An intermediate Django interview typically covers key concepts beyond the basics, focusing on real-world scenarios and in-depth knowledge of Django's features. Here’s a summary of the key topics that could be covered:

1. **Django Models & ORM:**
   - Explain Django’s ORM and how to use models to interact with the database.
   - Discuss how relationships like `OneToOne`, `ForeignKey`, and `ManyToMany` work.
   - Explain model methods, custom querysets, and using raw SQL with Django ORM.
   - Discuss migrations, including how to handle database schema changes in production.

2. **Views & URL Routing:**
   - Difference between function-based views (FBVs) and class-based views (CBVs).
   - Understanding generic views like `ListView`, `DetailView`, `CreateView`, etc.
   - Explain how Django's URL dispatcher works and how to use URL patterns effectively.

3. **Forms & Validation:**
   - Handling forms in Django, including `ModelForm`, form validation, and custom form fields.
   - Discuss handling form submissions, CSRF protection, and AJAX form handling.

4. **Authentication & Authorization:**
   - Explain Django’s built-in authentication system, user login, logout, and password management.
   - Working with `User` models, permissions, groups, and custom user models.
   - How to implement role-based access control and manage permissions.

5. **Middlewares:**
   - Discuss the role of middleware in Django’s request/response cycle.
   - Examples of when to use custom middleware and how to write it.

6. **Django REST Framework (DRF):**
   - Basics of Django REST framework, including serializers, viewsets, routers, and how to create APIs.
   - Authentication in DRF (token-based, session-based, OAuth).
   - How to handle pagination, filtering, and versioning in APIs.

7. **Static & Media Files:**
   - How Django handles static files and media files in development and production.
   - Use of `django-storages` and other tools for managing media files in cloud services like AWS S3.

8. **Deployment & Performance:**
   - Discuss deployment strategies (e.g., using WSGI, ASGI, Gunicorn, Nginx).
   - Common performance optimization techniques, including caching (Memcached, Redis), database query optimization, and static files handling (via CDN).
   - How to set up Django for handling heavy traffic and scaling.

9. **Testing:**

  - Explain unit testing in Django, using `TestCase` and `Client`.
  - Discuss the importance of writing tests for models, views, and APIs.
  - How to use Django's built-in testing tools and coverage reports.

10. **Security:**
   - Common Django security features (CSRF protection, SQL injection prevention, XSS prevention).
   - How to secure sensitive data, manage sessions, and implement secure authentication.

11. **Advanced Topics:**
   - Signals and how to use them for decoupling components.
   - Asynchronous views in Django (ASGI) and when to use them.
   - Working with Celery for task queues and background jobs.

These topics will help gauge the candidate's proficiency in Django and their ability to handle real-world scenarios in a professional setting.