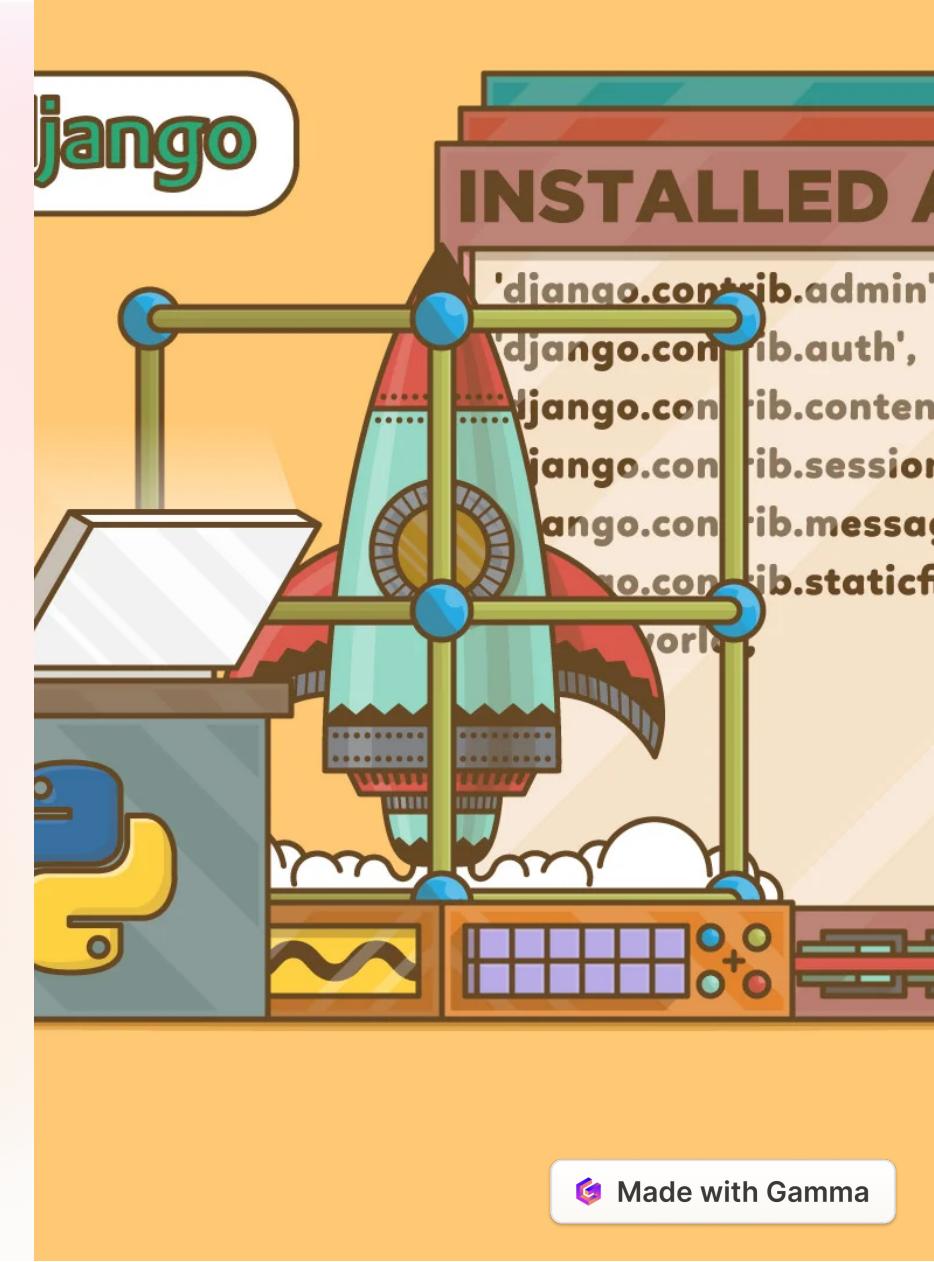


# Models and Databases in Django

prepared by :

ENG : / Ghada Atef



A Django model is a table in your database.

Up until now in this tutorial, output has been static data from Python or HTML templates.

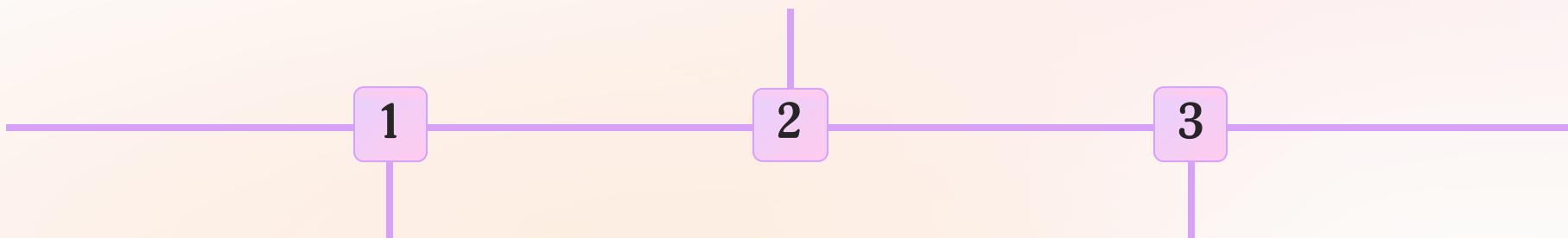
Now we will see how Django allows us to work with data, without having to change or upload files in the process.

In Django, data is created in objects, called Models, and is actually tables in a database.

# Django Models

## Creating models and fields

Models are created as classes that inherit from the Django `models.Model` class. Fields define the data that a table can store, such as text, numbers, or dates.



### Definition and Purpose

Models are the blueprint for creating database tables in Django. Each model represents a table in the database.

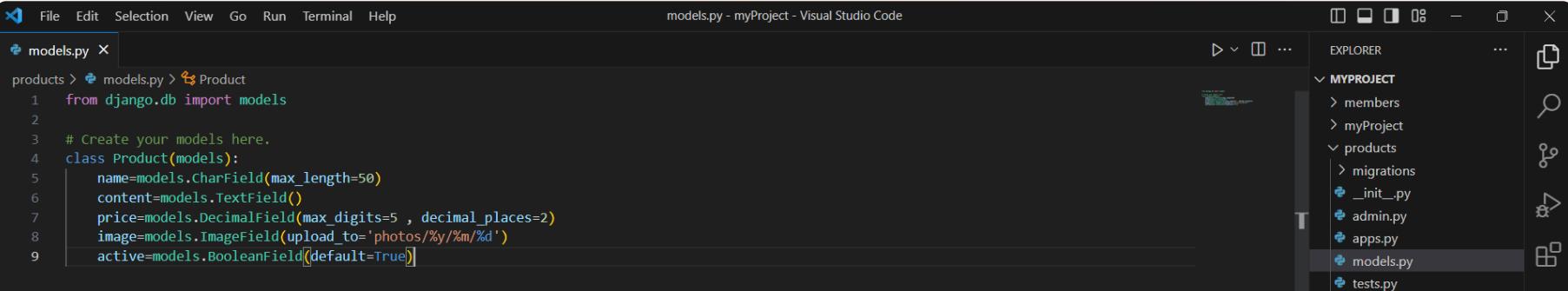
### Database operations in Django

Django provides many built-in methods to create, read, update, and delete data from the database.

# Create Table (Model)

To create a model, navigate to the `models.py` file in the `/products/` folder. (new app)

Open it, and add a Product table by creating a `Product` class, and describe the table fields in it:



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** models.py - myProject - Visual Studio Code.
- Code Editor:** The `models.py` file is open, showing Python code for a `Product` model. The code includes imports from `django.db.models` and defines a `Product` class with fields: `name`, `content`, `price`, `image`, and `active`.
- Explorer Panel:** Shows the project structure under `MYPROJECT`. The `products` app contains `migrations`, `__init__.py`, `admin.py`, `apps.py`, `models.py`, and `tests.py`.

# SQLite Database

When we created the Django project, we got an empty SQLite database.

It was created in the `myproject` root folder, and has the filename `db.sqlite3`.

By default, all Models created in the Django project will be created as tables in this database.

# Migrate

Now when we have described a Model in the `models.py` file, we must run a command to actually create the table in the database.

Navigate to the `/myproject/` folder and run this command:

```
py manage.py makemigrations
```

may be appear error like this

The screenshot shows a Visual Studio Code interface with a Python Django project named "myProject". The code editor displays `models.py` containing the following code:

```
products > models.py > Product
1  from django.db import models
2  class Product(models.Model):
3      name=models.CharField(max_length=50)
4      content=models.TextField()
5      price=models.DecimalField(max_digits=5 , decimal_places=2)
6      image=models.ImageField(upload_to='photos/%y/%m/%d')
7      active=models.BooleanField(default=True)
```

The terminal window shows a `SystemCheckError` message:

```
SystemCheckError: System check identified some issues:
ERRORS:
products.Product.image: (fields.E210) Cannot use ImageField because Pillow is not installed.
    HINT: Get Pillow at https://pypi.org/project/Pillow/ or run command "python -m pip install Pillow".
PS C:\Users\HP 15\myProject>
```

to solve this we should write this command :

```
pip install Pillow
```

The screenshot shows the same Visual Studio Code interface after running the `pip install Pillow` command. The terminal window now displays the output of the pip installation:

```
SystemCheckError: System check identified some issues:
ERRORS:
products.Product.image: (fields.E210) Cannot use ImageField because Pillow is not installed.
    HINT: Get Pillow at https://pypi.org/project/Pillow/ or run command "python -m pip install Pillow".
PS C:\Users\HP 15\myProject> pip install Pillow
Defaulting to user installation because normal site-packages is not writeable
Collecting Pillow
  Downloading Pillow-10.0.0.tar.gz (50.5 MB)
    41.5/50.5 MB 433.6 kB/s eta 0:00:21
```

Django creates a file describing the changes and stores the file in the /migrations/ folder:

Run the migrate command:

```
py manage.py migrate
```

Now you have a Product table in your database!

# Django ORM



## What is ORM?

ORM stands for Object-Relational Mapping. In Django, the ORM is used to translate between Python objects and database tables.



## Advantages of using ORM

ORM allows developers to write database queries using Python instead of SQL, which makes querying the database much easier and less error-prone.

A screenshot of a code editor showing a file named "base\_user.py". The code defines a "BaseUser" model with fields for password, name, and is\_active. It also includes methods for authenticate and save, and a Meta class. The code is color-coded for syntax highlighting.

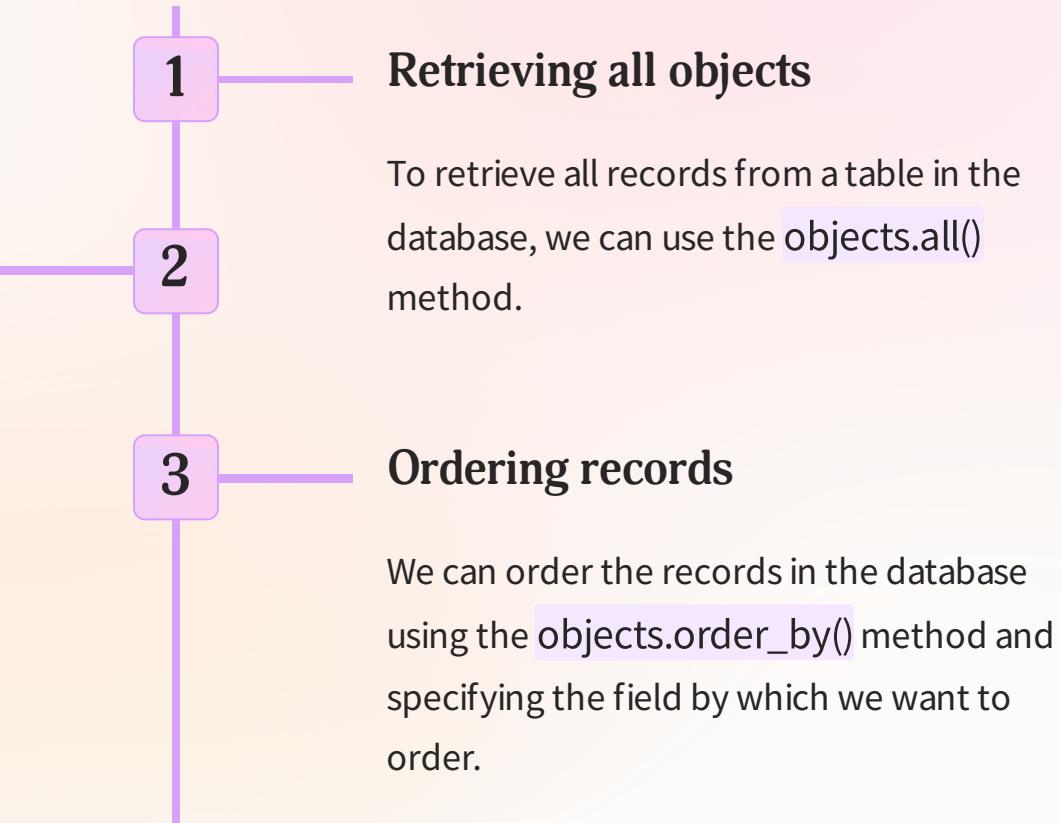
## Sample code snippet

To create a new record in the database using Django ORM, we can use the following code:

# Querying the database using Django

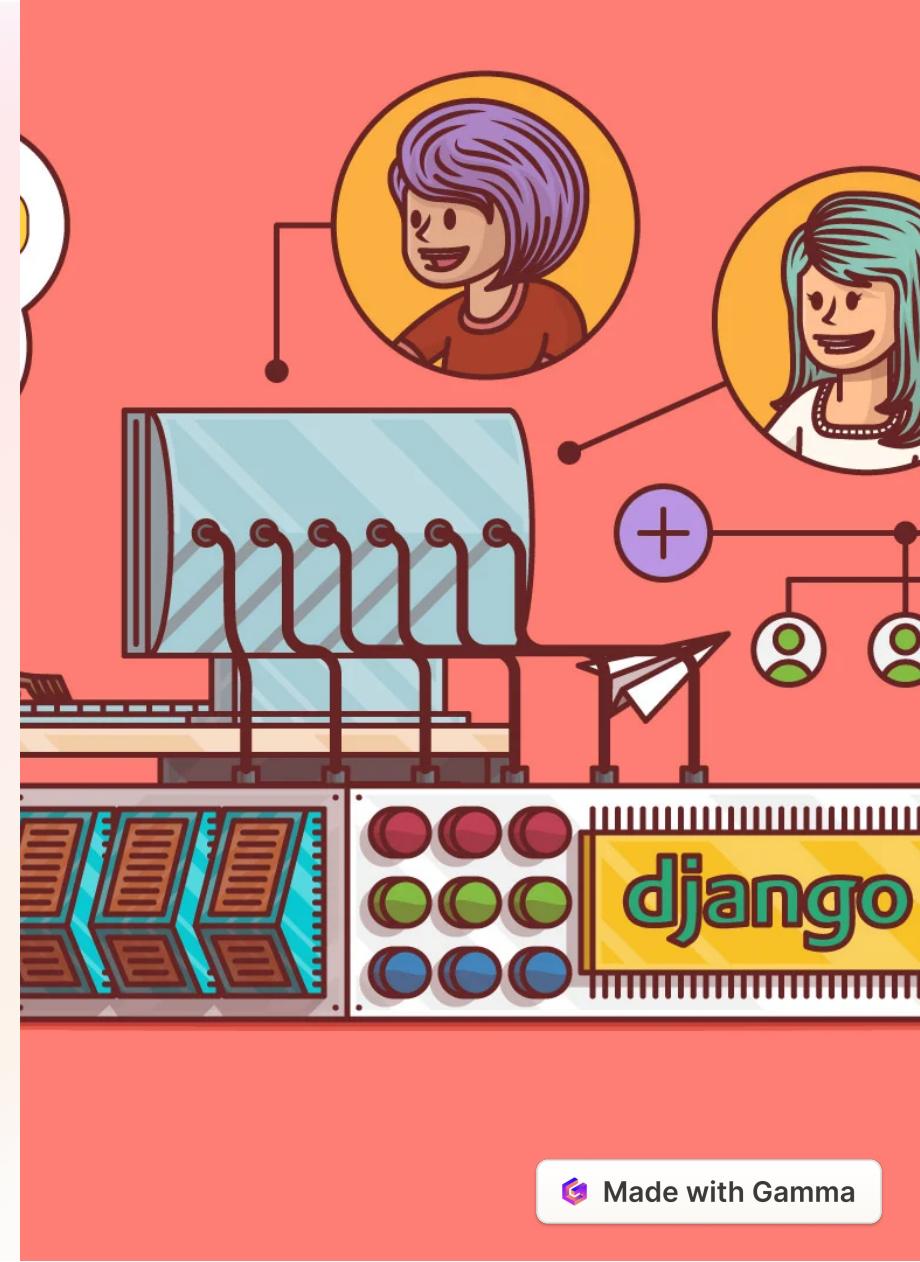
## Filtering records

We can filter records in the database using the `objects.filter()` method and specifying the conditions for the filter.



# Conclusion

Models and databases are a crucial aspect of web development with Django. By following best practices and utilizing Django's built-in tools, we can create efficient and maintainable web applications that scale with ease.



# Thank You

Any Questions ?