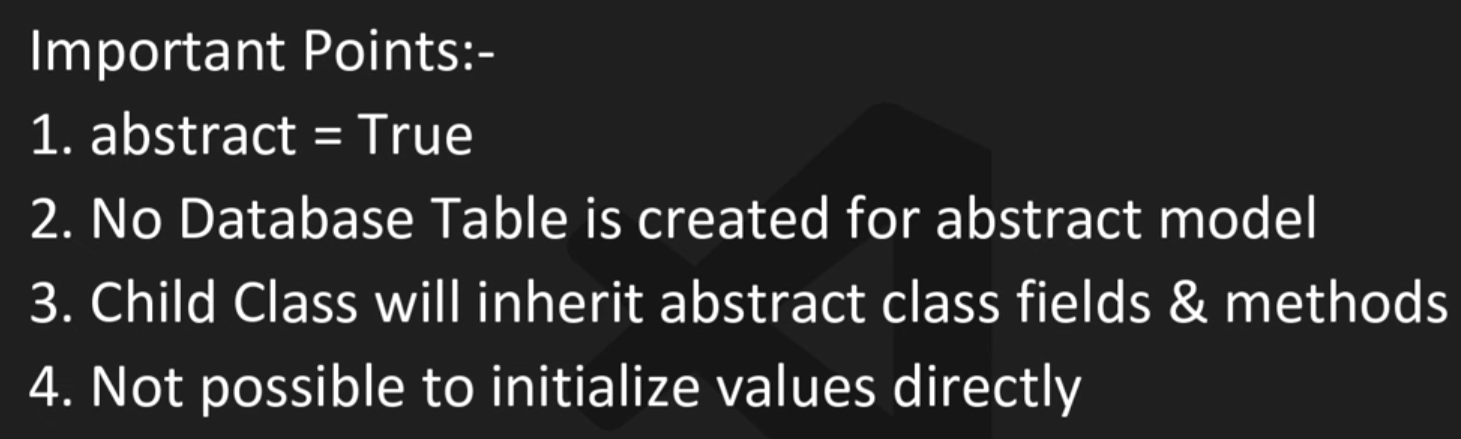
**Videofile:72**

**Codefile:ch62**

**Model Inheritance Abstract Model Multi Table Inheritance and Proxy Model in Django 5**

## **Model Inheritance in Django**



Django allows you to reuse and organize models using inheritance. There are **3 main types**:

### 1. **Abstract Base Classes**

* Used when you want to put **common fields** in one base class, but don’t want a separate table for it.
* Only the child models create database tables.
* **Use case:** When multiple models need same fields (like created\_at, updated\_at).

## **2. Multi-Table Inheritance**

* Each model (parent + child) gets its **own database table**.
* Django automatically links them with a **One-to-One relation**.
* **Use case:** When you want to extend a model with extra fields but still keep parent’s table.

### **3. Proxy Models**

* No new table is created.
* Lets you **change behavior** (methods, managers, ordering) of an existing model.
* **Use case:** When you just want to add custom behavior but don’t need new fields.

✅ **Summary for Quick Notes:**

* **Abstract:** Share fields, no parent table.
* **Multi-table:** New table for child, linked to parent.
* **Proxy:** No new table, only change behavior.

Self documents:

## **Model Inheritance in Django**

Views.py:

from django.db import models

# ---------- Abstract Base Class ----------

# BaseModel is marked abstract, so Django will NOT create a table for it.

# Instead, its fields will be inherited by all child models.

class BaseModel(models.Model):

    name = models.CharField(max\_length=50)

    age = models.IntegerField()

    join\_date = models.DateField()

    class Meta:

        abstract = True   # makes this model abstract

# ---------- Child Models ----------

# Student inherits from BaseModel, so it gets name, age fields.

# 'fees' is added only for Student.

# 'join\_date = None' removes the inherited join\_date field for Student.

class Student(BaseModel):

    fees = models.IntegerField()

    join\_date = None   # This removes join\_date field from Student

# Teacher also inherits from BaseModel and adds salary field.

# Teacher will have name, age, join\_date, salary.

class Teacher(BaseModel):

    salary = models.IntegerField()

# Contractor inherits BaseModel too but overrides join\_date with DateTimeField.

# This means Contractor will have name, age, payment, and a NEW join\_date (datetime).

class Contractor(BaseModel):

    payment = models.IntegerField()

    join\_date = models.DateTimeField()   # overrides parent field

🔑 Key Points to Note:

BaseModel is abstract, so no table is created for it.

Student, Teacher, Contractor each get their own separate table with inherited fields.

Student removes a field (join\_date).

Contractor overrides a field (join\_date).

⚡ This is Abstract Model Inheritance → only child models have tables, parent (abstract) doesn’t.

## **2. Multi-Table Inheritance**

IN THIS IF WE INSERT DATA IN CANDIATE CLASS SO DATA IS AUTOMATELY ADD INTO THE EXAM CNETER BECAUSE ONE TO ONE RELATIONSHIP IN THAT

AND ALSO IF WE DELETE DATA FROM CANDIATE IT DELET’S FROM EXAM CENTER

THIS IS USED WHEN WE MAKE OUR DATA SEPARATE FROM EACH TABLE

VIEWS.PY:

# -----------------------------

# 2. MULTI-TABLE INHERITANCE

# -----------------------------

# Parent table (creates its own DB table)

class ExamCenter(models.Model):

    center\_name = models.CharField(max\_length=255)

    center\_city = models.CharField(max\_length=255)

# Child model (creates separate table + OneToOne link to parent table)

class Candidate(ExamCenter):

    name = models.CharField(max\_length=255)

    roll = models.IntegerField()

### **3. Proxy Models**

In this ordering start’s the reverse

# -----------------------------

# Proxy Model

# -----------------------------

class Product(models.Model):

    name = models.CharField(max\_length=255)

    price = models.IntegerField()

    stock = models.IntegerField()

    def \_\_str\_\_(self):

        return self.name

# Proxy model: No new table created, it reuses Product table

class DiscountProduct(Product):

    class Meta:

        proxy = True      # tells Django not to create a new table

        ordering = ['id'] # changes default behavior (sort by id)