Django ORM Task 1 (share the screenshots of the task)

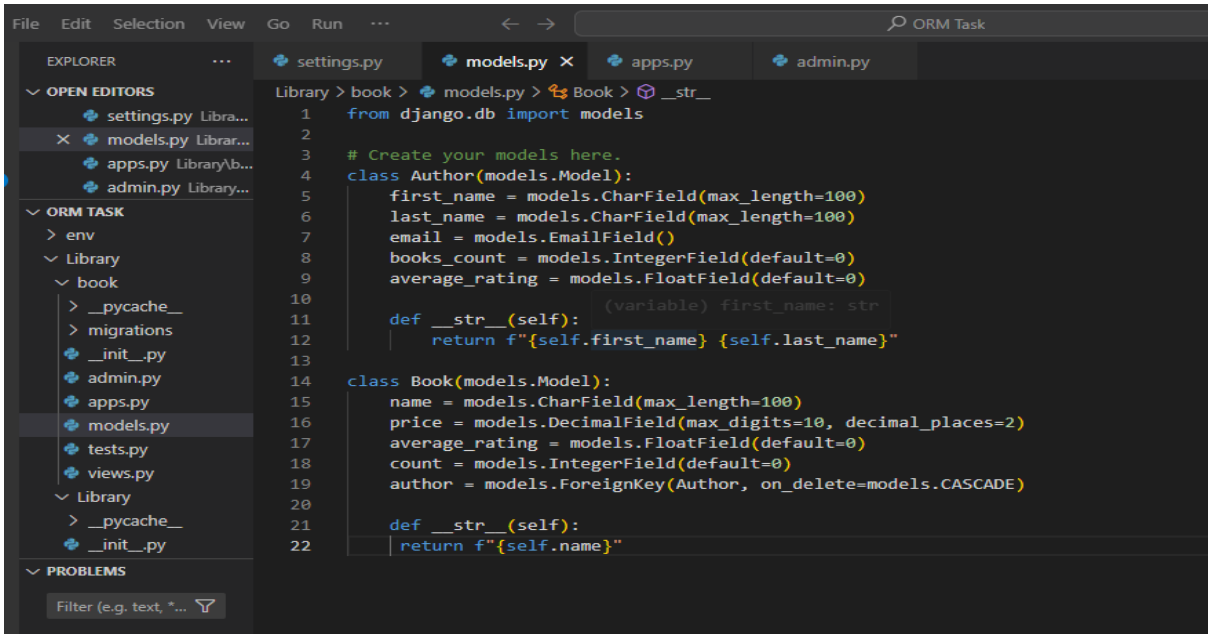
1. Create a django project named library

2. Inside the project create and app named book

3. create 2 models in the app for storing data of books and authors

-- authors[first_name, last_name, email, books_count, average_rating]

-- books[name, price, average_rating, count, author(foreign key)]



4. install django-extensions package and use shell_plus to run queries

-- create 5 authors

-- create 10 books

```
>>>
>>> from .models import Author
Traceback (most recent call last):
  File "<console>", line 1, in <module>
KeyError: "'__name__' not in globals"
>>> from book.models import Author
>>> Author.objects.bulk_create([
...     Author(first_name='Author1', last_name='Lastname1', email='author1@example.com'),
...     Author(first_name='Author2', last_name='Lastname2', email='author2@example.com'),
...     Author(first_name='Author3', last_name='Lastname3', email='author3@example.com'),
...     Author(first_name='Author4', last_name='Lastname4', email='author4@example.com'),
...     Author(first_name='Author5', last_name='Lastname5', email='author5@example.com'),
... ])
[<Author: Author1 Lastname1>, <Author: Author2 Lastname2>, <Author: Author3 Lastname3>, <Author: Author4 Lastname4>, <Author: Author5 Lastnam
e5>]
>>> from book.models import Book
>>> Book.objects.bulk_create([
...     Book(name='Book1', price=100, author_id=1),
...     Book(name='Book2', price=200, author_id=2),
...     Book(name='Book3', price=300, author_id=3),
...     Book(name='Book4', price=400, author_id=4),
...     Book(name='Book5', price=500, author_id=5),
...     Book(name='Book6', price=600, author_id=1),
...     Book(name='Book7', price=700, author_id=2),
...     Book(name='Book8', price=800, author_id=3),
...     Book(name='Book9', price=900, author_id=4),
...     Book(name='Book10', price=1000, author_id=5),
... ])
[<Book: Book object (1)>, <Book: Book object (2)>, <Book: Book object (3)>, <Book: Book object (4)>, <Book: Book object (5)>, <Book: Book obj
ect (6)>, <Book: Book object (7)>, <Book: Book object (8)>, <Book: Book object (9)>, <Book: Book object (10)>]
>>>
```

-- show all the objects in books and authors model

```
... ])
[<Book: Book object (1)>, <Book: Book object (2)>, <Book: Book object (3)>, <Book: Book object (4)>, <Book: Book object (5)>, <Book: Book obj
ect (6)>, <Book: Book object (7)>, <Book: Book object (8)>, <Book: Book object (9)>, <Book: Book object (10)>]
>>> Author.objects.all()
<QuerySet [<Author: Haruki Murakami>, <Author: J.K Rowling>, <Author: Stephen King>, <Author: Toni Morrison>, <Author: Gabriel García Márquez
>]>
>>> Book.objects.all()
<QuerySet [<Book: Book object (1)>, <Book: Book object (2)>, <Book: Book object (3)>, <Book: Book object (4)>, <Book: Book object (5)>, <Book
: Book object (6)>, <Book: Book object (7)>, <Book: Book object (8)>, <Book: Book object (9)>, <Book: Book object (10)>]>
>>>
```

-- filter and show some authors using first_name field

-- filter and show some authors that does not have a last_name

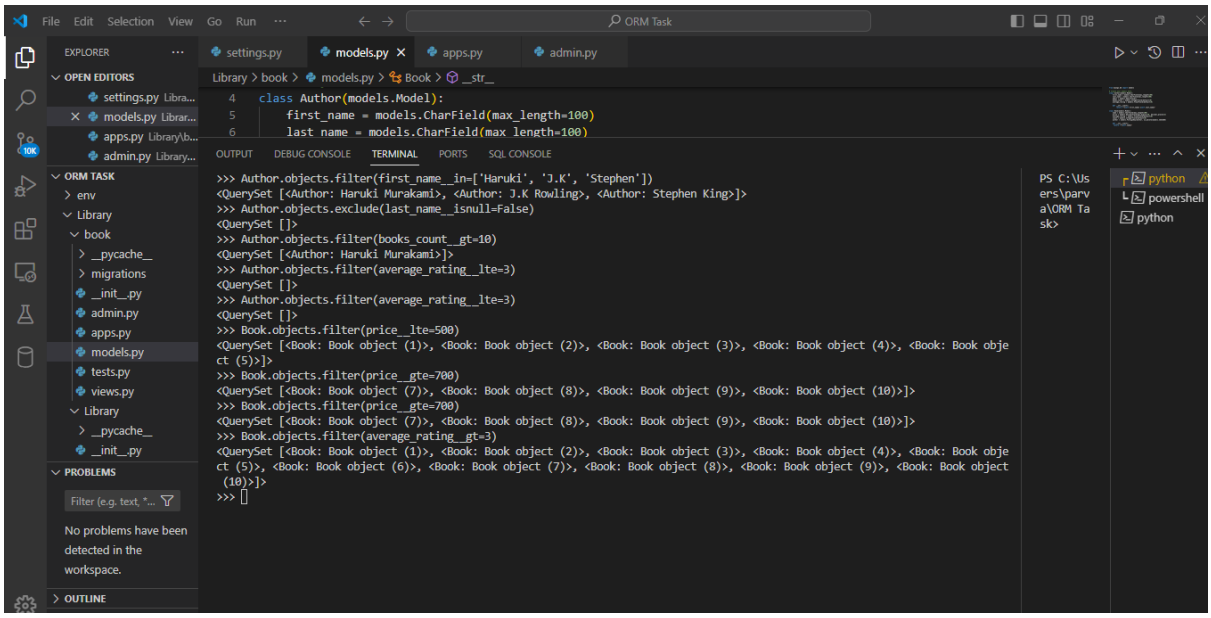
-- filter and show some authors that have books count greater than 10

-- filter and show some authors that have an average rating less than or equal to 3

-- filter and show some books that have price less than or equal to 500

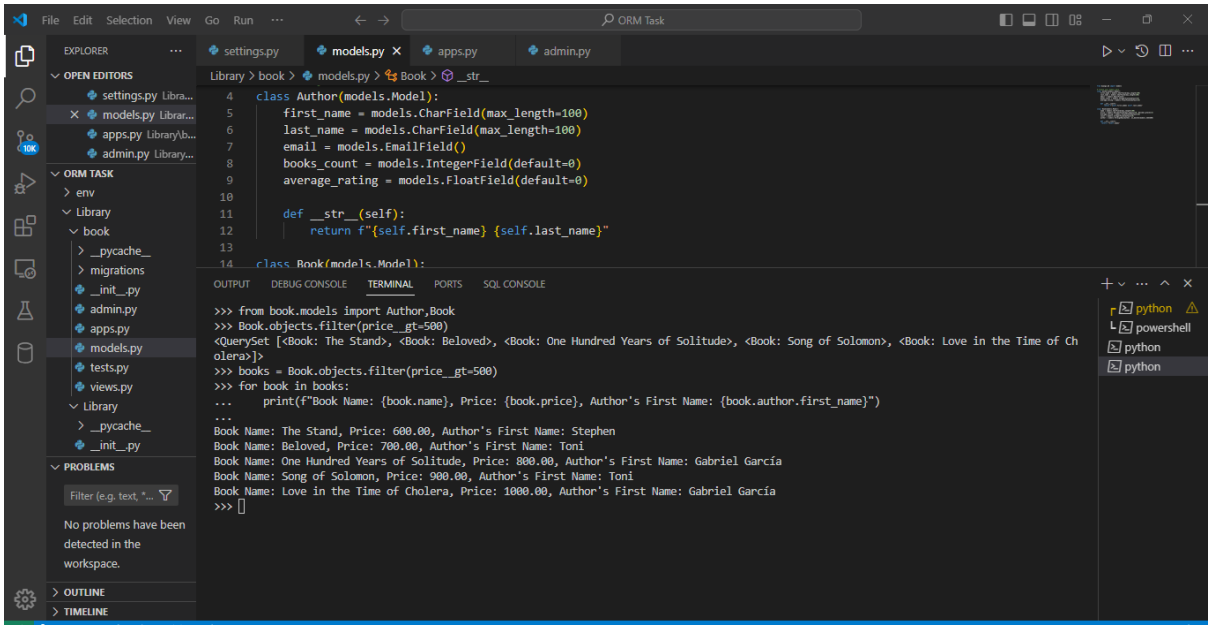
-- filter and show some books that have proce grater than or equal to 700

-- filter and show some books that has an average rating greater than 3



-- filter and show some books that has count greater than 100

-- filter some books that has price greater than 500 and show the names of all one by one along with its price and author's first name



-- fetch the first book and show its name

-- fetch the last book and show its authors full name

-- fetch a book using its id, show its name and author's full name

-- show the total count of books and authors

-- fetch a book by its id and update the author and save

```
Book Name: Love in the Time of Cholera, Price: 1000.00, Author's First Name: Gabriel García
>>> Book.objects.first()
<Book: Norwegian Wood>
>>> print(first_book.name)
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'first_book' is not defined
>>> first_book = Book.objects.first()
>>> print(first_book.name)
Norwegian Wood
>>> last_book = Book.objects.last()
>>> print(f"{last_book.name} - {last_book.author.first_name}-{last_book.author.last_name}")
Love in the Time of Cholera - Gabriel García-Márquez
>>> book = Book.objects.get(id=1)
>>> print(f"{book.name} - - {book.author.first_name}-{book.author.last_name}")
Norwegian Wood - - Haruki-Murakami
>>> total_books_count = Book.objects.count()
>>> total_authors_count = Author.objects.count()
>>> print("Total number of books:", total_books_count)
Total number of books: 10
>>> print("Total number of authors:", total_authors_count)
Total number of authors: 5
>>> book_id = 1
>>> book = Book.objects.get(pk=book_id)
>>> author_id = 2
>>> new_author = Author.objects.get(pk=author_id)
>>> book.author = new_author
>>> book.save()
>>> print("Book name after updating author:", book.name)
Book name after updating author: Norwegian Wood
>>>
```

-- show the total count of authors

-- show the count of authors that has an average rating greater than or equal to 3

-- check whether an author exists in the table using email field

-- check whether a book exists in the table with its name

```
Book Name: Beloved, Price: 700.00, Author's First Name: Toni
Book Name: One Hundred Years of Solitude, Price: 800.00, Author's First Name: Gabriel García
Book Name: Song of Solomon, Price: 900.00, Author's First Name: Toni
Book Name: Love in the Time of Cholera, Price: 1000.00, Author's First Name: Gabriel García
>>> Book.objects.first()
<Book: Norwegian Wood>
>>> print(first_book.name)
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'first_book' is not defined
>>> first_book = Book.objects.first()
>>> print(first_book.name)
Norwegian Wood
>>> last_book = Book.objects.last()
>>> print(f"{last_book.name} - {last_book.author.first_name}-{last_book.author.last_name}")
Love in the Time of Cholera - Gabriel García-Márquez
>>> book = Book.objects.get(id=1)
>>> print(f"{book.name} - - {book.author.first_name}-{book.author.last_name}")
Norwegian Wood - - Haruki-Murakami
>>> total_books_count = Book.objects.count()
>>> total_authors_count = Author.objects.count()
>>> print("Total number of books:", total_books_count)
Total number of books: 10
>>> print("Total number of authors:", total_authors_count)
Total number of authors: 5
>>> book_id = 1
>>> book = Book.objects.get(pk=book_id)
>>> author_id = 2
>>> new_author = Author.objects.get(pk=author_id)
>>> book.author = new_author
>>> book.save()
>>> print("Book name after updating author:", book.name)
Book name after updating author: Norwegian Wood
>>> total_authors_count = Author.objects.count()
>>> print("Total number of authors:", total_authors_count)
Total number of authors: 5
```

-- show all the books in the order of its name

-- delete 2 books from the table after fetching it with name

```
>>> high_rated_authors_count = Author.objects.filter(average_rating__gte=3).count()
>>> print(f"Authors with average rating >= 3: {high_rated_authors_count}")
Authors with average rating >= 3: 5
>>> email_to_check = "Gabriel@gmail.com"
>>> if Author.objects.filter(email=email_to_check).exists():
...     print("Author with email", email_to_check, "exists.")
... else:
...     print("Author with email", email_to_check, "does not exist.")
...
Author with email Gabriel@gmail.com exists.
>>> book_name_to_check = "Love in the Time of Cholera"
>>> if Book.objects.filter(name=book_name_to_check).exists():
...     print("Book with name", book_name_to_check, "exists.")
... else:
...     print("Book with name", book_name_to_check, "does not exist.")
...
Book with name Love in the Time of Cholera exists.
>>> books = Book.objects.order_by('name')
>>> for book in books:
...     print(book.name)
...
Beloved
Harry Potter and the Deathly Hallows
Harry Potter and the Philosopher's Stone
Kafka on the Shore
Love in the Time of Cholera
Norwegian Wood
One Hundred Years of Solitude
Song of Solomon
The Shining
The Stand
>>> book_names_to_delete = ["Harry Potter and the Deathly Hallows", "The Stand"]
>>> books_to_delete = Book.objects.filter(name__in=book_names_to_delete)
>>> deleted_count, _ = books_to_delete.delete()
>>> print(f"{deleted_count} books deleted.")
2 books deleted.
```

-- update the books count value after fetching a book using its name

```
Harry Potter and the Philosopher's Stone
Kafka on the Shore
Love in the Time of Cholera
Norwegian Wood
One Hundred Years of Solitude
Song of Solomon
The Shining
The Stand
>>> book_names_to_delete = ["Harry Potter and the Deathly Hallows", "The Stand"]
>>> books_to_delete = Book.objects.filter(name__in=book_names_to_delete)
>>> deleted_count, _ = books_to_delete.delete()
>>> print(f"{deleted_count} books deleted.")
2 books deleted.
>>> book_name = "Song of Solomon"
>>> try:
...     book = Book.objects.get(name=book_name)
...     book.books_count += 1
...     book.save()
...     print("Books count for", book_name, "updated successfully.")
... except Book.DoesNotExist:
...     print("Book with name", book_name, "does not exist.")
...
Traceback (most recent call last):
  File "<console>", line 3, in <module>
AttributeError: 'Book' object has no attribute 'books_count'
>>> book_name = "Song of Solomon"
>>> try:
...     book = Book.objects.get(name=book_name)
...     book.count += 1
...     book.save()
...     print("Books count for", book_name, "updated successfully.")
... except Book.DoesNotExist:
...     print("Book with name", book_name, "does not exist.")
...
Books count for Song of Solomon updated successfully.
>>>
```

-- filter and show the name of books that has books count less than or equal to 20

-- show all the authors that has first name starting wth "a"

```
2 books deleted.
>>> book_name = "Song of Solomon"
>>> try:
...     book = Book.objects.get(name=book_name)
...     book.books_count += 1
...     book.save()
...     print("Books count for", book_name, "updated successfully.")
... except Book.DoesNotExist:
...     print("Book with name", book_name, "does not exist.")
...
Traceback (most recent call last):
  File "<console>", line 3, in <module>
AttributeError: 'Book' object has no attribute 'books_count'
>>> book_name = "Song of Solomon"
>>> try:
...     book = Book.objects.get(name=book_name)
...     book.count += 1
...     book.save()
...     print("Books count for", book_name, "updated successfully.")
... except Book.DoesNotExist:
...     print("Book with name", book_name, "does not exist.")
...
Books count for Song of Solomon updated successfully.
>>> books_with_count_less_than_20 = Book.objects.filter(count__lte=20).values_list('name', flat=True)
>>> for book_name in books_with_count_less_than_20:
...     print(book_name)
...
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 of Cholera
>>> authors_starting_with_a = Author.objects.filter(first_name__istartswith='a')
```

-- show all the authors that contains "d" in the first name

-- delete the books that has book count 0

```
>>>
>>>
>>>
>>>
>>> authors_with_d_in_first_name = Author.objects.filter(first_name__icontains='d')
>>> for author in authors_with_d_in_first_name:
...     print(author.first_name, author.last_name)
...
>>> authors_starting_with_a = Author.objects.filter(first_name__istartswith='a')
>>> for author in authors_starting_with_a:
...     print(author.first_name)
...
>>> authors_starting_with_a = Author.objects.filter(first_name__istartswith='t')
>>> for author in authors_starting_with_a:
...     print(author.first_name)
...
Toni
>>>
>>>
>>>
>>>
>>>
>>>
>>> authors_starting_with_a = Author.objects.filter(first_name__istartswith='t')
>>> for author in authors_starting_with_a:
...     print(author.first_name)
...
Toni
>>> authors_with_d_in_first_name = Author.objects.filter(first_name__icontains='d')
>>> for author in authors_with_d_in_first_name:
...     print(author.first_name, author.last_name)
...
>>> deleted_count, _ = Book.objects.filter(count=0).delete()
>>> print(f"{deleted_count} books deleted.")
0 books deleted.
>>>
```