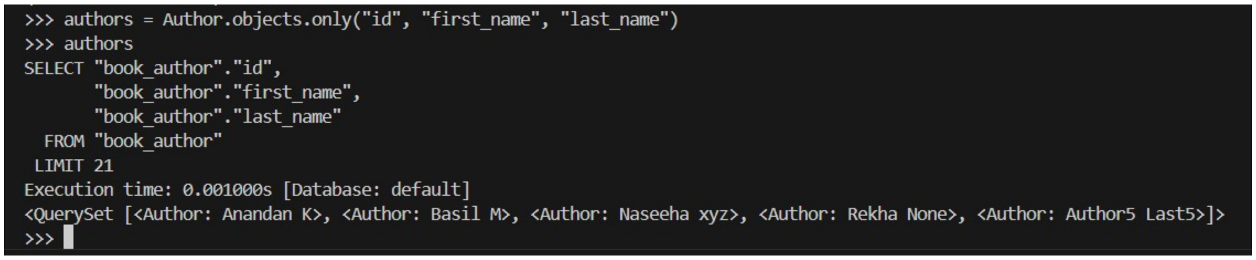
Django ORM Task 2

Note - avoid N+1 query problems

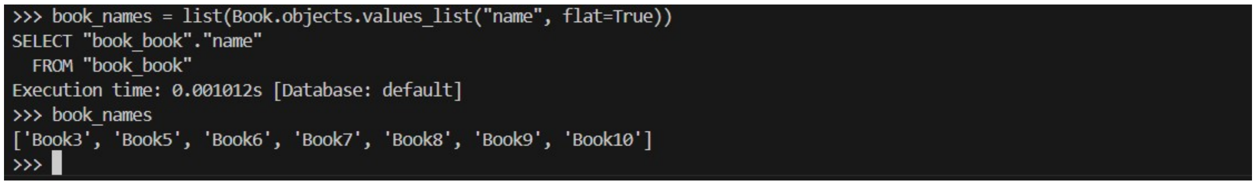
1. Using the shell plus and the django project you created, run these queries to:

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



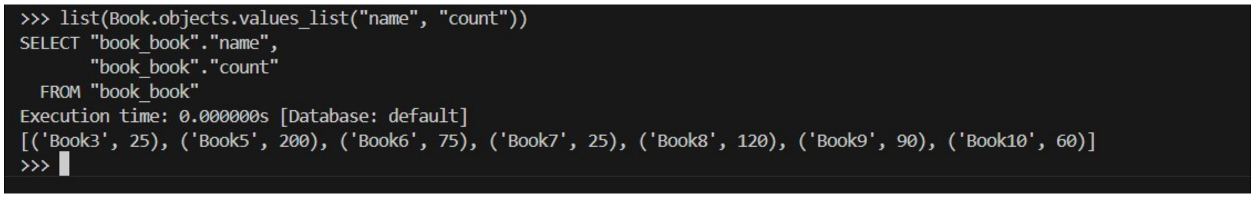
```
>>> authors = Author.objects.only("id", "first_name", "last_name")
>>> authors
SELECT "book_author"."id",
       "book_author"."first_name",
       "book_author"."last_name"
  FROM "book_author"
 LIMIT 21
Execution time: 0.001000s [Database: default]
<QuerySet [<Author: Anandan K>, <Author: Basil M>, <Author: Naseeha xyz>, <Author: Rekha None>, <Author: Author5 Last5>]>
>>> 
```

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



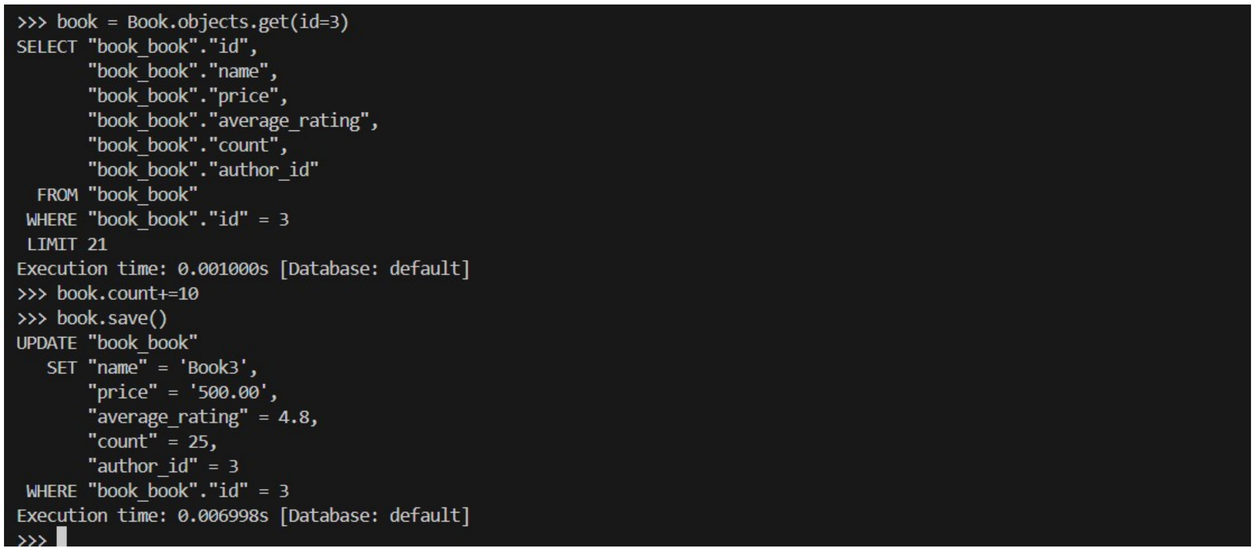
```
>>> book_names = list(Book.objects.values_list("name", flat=True))
SELECT "book_book"."name"
  FROM "book_book"
Execution time: 0.001012s [Database: default]
>>> book_names
['Book3', 'Book5', 'Book6', 'Book7', 'Book8', 'Book9', 'Book10']
>>> 
```

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



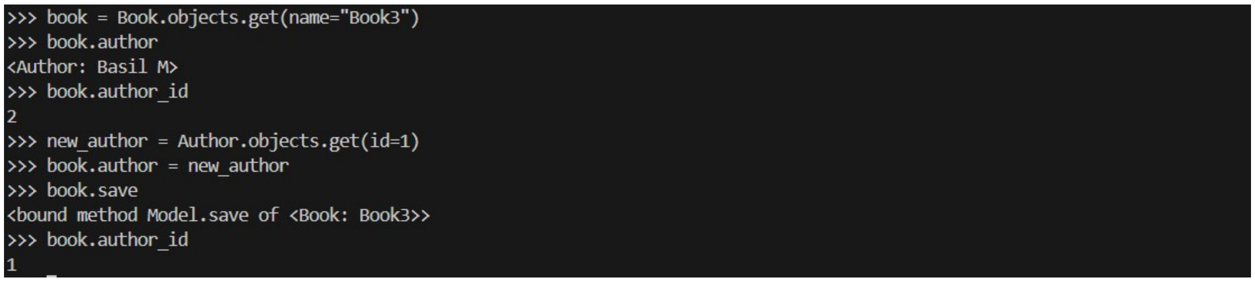
```
>>> list(Book.objects.values_list("name", "count"))
SELECT "book_book"."name",
       "book_book"."count"
  FROM "book_book"
Execution time: 0.000000s [Database: default]
[('Book3', 25), ('Book5', 200), ('Book6', 75), ('Book7', 25), ('Book8', 120), ('Book9', 90), ('Book10', 60)]
>>> 
```

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



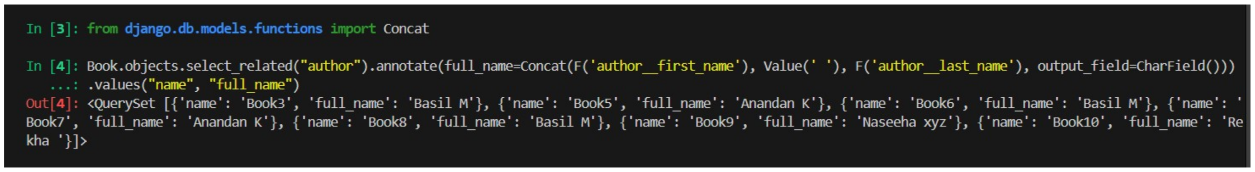
```
>>> book = Book.objects.get(id=3)
SELECT "book_book"."id",
       "book_book"."name",
       "book_book"."price",
       "book_book"."average_rating",
       "book_book"."count",
       "book_book"."author_id"
  FROM "book_book"
 WHERE "book_book"."id" = 3
 LIMIT 21
Execution time: 0.001000s [Database: default]
>>> book.count+=10
>>> book.save()
UPDATE "book_book"
   SET "name" = 'Book3',
       "price" = '500.00',
       "average_rating" = 4.8,
       "count" = 25,
       "author_id" = 3
 WHERE "book_book"."id" = 3
Execution time: 0.006998s [Database: default]
>>> 
```

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



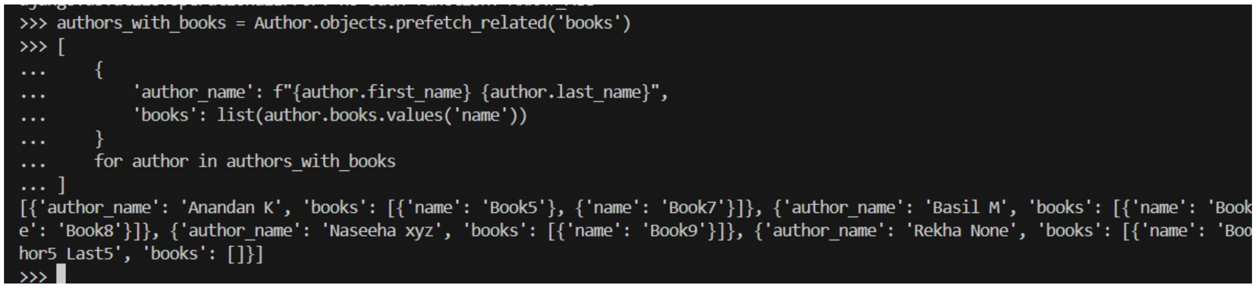
```
>>> book = Book.objects.get(name="Book3")
>>> book.author
<Author: Basil M>
>>> book.author_id
2
>>> new_author = Author.objects.get(id=1)
>>> book.author = new_author
>>> book.save
<bound method Model.save of <Book: Book3>>
>>> book.author_id
1
```

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



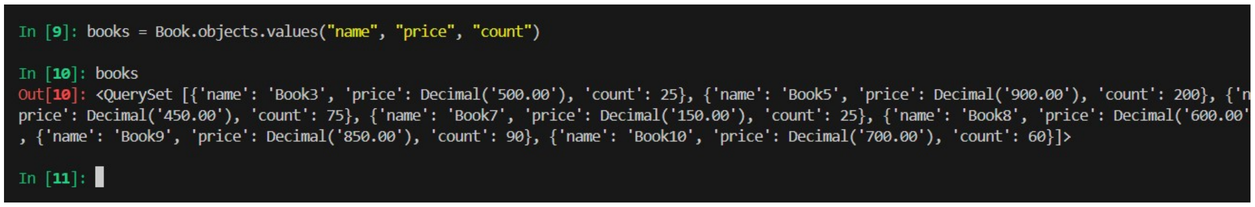
```
In [3]: from django.db.models.functions import Concat

In [4]: Book.objects.select_related("author").annotate(full_name=Concat(F('author__first_name'), Value(' '), F('author__last_name'), output_field=CharField()))
   ...: .values("name", "full_name")
Out[4]: <QuerySet [{'name': 'Book3', 'full_name': 'Basil M'}, {'name': 'Book5', 'full_name': 'Anandan K'}, {'name': 'Book6', 'full_name': 'Basil M'}, {'name': '
Book7', 'full_name': 'Anandan K'}, {'name': 'Book8', 'full_name': 'Basil M'}, {'name': 'Book9', 'full_name': 'Naseeha xyz'}, {'name': 'Book10', 'full_name': 'Re
kha '}]>
```

-- 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



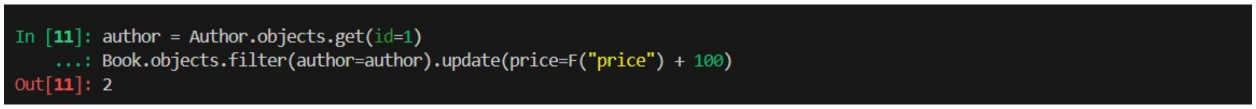
```
>>> authors_with_books = Author.objects.prefetch_related('books')
>>> [
...     {
...         'author_name': f"{author.first_name} {author.last_name}",
...         'books': list(author.books.values('name'))
...     }
...     for author in authors_with_books
... ]
[{'author_name': 'Anandan K', 'books': [{'name': 'Book5'}, {'name': 'Book7'}]}, {'author_name': 'Basil M', 'books': [{'name': 'Book
e': 'Book8'}]}, {'author_name': 'Naseeha xyz', 'books': [{'name': 'Book9'}]}, {'author_name': 'Rekha None', 'books': [{'name': 'Boo
hor5 Last5', 'books': []}]
>>> 
```

-- 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



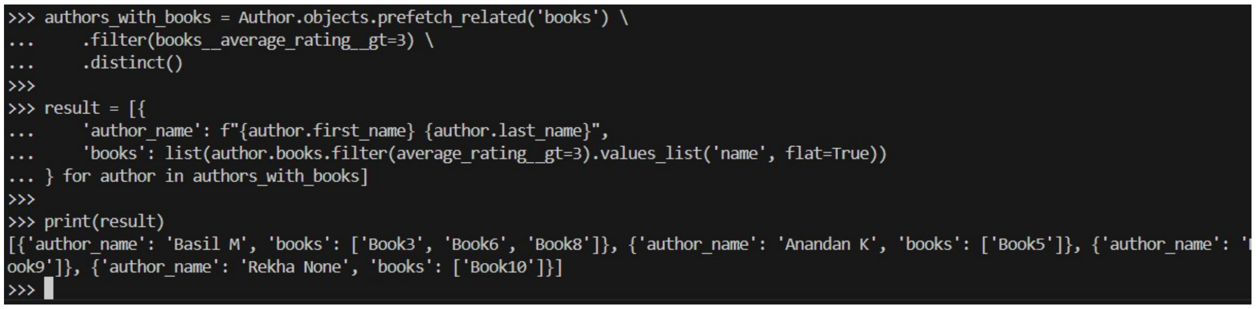
```
In [9]: books = Book.objects.values("name", "price", "count")

In [10]: books
Out[10]: <QuerySet [{'name': 'Book3', 'price': Decimal('500.00'), 'count': 25}, {'name': 'Book5', 'price': Decimal('900.00'), 'count': 200}, {'n
price': Decimal('450.00'), 'count': 75}, {'name': 'Book7', 'price': Decimal('150.00'), 'count': 25}, {'name': 'Book8', 'price': Decimal('600.00'
, {'name': 'Book9', 'price': Decimal('850.00'), 'count': 90}, {'name': 'Book10', 'price': Decimal('700.00'), 'count': 60}]>

In [11]: 
```

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



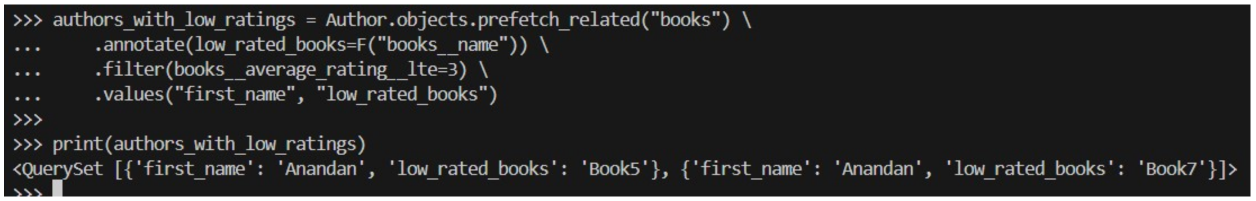
```
In [11]: author = Author.objects.get(id=1)
    ...: Book.objects.filter(author=author).update(price=F("price") + 100)
Out[11]: 2
```

-- 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



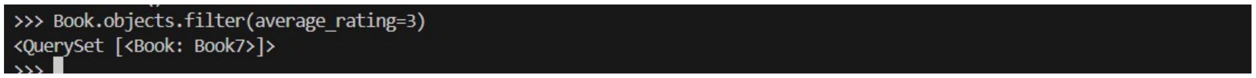
```
>>> authors_with_books = Author.objects.prefetch_related('books') \
...     .filter(books__average_rating__gt=3) \
...     .distinct()
>>>
>>> result = [{
...     'author_name': f"{author.first_name} {author.last_name}",
...     'books': list(author.books.filter(average_rating__gt=3).values_list('name', flat=True))
... } for author in authors_with_books]
>>>
>>> print(result)
[{'author_name': 'Basil M', 'books': ['Book3', 'Book6', 'Book8']}, {'author_name': 'Anandan K', 'books': ['Book5']}, {'author_name': '
ook9']}, {'author_name': 'Rekha None', 'books': ['Book10']}]
>>>
```

-- 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



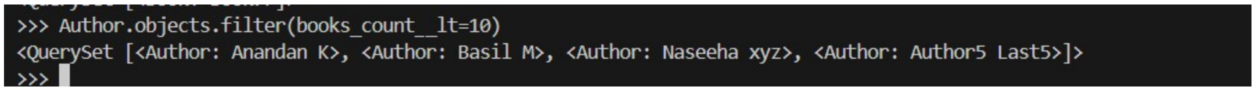
```
>>> authors_with_low_ratings = Author.objects.prefetch_related("books") \
...     .annotate(low_rated_books=F("books__name")) \
...     .filter(books__average_rating__lte=3) \
...     .values("first_name", "low_rated_books")
>>>
>>> print(authors_with_low_ratings)
<QuerySet [{'first_name': 'Anandan', 'low_rated_books': 'Book5'}, {'first_name': 'Anandan', 'low_rated_books': 'Book7'}]>
>>>
```

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



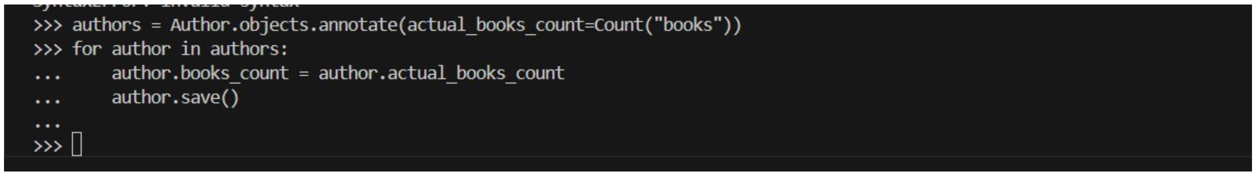
```
>>> Book.objects.filter(average_rating=3)
<QuerySet [<Book: Book7>]>
>>>
```

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



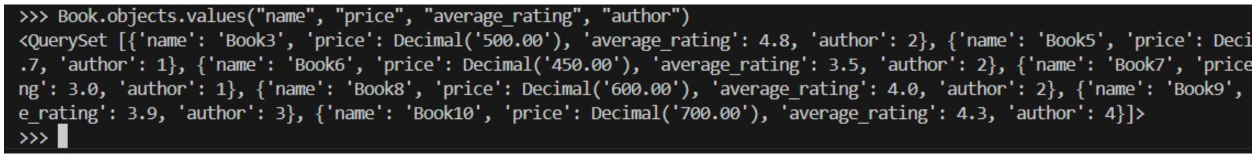
```
>>> Author.objects.filter(books_count__lt=10)
<QuerySet [<Author: Anandan K>, <Author: Basil M>, <Author: Naseeha xyz>, <Author: Author5 Last5>]>
>>>
```

-- 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



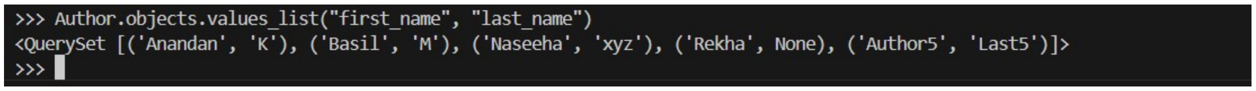
```
>>> authors = Author.objects.annotate(actual_books_count=Count("books"))
>>> for author in authors:
...     author.books_count = author.actual_books_count
...     author.save()
...
>>>
```

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



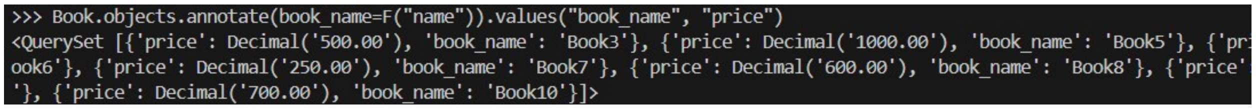
```
>>> Book.objects.values("name", "price", "average_rating", "author")
<QuerySet [{'name': 'Book3', 'price': Decimal('500.00'), 'average_rating': 4.8, 'author': 2}, {'name': 'Book5', 'price': Deci
.7, 'author': 1}, {'name': 'Book6', 'price': Decimal('450.00'), 'average_rating': 3.5, 'author': 2}, {'name': 'Book7', 'price
ng': 3.0, 'author': 1}, {'name': 'Book8', 'price': Decimal('600.00'), 'average_rating': 4.0, 'author': 2}, {'name': 'Book9',
e_rating': 3.9, 'author': 3}, {'name': 'Book10', 'price': Decimal('700.00'), 'average_rating': 4.3, 'author': 4}]>
>>>
```

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



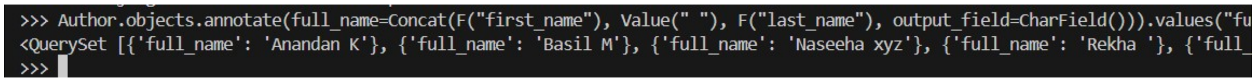
```
>>> Author.objects.values_list("first_name", "last_name")
<QuerySet [('Anandan', 'K'), ('Basil', 'M'), ('Naseeha', 'xyz'), ('Rekha', None), ('Author5', 'Last5')]>
>>>
```

-- 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



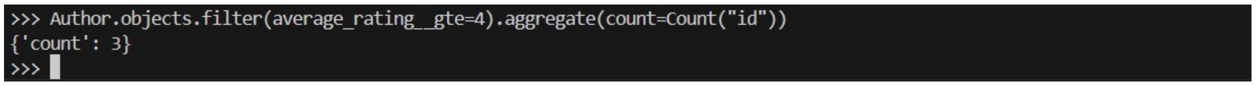
```
>>> Book.objects.annotate(book_name=F("name")).values("book_name", "price")
<QuerySet [{'price': Decimal('500.00'), 'book_name': 'Book3'}, {'price': Decimal('1000.00'), 'book_name': 'Book5'}, {'pr
ook6'}, {'price': Decimal('250.00'), 'book_name': 'Book7'}, {'price': Decimal('600.00'), 'book_name': 'Book8'}, {'price'
'}, {'price': Decimal('700.00'), 'book_name': 'Book10'}]>
```

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



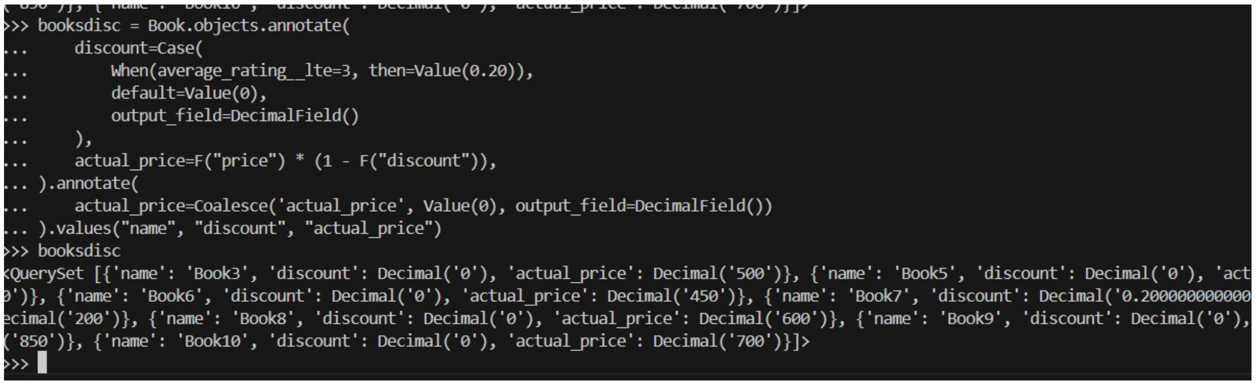
```
>>> Author.objects.annotate(full_name=Concat(F("first_name"), Value(" "), F("last_name"), output_field=CharField())).values("fu
<QuerySet [{'full_name': 'Anandan K'}, {'full_name': 'Basil M'}, {'full_name': 'Naseeha xyz'}, {'full_name': 'Rekha '}, {'full_
>>>
```

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



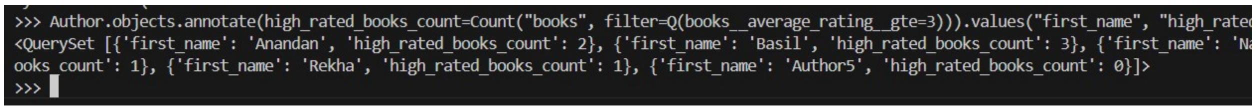
```
>>> Author.objects.filter(average_rating__gte=4).aggregate(count=Count("id"))
{'count': 3}
>>>
```

-- 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 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



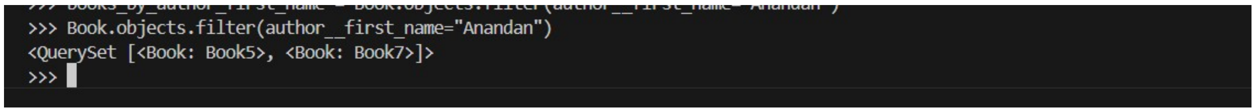
```
>>> booksdisc = Book.objects.annotate(
...     discount=Case(
...         When(average_rating__lte=3, then=Value(0.20)),
...         default=Value(0),
...         output_field=DecimalField()
...     ),
...     actual_price=F("price") * (1 - F("discount")),
... ).annotate(
...     actual_price=Coalesce('actual_price', Value(0), output_field=DecimalField())
... ).values("name", "discount", "actual_price")
>>> booksdisc
<QuerySet [{'name': 'Book3', 'discount': Decimal('0'), 'actual_price': Decimal('500')}, {'name': 'Book5', 'discount': Decimal('0'), 'act
0')}, {'name': 'Book6', 'discount': Decimal('0'), 'actual_price': Decimal('450')}, {'name': 'Book7', 'discount': Decimal('0.200000000000
ecimal('200')}, {'name': 'Book8', 'discount': Decimal('0'), 'actual_price': Decimal('600')}, {'name': 'Book9', 'discount': Decimal('0'),
('850')}, {'name': 'Book10', 'discount': Decimal('0'), 'actual_price': Decimal('700')}]>
>>>
```

-- 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



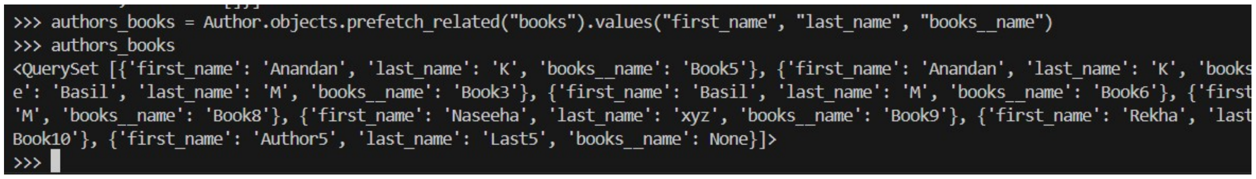
```
>>> Author.objects.annotate(high_rated_books_count=Count("books", filter=Q(books__average_rating__gte=3))).values("first_name", "high_rated
<QuerySet [{'first_name': 'Anandan', 'high_rated_books_count': 2}, {'first_name': 'Basil', 'high_rated_books_count': 3}, {'first_name': 'Na
ooks_count': 1}, {'first_name': 'Rekha', 'high_rated_books_count': 1}, {'first_name': 'Author5', 'high_rated_books_count': 0}]>
>>> 
```

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



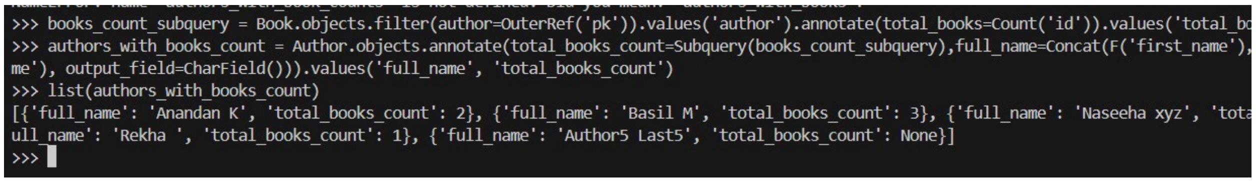
```
>>> Book.objects.filter(author__first_name="Anandan")
<QuerySet [<Book: Book5>, <Book: Book7>]>
>>> 
```

-- fetch all the authors along with their books



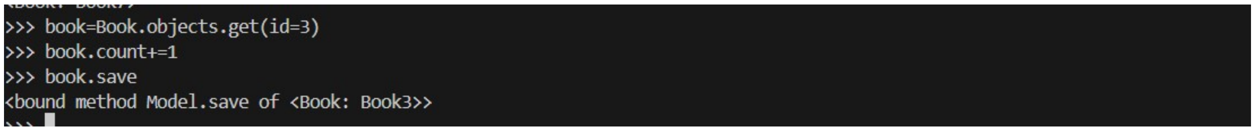
```
>>> authors_books = Author.objects.prefetch_related("books").values("first_name", "last_name", "books__name")
>>> authors_books
<QuerySet [{'first_name': 'Anandan', 'last_name': 'K', 'books__name': 'Book5'}, {'first_name': 'Anandan', 'last_name': 'K', 'books
e': 'Basil', 'last_name': 'M', 'books__name': 'Book3'}, {'first_name': 'Basil', 'last_name': 'M', 'books__name': 'Book6'}, {'first
'M', 'books__name': 'Book8'}, {'first_name': 'Naseeha', 'last_name': 'xyz', 'books__name': 'Book9'}, {'first_name': 'Rekha', 'last
Book10'}, {'first_name': 'Author5', 'last_name': 'Last5', 'books__name': None}]>
>>> 
```

-- 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



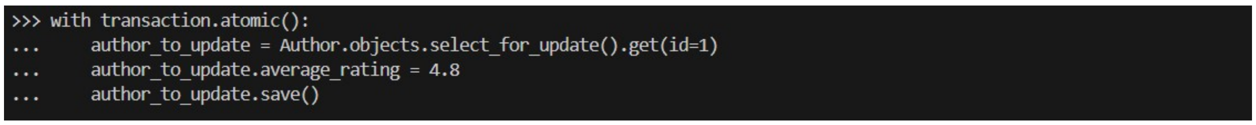
```
>>> books_count_subquery = Book.objects.filter(author=OuterRef('pk')).values('author').annotate(total_books=Count('id')).values('total_bo
>>> authors_with_books_count = Author.objects.annotate(total_books_count=Subquery(books_count_subquery),full_name=Concat(F('first_name'),
me'), output_field=CharField())).values('full_name', 'total_books_count')
>>> list(authors_with_books_count)
[{'full_name': 'Anandan K', 'total_books_count': 2}, {'full_name': 'Basil M', 'total_books_count': 3}, {'full_name': 'Naseeha xyz', 'tota
ull_name': 'Rekha ', 'total_books_count': 1}, {'full_name': 'Author5 Last5', 'total_books_count': None}]
>>> 
```

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



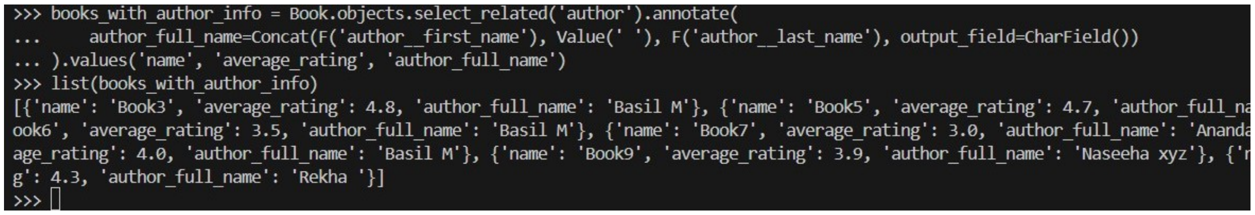
```
>>> book=Book.objects.get(id=3)
>>> book.count+=1
>>> book.save
<bound method Model.save of <Book: Book3>>
```

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



```
>>> with transaction.atomic():
...     author_to_update = Author.objects.select_for_update().get(id=1)
...     author_to_update.average_rating = 4.8
...     author_to_update.save()
```

-- 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_with_author_info = Book.objects.select_related('author').annotate(
...     author_full_name=Concat(F('author__first_name'), Value(' '), F('author__last_name'), output_field=CharField())
... ).values('name', 'average_rating', 'author_full_name')
>>> list(books_with_author_info)
[{'name': 'Book3', 'average_rating': 4.8, 'author_full_name': 'Basil M'}, {'name': 'Book5', 'average_rating': 4.7, 'author_full_na
ook6', 'average_rating': 3.5, 'author_full_name': 'Basil M'}, {'name': 'Book7', 'average_rating': 3.0, 'author_full_name': 'Ananda
age_rating': 4.0, 'author_full_name': 'Basil M'}, {'name': 'Book9', 'average_rating': 3.9, 'author_full_name': 'Naseeha xyz'}, {'n
g': 4.3, 'author_full_name': 'Rekha '}]
>>> 
```