Lesson on Django REST Framework Query Filters

This lesson explains how to use query filters in Django REST Framework (DRF) when working with ModelViewSets. We'll cover simple filters, advanced filtering with FilterSets, searching, and ordering.

1. The Basics: Filtering with Query Params

By default, a ModelViewSet returns all records. To allow filtering, you need to enable query filters.

Example request (before filters):
GET /products/

Steps to enable filters:

1. Install django-filter:
   pip install django-filter

2. Add 'django_filters' to INSTALLED_APPS in settings.py.

3. Add default filter backend in settings.py:

```
REST_FRAMEWORK = {
   'DEFAULT_FILTER_BACKENDS': ['django_filters.rest_framework.DjangoFilterBackend']
}
```

2. Simple Field Filters

You can quickly enable filtering on model fields by specifying filterset_fields:

```
class ProductViewset(viewsets.ModelViewSet):
   queryset = Product.objects.all()
   serializer_class = ProductSerializer
   filter_backends = [DjangoFilterBackend]
   filterset_fields = ['category', 'price', 'in_stock']
```

Example requests:
GET /products/?category=Electronics
GET /products/?price=100
GET /products/?in_stock=True

3. Advanced Filtering with FilterSet

For more complex queries (e.g., price ranges), define a custom FilterSet:

```
class ProductFilter(django_filters.FilterSet):
  min_price = django_filters.NumberFilter(field_name='price', lookup_expr='gte')
  max_price = django_filters.NumberFilter(field_name='price', lookup_expr='lte')

  class Meta:
    model = Product
    fields = ['category', 'in_stock', 'min_price', 'max_price']
```

Then use it in your viewset:

```
filterset_class = ProductFilter
```

Example requests:

```
GET /products/?min_price=50&max_price=200
GET /products/?category=Books&in_stock=True
```

4. Searching

Enable keyword search across fields using SearchFilter:

```
filter_backends = [DjangoFilterBackend, SearchFilter]
search_fields = ['name', 'description']
```

Example:

```
GET /products/?search=laptop
```

5. Ordering

Enable sorting of query results with OrderingFilter:

```
filter_backends = [DjangoFilterBackend, SearchFilter, OrderingFilter]
ordering_fields = ['price', 'created_at']
ordering = ['-created_at']  # default ordering
```

Example:

```
GET /products/?ordering=price
GET /products/?ordering=-price
```

6. Combining Everything

Filters, searching, and ordering can be combined in a single query:

```
GET /products/?search=phone&min_price=200&max_price=800&ordering=-price
```

This query:
- Searches for 'phone'
- Filters results with price between 200–800
- Orders results by descending price

Summary

✔ Use filterset_fields for quick filters.
✔ Use a custom FilterSet for advanced filtering.
✔ Use SearchFilter for keyword searches.
✔ Use OrderingFilter for sorting results.