/ Review the essential list of subjects and classes relevant to Backend Django Development. You must finish these assignments. Below, you'll find crucial links that can assist you in comprehending the fundamental principles necessary to complete the tasks.

**Please keep in mind that you should set up Django with either a MySQL or PostgreSQL database.**

## Exercise 1 (Python Types & Basics of Django ORM):

The objective of this exercise is to enhance your conceptual grasp of Django Models, relationships, and querying. Ensure that you incorporate type annotations in all your Python code.

1. Learn Type hints in Python, TypeDict implementation for class, dictionary, List, primitives and functions (pyright extension)
2. Create 3 Models: **Country, State, City** along with their relations.
    1. Country must include fields like ***id (UUID Field), name, country_code, curr_symbol, phone_code, etc***
    2. State must include fields like ***id (UUID Field), name, state_code, gst_code, etc.***
    3. City must include fields like ***id (UUID Field), name, city_code, phone_code, population, avg_age, num_of_adult_males, num_of_adult_females etc.***
3. Write a Query to Insert data in Country, State and City
4. Write a Query to Bulk Insert Data in Country, State and City-
5. Write a Query to Bulk Update Data in Country, State and City***
6. Write a Query to fetch all Countries, States, and Cities
7. Write a Query to fetch all cities of a State
8. Write a Query to fetch all states of a Country
9. Write a Query to fetch all Cities of a Country name
10. Write a Query to fetch a City of a Country with Minim um and Maximum Population

### Resources:

- Fundamentals of Python
- Python OOP
- SQL in Python (PostgreSQL)
- Git
- Fundamentals of Django
- Django Model Relationship
    - (Documentation) https://docs.djangoproject.com/en/3.1/topics/db/examples/
- Django ORM Querying
    - (Documentation) https://docs.djangoproject.com/en/3.1/ref/models/querysets/
    - (Practice Material) https://docs.djangoproject.com/en/3.1/topics/db/queries/
- Django Expressions (Subqueries, Aggregates, Window Functions)
    - (Documentation) https://docs.djangoproject.com/en/3.1/ref/models/expressions/
- SQSum (Practice) https://stackoverflow.com/questions/61387771/calculate-percentage-in-django-database-query

## Exercise 2 (Custom Migration):

In this exercise, you'll work on customizing the database schema of a Django project by creating a custom user model and implementing a custom migration. Here are the steps involved:

11. Create a Custom User Model (Email as username field)
12. Add my_user ForeignKeyField in Country Model to save each user's data independently (add field with null=True).
13. Write Custom Migration to populate a random user to each country.**
14. Remove null=True attribute from User ForeignKey in Country model and migrate

### Resources:

- Migrations | Django documentation | Django (djangoproject.com)
- Migration Operations | Django documentation | Django (djangoproject.com)

## Exercise 3 (Django Rest Framework, Authentications & Debugging):

The purpose of Exercise 3 is to develop a solid understanding of REST API endpoints. These endpoints serve as the interfaces through which external systems interact with your backend. Think of them as functions that take input and produce output, typically in the form of JSON data. API endpoints can incorporate validation and security measures to ensure that the input comes from an authorized source.

When building an application, you'll have various entities, each responsible for operations like creating, updating, retrieving, deleting, and listing data. With Django Rest Framework (DRF), you can efficiently construct these endpoints following established conventions. Advanced endpoints can even handle complex input data in the form of nested JSON structures.

15. Write class-based generic views (CRUD) API endpoints for Users (Use Cursor pagination in List Endpoint)
16. Write Api endpoints for Signin and Signout using Token Authentication
    4. User List, Retrieve, Update and Destroy must have isAuthenticated Permission
    5. User Create will have Allow Any Permissions
17. Write class based generic views (CRUD) api endpoints for Country, State and City using DRF (Note these endpoints must be only used by Authenticated User, add Permission Classes, Each Entity should be stored for their respective User. Data of another user must not be accessed or written by any other user)
    ***6.*** State: Add SerializerMethodField ***my_country__name***, and ***my_country__my_user__name***
    ***7.*** City: Add SerializerMethodField ***my_state__name***
    8. Country Code must be unique
    9. Country's and City's Phone Code must be unique
    10. Gst Code must be unique
    11. One Country should not have same state name
    12. One State should not have duplicate city name
    13. One State should not have duplicate city_code
    14. City's population must be greater than the summation of num_of_adult males and females.

15. All your code must have comments explaining your code, All classes must be in Pascal Case and methods, variables must be in Snake Case. Line spacing and proper indentation should be used to format your code.
18. Replace Token Authentication with JWT Authentication and reverify all endpoints
19. Write CRUD nested Api endpoints which should allow the client to Create, Retrieve, Update, and List Country-States-Cities in the following manner: (All validations must apply on these endpoints as well)

```
{
    “name”: “India”,
   “country_code”: “IN”,
    …
    “states”: [
        {
            “name”: “Gujarat”,
            ….,
            “cities”: [
                {
                    “name”: “Surat”,
                    ….
                }
            ]
        }
    ]
}
```

20. Use django silk to analyse the number of SQLs being fired on each endpoint. Try to minimize the SQLs using select_related and prefetch_related.

## Resources:

- DRF Best Practices
    - Generic Views https://www.django-rest-framework.org/api-guide/generic-views/
    - Serializers https://www.django-rest-framework.org/api-guide/serializers/
    - Serializer Fields https://www.django-rest-framework.org/api-guide/fields/
    - Serializer Relations https://www.django-rest-framework.org/api-guide/relations/

  - Serializer Validations https://www.django-rest-framework.org/api-guide/validators/
  - Authentication https://www.django-rest-frmework.org/api-guide/authentication/
  - Permissions https://www.django-rest-framework.org/api-guide/permissions/
  - Pagination https://www.django-rest-framework.org/api-guide/pagination/
  - Status Codes https://www.django-rest-framework.org/api-guide/status-codes/
  - Exceptions https://www.django-rest-framework.org/api-guide/exceptions/
  - Testing https://www.django-rest-framework.org/api-guide/testing/
- PDB: https://www.youtube.com/watch?v=ChuU3NlYRLQ (Used it for debugging the code)