1. Useful Methods:

- filter - retrieves a subset of objects from DB, takes one or more keyword arguments, returns queryset

```
Employee.objects.filter(job_level='Sr.')

def find_books_by_genre_and_language(book_genre, book_language):
    return Book.objects.filter(genre=book_genre, language=book_language)
```

- exclude - retrieves a subset of objects from DB, takes one or more keyword arguments, returns queryset

```
Employee.objects.exclude(job_level='Sr.')

def find_authors_nationalities():
    authors = Author.objects.exclude(nationality=None)
    # authors = Author.objects.filter(nationality__isnull=False)
    result = [
        f"{author.first_name} {author.last_name} is {author.nationality}"
        for author in authors
    ]

    return "\n".join(result)
```

- order - retrieves objects from the DB in a specific order, takes one or more field names as arguments, returns a queryset

```
Employee.objects.order_by('-last_name')

def order_books_by_year():
    books = Book.objects.order_by('publication_year', 'title')
    result = [
        f"{book.publication_year} year: {book.title} by {book.author}"
        for book in books
    ]
    return "\n".join(result)
```

- count - retrieves the number of objects that match a specific query or filter condition, returns int
    - more efficient than len() method

```
Employee.objects.count()
```

- get - selecting a single object
    - MultipleObjectsReturned Exception

```
employee = Employee.objects.get(id=2)
print(employee.id)
print(employee.pk)


def delete_review_by_id(reviewer_id):
    # Review.objects.filter(pk=reviewer_id).delete()
    review = Review.objects.get(id=reviewer_id)
    review.delete()
    return f"Review by {review.reviewer_name} was deleted."
```

- chaining methods - construct complex queries

```
def filter_authors_by_nationalities(nationality):
    authors =
```

```
Author.objects.filter(nationality=nationality).order_by('first_name', 'last_name')
                  result = [
                        author.biography
                        if author.biography is not None
                        else f"{author.first_name} {author.last_name}"
                        for author in authors
                  ]
                  return "n".join(result)


2. Lookup Keys(Field Lookups):
      employees = Employee.objects.filter(id__lte=5)

      - exact
      - iexact
      - contains
      - icontains
      - in
      - lt / lte / gt /gte
      - startswith / endswith
      - istartswith / iendswith
      - range
      - date
      - year
      - isnull

      def filter_authors_by_birth_year(first_year, end_year):
            #
Author.objects.filter(birth_date__year__gte=first_year).filter(birth_date__year__lt
e=end_year)
            authors = Author.objects.filter(
                  birth_date__year__range=(first_year, end_year)
            ).order_by('-birth_date')

            result = [
                  f"{author.birth_date}: {author.first_name} {author.last_name}"
                  for author in authors
            ]

            return "\n".join(result)


3. Bulk Methods
      - bulk create, update, delete

      bulk_employees = Employee.objects.bulk_create(new_employees)
      Employee.objects.filter(job_level='Jr.').update(job_level='Sr.)

      def change_reviewer_name(reviewer_name, reviewer_new_name):


Review.objects.filter(reviewer_name=reviewer_name).update(reviewer_new_name=reviewe
r_new_name)

            return Review.objects.all()


******
```

```
from django.db.models import Q

names = ['john', 'jane']

query = Q()

for name in names:
      query.add(Q(first_name__icontains=name), Q.OR)
print(query)
print('result', Author.objects.filter(query))
```