Hii Sir,   
  
I am Sudha Rajendran,

I attended the first bootcamp session, and it was a great experience. I learned a lot about your company's workflows, which I found very interesting. Your focus on a generative AI platform for creating multilingual and multi-modal content is impressive. You cover many aspects of content, including translation, creation, transformation, and discovery.

I like how your workflows combine AI and human expertise to ensure high-quality results. It’s not just about using AI; you offer many customizable features, like allowing users to add glossaries and personalize translations. The options for choosing words, automatic page setup, and layout formatting are excellent, supporting more than 25 file formats.

The design is truly impressive! The design features are creative and allow users to translate and create stunning designs, such as posters and social media graphics. Additionally, the writing module is especially interesting because it helps users brainstorm ideas, rewrite content, create drafts, and translate them into different languages.

Additionally, your audio workflows enable speech-to-text and text-to-speech conversions. The "Chatbooks" concept is innovative, letting users turn their content into a chat-friendly format. Users can upload documents, whether they are single files or entire catalogs, and interact with the content in any language. For example, a document in Chinese can be discussed in a different language.

Finally, "Ailaysa" is a collaborative platform for project management, managing linguistic assets, team coordination, assignment tracking, and integration. Overall, the session gave me valuable insights into your company's creative solutions.

You taught the basics of the Django framework and guided us through a simple project called "movie\_portal." This helped us understand how to start a project and run it using Django commands. You also emphasized the importance of table relationships, which are crucial for organizing and managing data effectively.

**My work:**  
  
I am working on a backend project called Library Access using the Django framework. Following your instructions, I created the necessary table relationships in the program.  
  
I created total eight tables, Here the table names,

* Author
* Publisher
* Category
* Book
* Member
* BorrowBooks
* Reservation
* Penalty
* Inventory
* Review

Three types of Relationships in table:

* **One-to-one**. When each item in each table only appears once. For example, each employee can have only one company car to use.
* **One-to-many**. When one item in one table can have a relationship to multiple items in another table. For example, each purchase order can include multiple products.
* **Many-to-many**. When one or more items in one table can have a relationship to one or more items in another table. For example, each order can have multiple products, and each product can appear on many orders.

**Relationships in my Django models:**

* **Author and Book:** One-to-Many relationship (one author can write multiple books, but each book has only one author).
* **Publisher and Book:** One-to-Many relationship (one publisher can publish multiple books, but each book is published by only one publisher).
* **Category and Book**: Many-to-Many relationship (one book can belong to multiple categories, and each category can include multiple books).
* **Members and BorrowBooks:** One-to-Many relationship (one member can borrow multiple books, but each borrowing record relates to one specific member).
* **Book and BorrowBooks**: One-to-Many relationship (one book can be borrowed multiple times by different members, but each borrowing entry is linked to only one book).
* **Member and Reservation**: One-to-Many relationship (one member can have multiple reservations, but each reservation is linked to only one member).
* **Book and Reservation**: One-to-Many relationship (one book can be reserved multiple times by different members, but each reservation entry is linked to only one book).
* **Member and Penalty**: One-to-Many relationship (one member can have multiple penalties, but each penalty is linked to only one member).
* **BorrowBooks and Penalty**: One-to-Many relationship (one borrowing entry can have multiple penalties, but each penalty is linked to only one borrowing entry).
* **Book and Inventory**: One-to-Many relationship (one book can have multiple inventory entries, but each inventory entry is linked to only one book).
* **Member and Review**: One-to-Many relationship (one member can write multiple reviews, but each review is linked to only one member).
* **Book and Review**: One-to-Many relationship (one book can have multiple reviews, but each review is linked to only one book).

**CODING:**  
  
**Model.py/library\_access**

from django.db import models

from django.contrib.auth.models import User

class Author(models.Model):

    """

    To Store Information About Authors

    """

    name = models.CharField(max\_length=100)

    bio = models.TextField(blank=True)

    def \_\_str\_\_(self):

        return self.name

class Publisher(models.Model):

    """

    To Store Information About Publishers

    """

    name = models.CharField(max\_length=100)

    address = models.TextField(blank=True)

    def \_\_str\_\_(self):

        return self.name

class Category(models.Model):

    """

    To Store Information About Categories

    """

    name = models.CharField(max\_length=100)

    def \_\_str\_\_(self):

        return self.name

class Book(models.Model):

    """

    To Store Information About Categories

    """

    title = models.CharField(max\_length=200)

    author = models.ForeignKey(Author, related\_name='books', on\_delete=models.CASCADE) # Many to one Relationship

    publisher = models.ForeignKey(Publisher, on\_delete=models.CASCADE)

    isbn = models.CharField(max\_length=15, unique=True)

    publish\_date = models.DateField()

    categories = models.ManyToManyField(Category) # Many to Many Relationships

    def \_\_str\_\_(self):

        return self.title

class Member(models.Model):

    """

    To Store Information About Member of the Library

    """

    first\_name = models.CharField(max\_length=50)

    last\_name = models.CharField(max\_length=50)

    email = models.EmailField(unique=True)

    phone = models.CharField(max\_length=15)

    join\_date = models.DateField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        return f'{self.first\_name} {self.last\_name}'

class BorrowBooks(models.Model):

    """

    To Store Information About Borrowed book details

    """

    book = models.ForeignKey(Book, related\_name='borrows', on\_delete=models.CASCADE) # Many to one Relationship

    member = models.ForeignKey(Member, related\_name='borrows', on\_delete=models.CASCADE) # Many to one Relationship

    loan\_date = models.DateField(auto\_now\_add=True)

    return\_date = models.DateField(null=True, blank=True)

    def \_\_str\_\_(self):

        return f'{self.book.title} borrowed by {self.member.first\_name} {self.member.last\_name}'

class Reservation(models.Model):

    """

    To Store Information About Users Who Wants To Reserve Books That Are Currently Unavailable.

    """

    user = models.ForeignKey(Member, related\_name='reservations', on\_delete=models.CASCADE) # Many to one Relationship

    book = models.ForeignKey(Book, related\_name='reservations', on\_delete=models.CASCADE) # Many to one Relationship

    reservation\_date = models.DateField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        return f"{self.user.username} reserved {self.book.title}"

class Penalty(models.Model):

    """

    To Store Information About Managing Penalties For Late Book Returns

    """

    user = models.ForeignKey(Member, related\_name='penalties', on\_delete=models.CASCADE) # Many to one Relationship

    borrowing = models.ForeignKey(BorrowBooks, related\_name='penalties', on\_delete=models.CASCADE) # Many to one Relationship

    amount = models.DecimalField(max\_digits=6, decimal\_places=2)

    paid = models.BooleanField(default=False)

    def \_\_str\_\_(self):

        return f"Penalty for {self.user.username}"

class Inventory(models.Model):

    """

    To Store Information About tracking the number of available copies of each book.

    """

    book = models.ForeignKey(Book, related\_name='inventory', on\_delete=models.CASCADE) # Many to one Relationship

    quantity = models.PositiveIntegerField()

    def \_\_str\_\_(self):

        return f"{self.book.title} - {self.quantity} available"

class Review(models.Model):

    """

    To store Information About User Reviews and Ratings for Books.

    """

    user = models.ForeignKey(Member, related\_name='reviews', on\_delete=models.CASCADE)  # Many to one relationship

    book = models.ForeignKey(Book, related\_name='reviews', on\_delete=models.CASCADE)  # Many to one relationship

    rating = models.PositiveIntegerField()

    comment = models.TextField()

    created\_at = models.DateTimeField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        return f"Review for {self.book.title} by {self.user.username}"

**admins.py/ library\_access**  
  
 from django.contrib import admin

from library\_manage.models import Author, Publisher, Category, Book, Member, BorrowBooks, Reservation, Penalty, Inventory, Review

# Register your models here.

admin.site.register(Author)

admin.site.register(Publisher)

admin.site.register(Category)

admin.site.register(Book)

admin.site.register(Member)

admin.site.register(BorrowBooks)

admin.site.register(Reservation)

admin.site.register(Penalty)

admin.site.register(Inventory)

admin.site.register(Review)

**OUTPUT:**



