Migrations

Learn:

- **What are Migrations?**
- Migrations are a way to manage changes to your database schema over time.
- They allow you to keep track of changes to your models and apply those changes to the database schema.

- **Key Concepts:**
- **Initial Migration:** The first migration created for an app, which defines the initial state of the database schema.
- **Schema Migration:** Subsequent migrations that alter the database schema based on changes to your models.
- **Migration Files:** Python files generated by Django to represent database schema changes.
- **Migration Operations:**

Instructions within migration files that define how to modify the database schema.

- **Migration Workflow:**
1. **Make Model Changes:** Modify your models in models.py to reflect desired schema changes.
2. **Generate Migrations:** Run python manage.py makemigrations to create migration files based on model changes.
3. **Apply Migrations:** Execute python manage.py migrate to apply migration files and update the database schema.
4. **Check Status:** Use python manage.py showmigrations to view the status of migrations.

Practice:

1. **Make Model Changes:**
- Update your models in models.py to add, modify, or

Instructions within migration files that define how to modify the database schema.

- **Migration Workflow:**
1. **Make Model Changes:** Modify your models in models.py to reflect desired schema changes.
2. **Generate Migrations:** Run python manage.py makemigrations to create migration files based on model changes.
3. **Apply Migrations:** Execute python manage.py migrate to apply migration files and update the database schema.
4. **Check Status:** Use python manage.py showmigrations to view the status of migrations.

Practice:

1. **Make Model Changes:**
- Update your models in models.py to add, modify, or

remove fields as needed.
- Ensure that your model changes accurately reflect the desired database schema changes.

2. **Generate Migrations:**
- Run `python manage.py makemigrations` to create migration files based on the model changes.
- Review the generated migration files in the `migrations` directory to verify the proposed schema changes.

3. **Apply Migrations:**
- Execute `python manage.py migrate` to apply migration files and update the database schema.
- Django will apply pending migrations in the correct order to ensure data integrity.

4. **Rollback Migrations:**
- If needed, you can roll back migrations using `python manage.py migrate <app_name> <migration_name>`.

- Be cautious when rolling back migrations, as it may result in data loss or inconsistencies.

Know More:

- **FAQs:**

1. **Can I customize migration files?**
- While Django generates migration files automatically, you can manually edit them to add custom operations or fine-tune the migration process.

2. **How can I apply migrations to a specific app?**
- You can specify the app name when running migrations, e.g., python manage.py migrate myapp .

3. **What if there are conflicts during migrations?**
- Django's migration system attempts to resolve conflicts automatically.

However, if conflicts arise, you may need to resolve them manually by adjusting migration files.

4. ** Is it possible to generate SQL migration files instead of Python files? **

- Yes, you can generate SQL migration files using `python manage.py sqlmigrate`. This command displays the SQL statements that would be executed for a specific migration without applying them.