, 'author_full_name': 'Gabriel García Márquez'}]
>>>
```