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 book.models import Book

books = Book.objects.annotate(book_name=F('name')).values('book_name', 'price')

books_list = list(books)

print(books_list)

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 F, Value

from django.db.models.functions import Concat

from book.models import Author

authors = Author.objects.annotate(full_name=Concat(F('first_name'), Value(' '), F('last_name')))

full_names = authors.values_list('full_name', flat=True)

full_names_list = list(full_names)

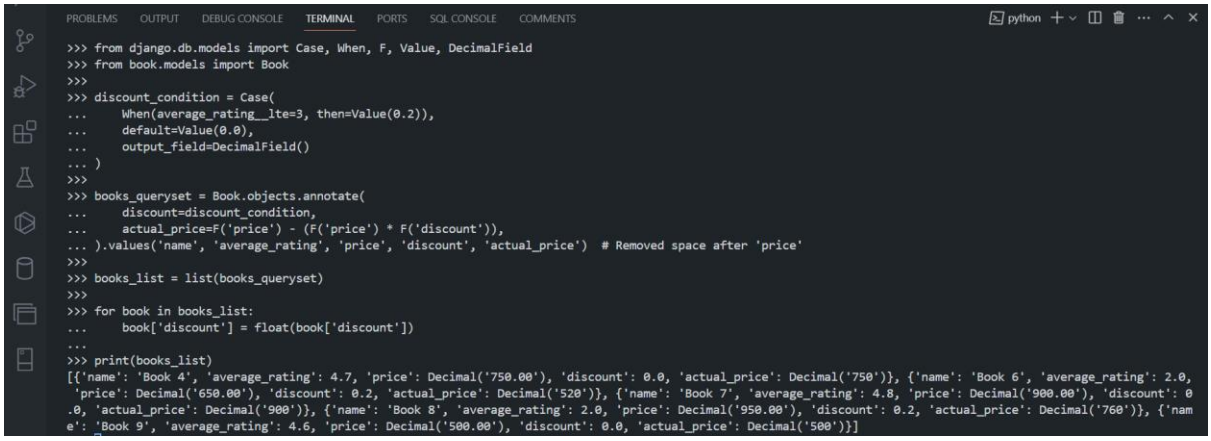
print(full_names_list)

Show the count of authors that has an average rating greater than or equal to 4 using aggregate

```
from django.db.models import Count, Avg

from book.models import Author

count_of_authors=Author.objects.filter(book__average_rating__gte=4).aggregate(num_
authors=Count('id'))

print(count_of_authors['num_authors'])
```

Fetch all the books and add a field to store discount using annotate, the book with an averate 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, Value, DecimalField

from book.models import Book

discount_condition = Case(

  When(average_rating__lte=3, then=Value(0.2)),

  default=Value(0.0),

  output_field=DecimalField()

)

books_queryset = Book.objects.annotate(

  discount=discount_condition,

  actual_price=F('price') - (F('price') * F('discount')),

).values('name', 'average_rating', 'price', 'discount', 'actual_price')  # Removed space
after 'price'

books_list = list(books_queryset)


for book in books_list:

  book['discount'] = float(book['discount'])

print(books_list)
```



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
from book.models import Book,Author
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)
```

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

```
from book.models import Book, Author
author_first_name = "Akhila"
books_from_author = Book.objects.filter(author__first_name=author_first_name)
books_list = list(books_from_author)
```

```
for book in books_list:
    print(book.name)
```

Fetch all the authors along with their books

```
from book.models import Author
authors_with_books = Author.objects.prefetch_related('book_set')
for author in authors_with_books:
    print(f"Author: {author.first_name} {author.last_name}")
    print("Books:")
    for book in author.book_set.all():
        print(f"- {book.name}")
    print(" ")
```

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, OuterRef, Count

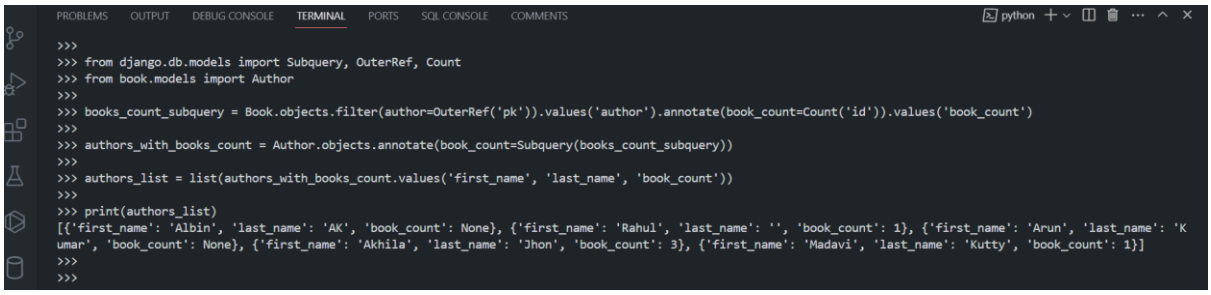
from book.models import Author

books_count_subquery = Book.objects.filter(author=OuterRef('pk')).values('author').annotate(book_count=Count('id')).values('book_count')

authors_with_books_count = Author.objects.annotate(book_count=Subquery(books_count_subquery))

authors_list = list(authors_with_books_count.values('first_name', 'last_name', 'book_count'))

print(authors_list)



Fetch an object from books table and update its count and save it

from book.models import Book

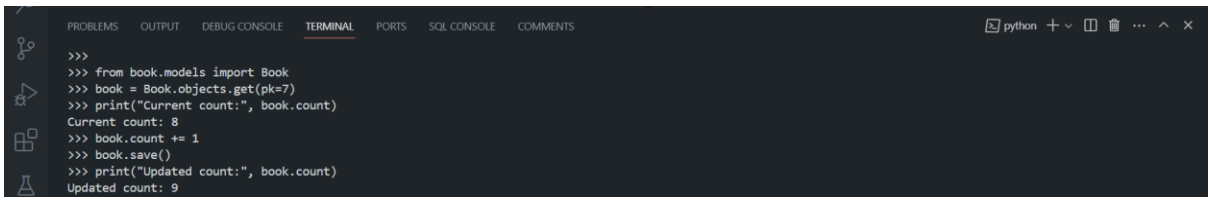
book = Book.objects.get(pk=7)

print("Current count:", book.count)

book.count += 1

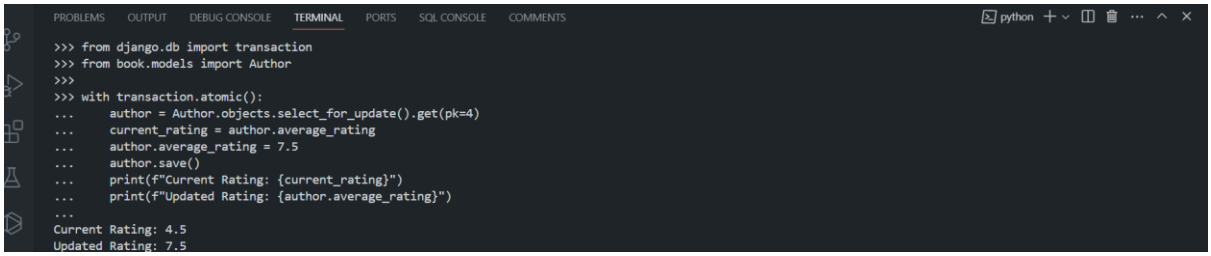
book.save()

print("Updated count:", book.count)



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(pk=4)
    current_rating = author.average_rating
    author.average_rating = 7.5
    author.save()
    print(f"Current Rating: {current_rating}")
    print(f"Updated Rating: {author.average_rating}")
```



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

```
from book.models import Book
books_data = Book.objects.values('name', 'average_rating', 'author__first_name', 'author__last_name')
books_list = list(books_data)
print(books_list)
```