1 select_related

Suppose we have the following `Author` and `Book` models:

```
class Author(models.Model):
    name = models.CharField(max_length=100)

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.ForeignKey(Author,
                               on_delete=models.CASCADE)
```

We want to retrieve a specific book object along with its associated author object using `select_related`. We can do this using the `select_related` method in our query:

```
book = Book.objects.select_related('author').get(id=1)
```

This generates the following SQL query:

```
SELECT books.id, books.title, books.author_id, authors.id, authors.name
FROM books
INNER JOIN authors ON books.author_id = authors.id
WHERE books.id = 1
```

As we can see, this query uses an INNER JOIN to retrieve the related author object along with the book object in a single query. The `SELECT` statement includes columns from both the `books` and `authors` tables, and the `INNER JOIN` clause ensures that the results include only those rows where the `author_id` column in the `books` table matches the `id` column in the `authors` table.

Now we can access the related author object through the `author` attribute of the `book` object, without any additional database hits:

```
author_name = book.author.name
```

This will return the name of the author associated with the book, without requiring another query to the database.

2 prefetch_related

Let's consider the following `Book` and `Genre` models:

```
class Book(models.Model):
    title = models.CharField(max_length=100)
    genres = models.ManyToManyField(Genre)
```

```
class Genre(models.Model):
    name = models.CharField(max_length=100)
```

We want to retrieve a specific book object along with all of its associated genre objects using `prefetch_related`. We can do this using the `prefetch_related` method in our query:

```
book = Book.objects.prefetch_related('genres').get(id=1)
```

This generates the following SQL queries:

SELECT *books.id*, *books.title*
FROM *books*
WHERE *books.id* = 1

SELECT *genres.id*, *genres.name*, *book_genres.book_id* AS *_prefetch_related_val_book_id*
FROM *genres*
INNER JOIN *book_genres* ON *genres.id* = *book_genres.genre_id*
WHERE *book_genres.book_id* IN (1)

The first query retrieves the book object, and the second query fetches all related genre objects for the book using an INNER JOIN. Note that the '_prefetch_related_val_book_id' alias is used to avoid conflicts with the 'id' column in the 'genres' table.

Now we can access all of the related genre objects through the 'genres' attribute of the 'book' object, without any additional database hits:

```
genres = book.genres.all()
```

This will return a queryset containing all of the genre objects associated with the book, without requiring another query to the database.

TL;DR

- select_related: reduces the number of queries to 1 by fetching related objects in a single query.
- prefetch_related: adds an additional query to fetch related objects, but can reduce the total number of queries by fetching related objects in a separate query.