**Theory**

**What is Serialization?**

**Serialization** in the context of Django REST Framework (DRF) refers to the process of converting complex data types, such as Django QuerySets or model instances, into a format that can be easily rendered into JSON, XML, or other content types. This is essential for sending data from your Django application to the client, typically in the form of a JSON response.

Serialization is a way of transforming data that might come from the database (like Django model instances or QuerySets) into a format that can be sent over HTTP (e.g., JSON), or vice versa (deserialization), where you take incoming JSON data and convert it into Python objects to be processed.

**Converting Django QuerySets to JSON**

Django ORM (Object-Relational Mapping) provides a way to interact with your database using Python objects. When you retrieve data from the database using Django's QuerySet, the data isn't in a format that can be directly sent as a response. This is where serialization comes into play.

For example, suppose you have a Doctor model in Django, and you want to return a list of doctors in JSON format. You need to serialize the QuerySet to convert it into JSON format.

**Using Serializers in Django REST Framework (DRF)**

Django REST Framework (DRF) provides a powerful and easy-to-use mechanism for serialization using the serializers.ModelSerializer class. This serializer class converts model instances to JSON (or other formats) and can handle data validation and conversion.

To serialize a model in DRF, you:

1. Create a serializer class that inherits from serializers.ModelSerializer.
2. Specify the model to be serialized and the fields to be included in the output.
3. Use the serializer in your view to convert the model data into JSON format.