**Videofile:67**

**Codefile:57**

**Working with Database Not Returns Queryset**

## **Working with Database – Not Returning QuerySet**

It shows only the unique data other wise it shows exception

In Django ORM, most database operations return a **QuerySet** (a collection of objects).  
But in some cases, methods return **a single object or a non-QuerySet value**.

### ✅ Key Points:

* These methods **do not return QuerySets**.
* If multiple records are found where only one is expected → Django raises an **exception**.
* Used when you specifically need **one object or one value**, not a list.

### 🔹 Common Examples

from school.models import Student

# Returns a single object (or raises DoesNotExist if no match)

student = Student.objects.get(id=1)

# Returns the first object from QuerySet (or None if not found)

student = Student.objects.first()

# Returns the last object from QuerySet (or None if not found)

student = Student.objects.last()

# Returns only one specific field value (not a QuerySet)

name = Student.objects.values\_list('name', flat=True).get(id=1)

# Returns the number of records (integer, not a QuerySet)

count = Student.objects.count()

# Returns True/False (not a QuerySet)

exists = Student.objects.filter(city="hasilpur").exists()

### 🔹 Exceptions

* get() raises **DoesNotExist** if no record is found.
* get() raises **MultipleObjectsReturned** if more than one record matches.

👉 **In short:**  
Methods like .get(), .first(), .last(), .count(), .exists() **do not return a QuerySet**.  
They return a **single object or value**, and may raise exceptions if the result is not exactly one.

Our own code:

In this in dataset we choose only colum that can unique means that cannot shows different data

Complete views.py:

from django.shortcuts import render

from school.models import Student

def home(request):

    # ---------------------------

    # Single Object Retrieval

    # ---------------------------

    # Get single object by primary key (pk) → raises DoesNotExist if not found

    student\_data = Student.objects.get(pk=1)

    # First object from all records (or None if no record exists)

    student\_data = Student.objects.first()

    # First object from ordered QuerySet

    student\_data = Student.objects.order\_by('name').first()

    # Last object from all records

    student\_data = Student.objects.last()

    # Last object from ordered QuerySet

    student\_data = Student.objects.order\_by('name').last()

    # Latest object based on given field (highest value in that field)

    student\_data = Student.objects.latest('pass\_date')

    # Can also pass multiple fields

    student\_data = Student.objects.latest('pass\_date', 'id')

    # Earliest object based on given field (lowest value in that field)

    student\_data = Student.objects.earliest('pass\_date')

    # ---------------------------

    # Checking Existence

    # ---------------------------

    # Check if any data exists in the table

    student\_data = Student.objects.all()

    print(student\_data.exists())

    # Check if a specific record exists (returns True/False)

    one\_data = Student.objects.get(pk=2)

    print(student\_data.filter(pk=one\_data.pk).exists())

    # ---------------------------

    # Creating Data

    # ---------------------------

    # Create a new record (returns the created object)

    student\_data = Student.objects.create(

        name='Sameer', roll=114, city='Bokaro', marks=60, pass\_date='2020-5-4'

    )

    # Get or create → returns tuple (object, created)

    # If record exists, returns it. If not, creates a new one.

    student\_data, created = Student.objects.get\_or\_create(

        name='Anisa', roll=115, city='Bokaro', marks=60, pass\_date='2020-5-4'

    )

    print(created)  # True if created, False if already existed

    # ---------------------------

    # Updating Data

    # ---------------------------

    # Update specific record

    student\_data = Student.objects.filter(id=4).update(name='Kabir', marks=80)

    # Update multiple records

    student\_data = Student.objects.filter(marks=100).update(city='Delhi')

    # Update or create → if record exists, update it; otherwise create it

    student\_data, created = Student.objects.update\_or\_create(

        id=10, name='Anisa',

        defaults={'name':'Kohli', 'roll':156}

    )

    student\_data, created = Student.objects.update\_or\_create(

        id=12, name='Anisa',

        defaults={'name':'Rohit', 'roll':177, 'marks':200, 'pass\_date':'2020-5-4'}

    )

    # ---------------------------

    # Bulk Create and Update

    # ---------------------------

    # Bulk create → insert multiple records at once

    objs = [

        Student(name='Atif', roll=116, city='Dhanbad', marks=70, pass\_date='2020-5-4'),

        Student(name='Sahil', roll=117, city='Bokaro', marks=50, pass\_date='2020-5-6'),

        Student(name='Kumar', roll=118, city='Dhanbad', marks=30, pass\_date='2020-5-9'),

    ]

    student\_data = Student.objects.bulk\_create(objs)

    # Bulk update → update multiple objects at once

    all\_student\_data = Student.objects.all()

    for stu in all\_student\_data:

        stu.city = 'Mumbai'

    student\_data = Student.objects.bulk\_update(all\_student\_data, ['city'])

    # ---------------------------

    # In Bulk Retrieval

    # ---------------------------

    # Get multiple objects in a dictionary format with pk as key

    student\_data = Student.objects.in\_bulk([1, 3])

    print(student\_data[3].name)

    # Empty list → returns empty dict

    student\_data = Student.objects.in\_bulk([])

    # No argument → returns all records in dict form

    student\_data = Student.objects.in\_bulk()

    # ---------------------------

    # Deleting Data

    # ---------------------------

    # Delete single object by pk

    student\_data = Student.objects.get(pk=2).delete()

    # Delete multiple objects

    student\_data = Student.objects.filter(marks=60).delete()

    # Delete all objects

    student\_data = Student.objects.all().delete()

    # ---------------------------

    # Aggregations

    # ---------------------------

    # Count total number of records (returns integer, not QuerySet)

    print(Student.objects.all().count())

    print("student\_data:", student\_data)

    return render(request, 'school/home.html', {'student': student\_data})