Django builtin database sqlite. MySQL er chotovai.

Ekta file e multiple table thakle kind of model bola jai.

ORM: object relational mapper is a programming technique that helps application to interact with database such as MySQL, SQLite etc.

* Create a database schema from defined classes or models
* Generate SQL from Python code form a particular database which means developers do not need to write sql code
* Helps to change database easily
* Use connectors to connect databases with a web application

QuerySet: it can be defined as a list containing all those objects we have created using the Django model.

It helps us to:

* Read data from the database
* Filter it
* Order it

Model: a model is the single, definitive source of information about our data.

It contains the:

* Essential fields and behaviors of the data
* Each model maps to a single database table

Model class:

* It is a class which will represent a table in database
* Each model is a python class that subclasses Django.db.models.Model

From Django.db import models. Inherit models.Model

* Each attribute represents a database field
* Django gives automatically-generated database-access API
* Django provides SQLite database by default.
* We can use other databases like MySQL, Oracle SQL etc.

Inside class we can specify which field will be primary key. If we don’t Django automatically make a primary key named ID and uses auto increment.

**Migrations:**

Migrations are way of propagating changes to make models(adding a field, deleting a model, etc.) into our database schema.

Makemigrations: is used to convert model class into sql statements. Create a file which will contain sql statements. This file is located in Applications migrations folder.

*Python manage.py makemigrations*

Migrate: is used to execute sql statements generated by makemigrations

*Python manage.py migrate*

Model class to SQL on makemigrations command. SQL run on migrate command. Every single change er por ei duita command dite hobe.

Showmigrations: this lists a project’s migrations

Built in Field Options:

Null: contain either True or False. If true Django will store empty values as NULL in the database. Default is False. True means must enter this data.

Blank: contain either true or false. If true, the field is allowed to be blank and no validation is done. False: cant go forward without validation.

Note: null is purely database related, whereas blank is validation related.

Default: default value for the field.

Verbose\_name: a human readable name for the field. If the verbose name isn’t given, Django will automatically create it using the fields attribute name, converting underscores to spaces.

Db\_column: the name of the database column to use for this field, if not given, Django will use the fields name. eta diye value access kora jabe.

Primary\_key: if true, that field will be the primary key for the model.

Unique: if true, this field must be unique throughout the table. This is enforced at the database level and by model validation.

Model form: in Django, a model form is a form that is automatically generated from an existing model.

A model form includes all the fields defined in the model, along with any additional validation and processing logic that you define in the form class.

Meta class: is a class that defines how other classes should behave, provides additional information about the model form. Here are some common option that can be defined in the meta class:

Model: the model that the form is based on.

Fields: a list of fields to include in the form. If not specified, all fields in the model will be included in the form.

Exclude: a list of fields to exclude from the form.

Widgets, Labels, Help\_texts, error\_messages.