EXERCISES QUE TRABAJAREMOS EN EL CUE

- **EXERCISE 1: CONSULTA DE DATOS**
- **EXERCISE 2: OBTENER LOS DATOS**
- **EXERCISE 3: OBTENER LOS VALORES DE FILAS ESPECÍFICAS**
- **EXERCISE 4: ORDENAMIENTO EN DJANGO QUERYSET**

EXERCISE 1: CONSULTA DE DATOS

En el siguiente ejemplo, usamos el método `.all()` para obtener todos los registros y campos del modelo `Automotriz` y `Modelos` de carros.

Partimos de que tenemos el siguiente modelo:

```python
from django.db import models
import datetime

annos_choices = []
for r in range(1950, (datetime.datetime.now().year+1)):
    annos_choices.append((r,r))


def anno_actual():
    return datetime.date.today().year

class Automotriz(models.Model):
    nombre = models.CharField(max_length=255)
    pais = models.CharField(max_length=255)

    def __str__(self):
        return self.nombre

class ModeloCarro(models.Model):
    nombre = models.CharField(max_length=255)
    automotriz = models.ForeignKey(Automotriz,
    on_delete=models.SET_NULL, blank=True, null=True)
    f_fabricacion = models.IntegerField(choices=annos_choices,
    default=anno_actual)
    costo_fabricacion = models.DecimalField(null=False,
    decimal_places=2, max_digits=10)
    precio_venta = models.DecimalField(null=False, decimal_places=2,
    max_digits=10)

    def __str__(self):
        return self.nombre
```

Nos conectamos a la consola de Python:

```
$ python manage.py shell
```

Importamos el modelo:

```
>>> from relaciones.models import Automotriz, ModeloCarro
```

El objeto se coloca en una variable llamada automotrices y `modelos_carros`, de la siguiente manera:

```
>>> automotrices = Automotriz.objects.all()

>>> modelos_carros = ModeloCarro.objects.all()
```

Visualizamos todos los objetos automotrices:

```
>>> automotrices
```

Salida:

```
<QuerySet [<Automotriz: Toyota>, <Automotriz: Fiat>, <Automotriz: Ford>,
<Automotriz: Chevrolet>]>
```

Visualizamos todos los objetos de `modelos_carros`:

```
>>> modelos_carros
```

Salida:

```
<QuerySet [<ModeloCarro: Fiat Uno>, <ModeloCarro: EcoSport>,
<ModeloCarro: F-150>, <ModeloCarro: Super Duty>, <ModeloCarro: Captiva>,
<ModeloCarro: Cruze>, <ModeloCarro: Toyota Corolla>, <ModeloCarro:
Toyota Fortuner>]>
```

EXERCISE 2: OBTENER LOS DATOS

Existen diferentes métodos para obtener datos de un modelo en un `QuerySet`.

El método `.values()` devuelve cada objeto como un diccionario de Python, con los nombres y valores en pares de clave/valor

Para automotrices:

```
automotrices.values()
```

Salida:

```
<QuerySet [{'id': 14, 'nombre': 'Toyota', 'pais': 'Japón'}, {'id': 11,
'nombre': 'Fiat', 'pais': 'Italia'}, {'id': 13, 'nombre': 'Ford',
'pais': 'Estados Unidos'}, {'id': 12, 'nombre': 'Chevrolet', 'pais':
'Canada'}]>
```

Luego:

```
automotrices.values()
```

Salida:

```
<QuerySet [{'id': 14, 'nombre': 'Toyota', 'pais': 'Japón'}, {'id': 11,
'nombre': 'Fiat', 'pais': 'Italia'}, {'id': 13, 'nombre': 'Ford',
'pais': 'Estados Unidos'}, {'id': 12, 'nombre': 'Chevrolet', 'pais':
'Canada'}]>
```

Para modelos de carros:

```
>>> modelos_carros.values()
```

Salida:

```
<QuerySet [{'id': 1, 'nombre': 'Fiat Uno', 'automotriz_id': 11,
'f_fabricacion': 2003, 'costo_fabricacion': Decimal('950.00'),
'precio_venta': Decimal('1200.00')}, {'id': 2, 'nombre': 'EcoSport',
'automotriz_id': 13, 'f_fabricacion': 2007, 'costo_fabricacion':
Decimal('2000.00'), 'precio_venta': Decimal('2500.00')}, {'id': 3,
'nombre': 'F-150', 'automotriz_id': 13, 'f_fabricacion': 2019,
'costo_fabricacion': Decimal('4500.00'), 'precio_venta':
```

```
Decimal('5500.00')}, {'id': 4, 'nombre': 'Super Duty', 'automotriz_id':
13, 'f_fabricacion': 2022, 'costo_fabricacion': Decimal('6500.00'),
'precio_venta': Decimal('8300.00')}, {'id': 5, 'nombre': 'Captiva',
'automotriz_id': 12, 'f_fabricacion': 2020, 'costo_fabricacion':
Decimal('20000.00'), 'precio_venta': Decimal('22000.00')}, {'id': 6,
'nombre': 'Cruze', 'automotriz_id': None, 'f_fabricacion': 2022,
'costo_fabricacion': Decimal('15000.00'), 'precio_venta':
Decimal('18000.00')}, {'id': 7, 'nombre': 'Toyota Corolla',
'automotriz_id': 14, 'f_fabricacion': 2020, 'costo_fabricacion':
Decimal('14000.00'), 'precio_venta': Decimal('18000.00')}, {'id': 8,
'nombre': 'Toyota Fortuner', 'automotriz_id': 14, 'f_fabricacion':
2020, 'costo_fabricacion': Decimal('25000.00'), 'precio_venta':
Decimal('32000.00')}]>
```

Obtener los datos de una columna específica con el método `.values_list()`:

```
>>> modelos_carros.values_list('nombre', 'f_fabricacion')
```

Salida:

```
<QuerySet [('Fiat Uno', 2003), ('EcoSport', 2007), ('F-150', 2019),
('Super Duty', 2022), ('Captiva', 2020), ('Cruze', 2022), ('Toyota
Corolla', 2020), ('Toyota Fortuner', 2020)]>
```

EXERCISE 3: OBTENER LOS VALORES DE FILAS ESPECÍFICAS

Puede filtrar la búsqueda para sólo un determinado registro específico, utilizando el método `filter()`:

```
>>> modelos_carros.filter(nombre='EcoSport').values()
```

Salida:

```
<QuerySet [{'id': 2, 'nombre': 'EcoSport', 'automotriz_id': 13,
'f_fabricacion': 2007, 'costo_fabricacion': Decimal('2000.00'),
'precio_venta': Decimal('2500.00')}]>
```

El método `filter()` se usa para filtrar su búsqueda, y le permite devolver sólo las filas que coinciden con el término de búsqueda. Éste toma los argumentos como `**kwargs` (argumentos de palabras clave), por lo que puede filtrar en más de un campo separándolos con una coma.

Ejemplo del condicional `AND` o (Y):

```
>>> modelos_carros.filter(nombre='EcoSport', id=2).values()
```

Salida:

```
<QuerySet [{'id': 2, 'nombre': 'EcoSport', 'automotriz_id': 13,
'f_fabricacion': 2007, 'costo_fabricacion': Decimal('2000.00'),
'precio_venta': Decimal('2500.00')}]>
```

Consulta generada en SQL:

```
>>> consulta_and = modelos_carros.filter(nombre='EcoSport',
id=2).values()
>>> str(consulta_and.query)
```

Salida:

```
'SELECT "relaciones_modelocarro"."id",
"relaciones_modelocarro"."nombre",
"relaciones_modelocarro"."automotriz_id",
"relaciones_modelocarro"."f_fabricacion",
"relaciones_modelocarro"."costo_fabricacion",
"relaciones_modelocarro"."precio_venta" FROM "relaciones_modelocarro"
WHERE ("relaciones_modelocarro"."id" = 2 AND
"relaciones_modelocarro"."nombre" = EcoSport)'
```

Ejemplo del condicional OR o (O):

```
>>> consulta_or = modelos_carros.filter(nombre='EcoSport').values() |
modelos_carros.filter(nombre='Captiva').values()

>>> consulta_or
```

Salida:

```
<QuerySet [{'id': 2, 'nombre': 'EcoSport', 'automotriz_id': 13,
'f_fabricacion': 2007, 'costo_fabricacion': Decimal('2000.00'),
'precio_venta': Decimal('2500.00')}, {'id': 5, 'nombre': 'Captiva',
'automotriz_id': 12, 'f_fabricacion': 2020, 'costo_fabricacion':
Decimal('20000.00'), 'precio_venta': Decimal('22000.00')}]>
```

Consulta generada en SQL:

```
>>> str(consulta_or.query)
```

Salida:

```
'SELECT "relaciones_modelocarro"."id",
"relaciones_modelocarro"."nombre",
"relaciones_modelocarro"."automotriz_id",
"relaciones_modelocarro"."f_fabricacion",
"relaciones_modelocarro"."costo_fabricacion",
"relaciones_modelocarro"."precio_venta" FROM "relaciones_modelocarro"
WHERE ("relaciones_modelocarro"."nombre" = EcoSport OR
"relaciones_modelocarro"."nombre" = Captiva)'
```

SINTAXIS DE BÚSQUEDAS DE CAMPO

Todas las palabras clave de búsqueda de campo deben especificarse con el nombre del campo, seguido de dos (__) caracteres de subrayado, y la palabra clave.

```
>>> consulta_campo =
modelos_carros.filter(nombre__startswith='E').values()

>>> consulta_campo
```

Salida:

```
<QuerySet [{'id': 2, 'nombre': 'EcoSport', 'automotriz_id': 13,
'f_fabricacion': 2007, 'costo_fabricacion': Decimal('2000.00'),
'precio_venta': Decimal('2500.00')}]>
```

Consulta generada en SQL:

```
>>> str(consulta_or.query)
```

Salida:

```
'SELECT "relaciones_modelocarro"."id",
"relaciones_modelocarro"."nombre",
"relaciones_modelocarro"."automotriz_id",
"relaciones_modelocarro"."f_fabricacion",
"relaciones_modelocarro"."costo_fabricacion",
"relaciones_modelocarro"."precio_venta" FROM "relaciones_modelocarro"
WHERE "relaciones_modelocarro"."nombre"::text LIKE E%'
```

Consulta de un campo que contiene alguna palabra:

```
>>> consulta_campo =
modelos_carros.filter(nombre__contains='e').values('nombre')

>>> consulta_campo
```

Salida:

```
<QuerySet [{'nombre': 'Super Duty'}, {'nombre': 'Cruze'}, {'nombre':
'Toyota Fortuner'}]>
```

Consulta generada en SQL:

```
>>> str(consulta_or.query)
```

Salida:

```
'SELECT "relaciones_modelocarro"."nombre" FROM "relaciones_modelocarro"
WHERE "relaciones_modelocarro"."nombre"::text LIKE %e%'
```

Puedes ver una referencia aquí.

EXERCISE 4: ORDENAMIENTO EN DJANGO QUERYSET

Ascendente:

```
>>> consulta_ordenamiento =
modelos_carros.all().order_by('nombre').values('nombre')

>>> consulta_ordenamiento
```

Salida:

```
<QuerySet [{'nombre': 'Captiva'}, {'nombre': 'Cruze'}, {'nombre':
'EcoSport'}, {'nombre': 'F-150'}, {'nombre': 'Fiat Uno'}, {'nombre':
'Super Duty'}, {'nombre': 'Toyota Corolla'}, {'nombre': 'Toyota
Fortuner'}]>
```

Consulta generada en SQL:

```
>>> str(consulta_ordenamiento.query)
```

Salida:

```
'SELECT "relaciones_modelocarro"."nombre" FROM "relaciones_modelocarro"
ORDER BY "relaciones_modelocarro"."nombre" ASC'
```

Descendente:

```
>>> consulta_ordenamiento = modelos_carros.all().order_by('-
nombre').values('nombre')

>>> consulta_ordenamiento
```

Salida:

```
<QuerySet [{'nombre': 'Toyota Fortuner'}, {'nombre': 'Toyota Corolla'},
{'nombre': 'Super Duty'}, {'nombre': 'Fiat Uno'}, {'nombre': 'F-150'},
{'nombre': 'EcoSport'}, {'nombre': 'Cruze'}, {'nombre': 'Captiva'}]>
```

Consulta generada en SQL:

```
>>> str(consulta_ordenamiento.query)
```

Salida:

```
'SELECT "relaciones_modelocarro"."nombre" FROM "relaciones_modelocarro"
ORDER BY "relaciones_modelocarro"."nombre" DESC'
```

LLAVES FORÁNEAS

Cuando se requiere cargar información donde se tiene relación unos a muchos, podemos utilizar `select_related` para mejorar el rendimiento.

```
>>> c_foranea = ModeloCarro.objects.all().select_related('automotriz')

>>> c_foranea
```

Salida:

```
<QuerySet [<ModeloCarro: Fiat Uno>, <ModeloCarro: EcoSport>,
<ModeloCarro: F-150>, <ModeloCarro: Super Duty>, <ModeloCarro: Captiva>,
<ModeloCarro: Cruze>, <ModeloCarro: Toyota Corolla>, <ModeloCarro:
Toyota Fortuner>]>
```

```
>>>for foranea in c_foranea:
...     print(foranea.automotriz)
...
```

Salida:

```
Fiat
Ford
Ford
Ford
Chevrolet
None
Toyota
Toyota
```

Consulta generada en SQL:

```
>>> str(c_foranea.query)
```

Salida:

```
'SELECT "relaciones_modelocarro"."id",
"relaciones_modelocarro"."nombre",
"relaciones_modelocarro"."automotriz_id",
"relaciones_modelocarro"."f_fabricacion",
"relaciones_modelocarro"."costo_fabricacion",
"relaciones_modelocarro"."precio_venta", "relaciones_automotriz"."id",
"relaciones_automotriz"."nombre", "relaciones_automotriz"."pais" FROM
"relaciones_modelocarro" LEFT OUTER JOIN "relaciones_automotriz" ON
("relaciones_modelocarro"."automotriz_id" =
"relaciones_automotriz"."id")'
```

Otra manera es:

```
>>> c_foranea = ModeloCarro.objects.all().values('nombre',
'automotriz__nombre')
```

Salida:

```
<QuerySet [{'nombre': 'Fiat Uno', 'automotriz__nombre': 'Fiat'},
{'nombre': 'EcoSport', 'automotriz__nombre': 'Ford'}, {'nombre': 'F-
150', 'automotriz__nombre': 'Ford'}, {'nombre': 'Super Duty',
'automotriz__nombre': 'Ford'}, {'nombre': 'Captiva',
'automotriz__nombre': 'Chevrolet'}, {'nombre': 'Cruze',
'automotriz__nombre': None}, {'nombre': 'Toyota Corolla',
'automotriz__nombre': 'Toyota'}, {'nombre': 'Toyota Fortuner',
'automotriz__nombre': 'Toyota'}]>
```

Consulta generada en SQL:

```
>>> str(c_foranea.query)
```

Salida:

```
'SELECT "relaciones_modelocarro"."nombre",
"relaciones_automotriz"."nombre" FROM "relaciones_modelocarro" LEFT
OUTER JOIN "relaciones_automotriz" ON
("relaciones_modelocarro"."automotriz_id" =
"relaciones_automotriz"."id")'
```