Explain the use of the save() method in Django models.

The save() method in Django models persists model instance data to the database. It handles both **inserting new records** and **updating existing ones** (auto-detected via primary key).

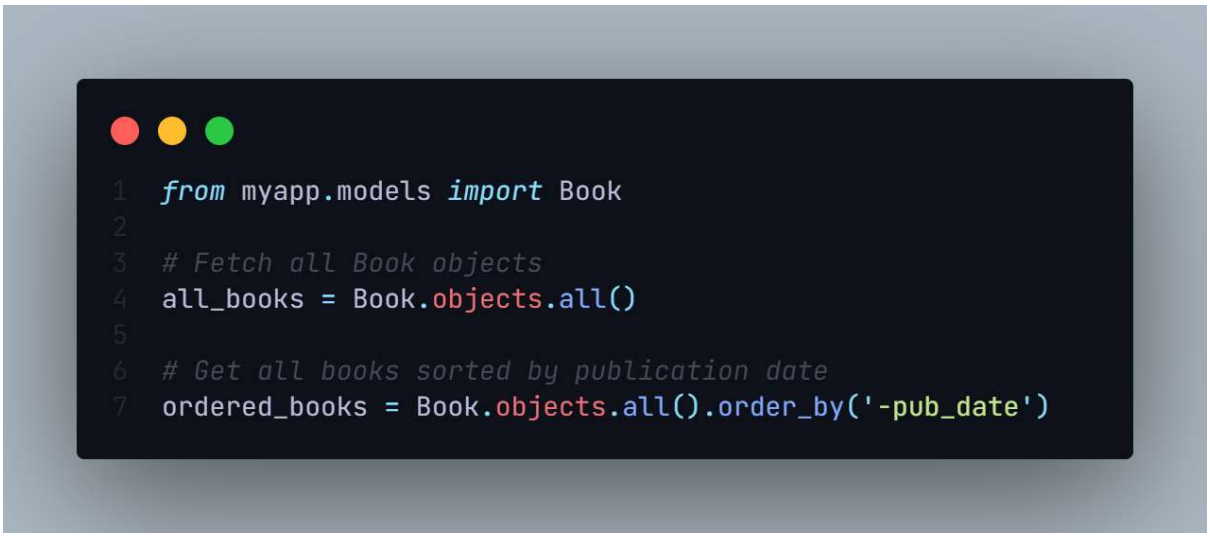
Key Uses:

1. **Create/Update**:
    - obj.save() → Inserts a new row if obj.id is None; updates existing row otherwise.

save() is the primary way to commit model changes to the database, with hooks for customization.

How do you retrieve all objects of a specific model from the database?

To retrieve **all objects** of a specific Django model from the database, use the all() method on the model's **default manager** (usually objects). This returns a **QuerySet** containing all instances of the model.



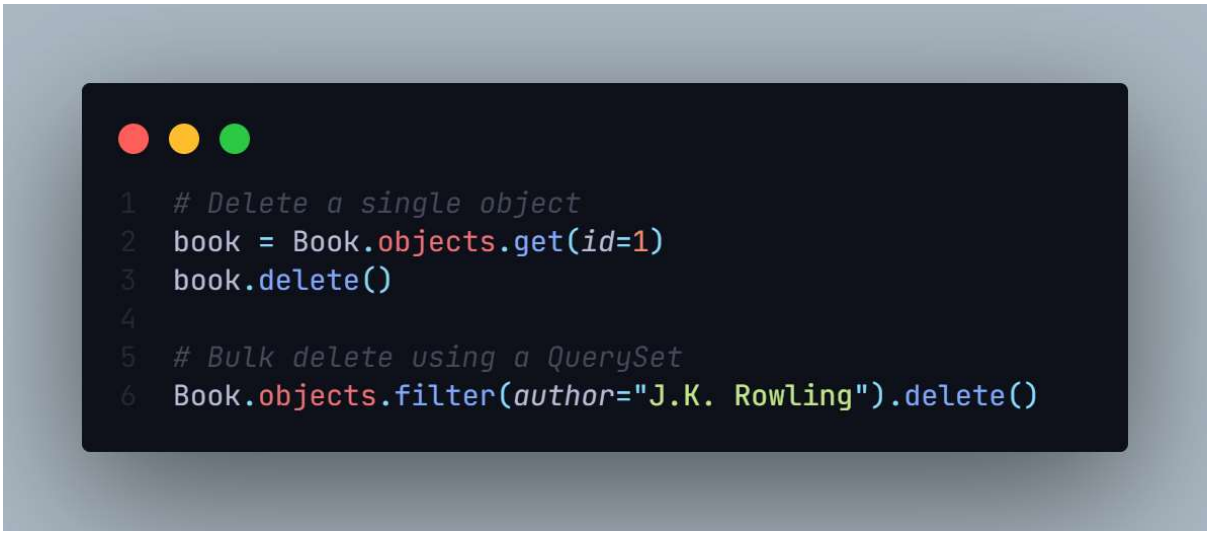
```python
from myapp.models import Book

# Fetch all Book objects
all_books = Book.objects.all()

# Get all books sorted by publication date
ordered_books = Book.objects.all().order_by('-pub_date')
```

How can you delete an object from a Django model?

1. **Retrieve the object** (e.g., using get() or filter()).
2. **Call delete()** on the instance or QuerySet.

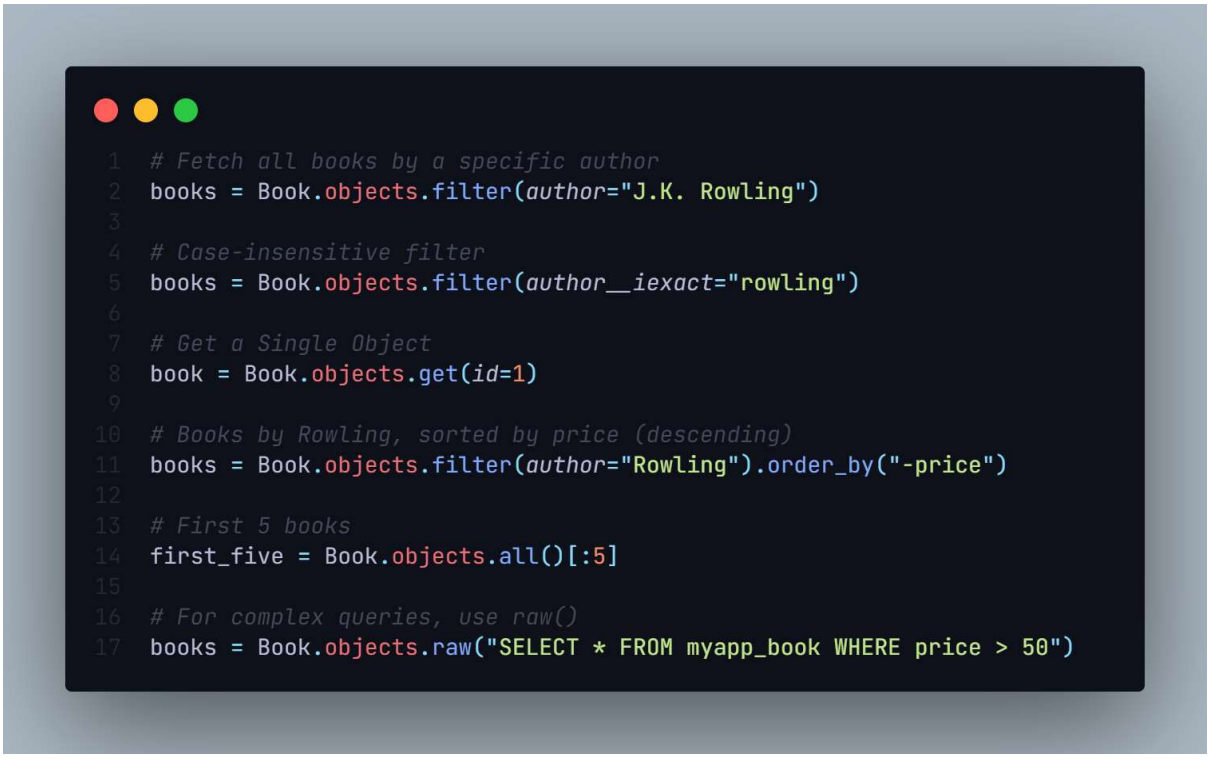


```python
# Delete a single object
book = Book.objects.get(id=1)
book.delete()

# Bulk delete using a QuerySet
Book.objects.filter(author="J.K. Rowling").delete()
```

How do you filter and query data from a Django model?

Use the filter() method to retrieve objects matching specific conditions:



```python
# Fetch all books by a specific author
books = Book.objects.filter(author="J.K. Rowling")

# Case-insensitive filter
books = Book.objects.filter(author__iexact="rowling")

# Get a Single Object
book = Book.objects.get(id=1)

# Books by Rowling, sorted by price (descending)
books = Book.objects.filter(author="Rowling").order_by("-price")

# First 5 books
first_five = Book.objects.all()[:5]

# For complex queries, use raw()
books = Book.objects.raw("SELECT * FROM myapp_book WHERE price > 50")
```