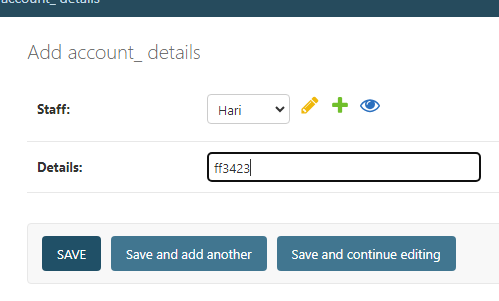
# **Relationships Between Django Models**

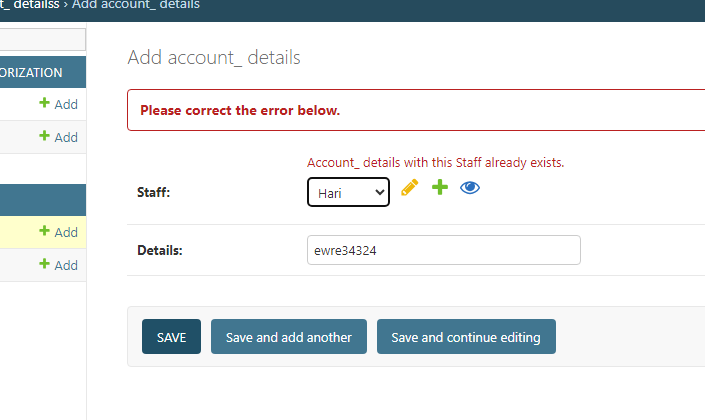
### **OneToOneField (One-to-One Relationship):**

* A OneToOneField is used to create a One-to-One relationship between two models.
* It represents a unique and singular association where each instance of one model is associated with exactly one instance of the other model, and vice versa.

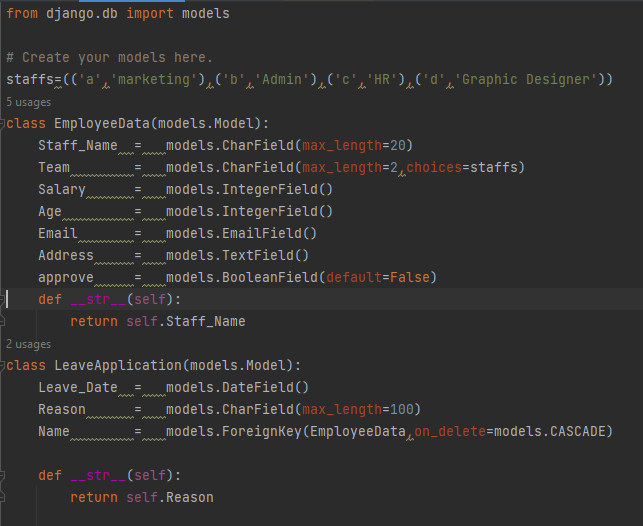


Django application: EmployeeData and Account\_Details, with a one-to-one relationship between them.

if you try to add Account\_Details for the same employee more than once with the same details, Django will raise an integrity error due to the uniqueness constraint on the Details field.



**Many-to-One Relationship:**

****

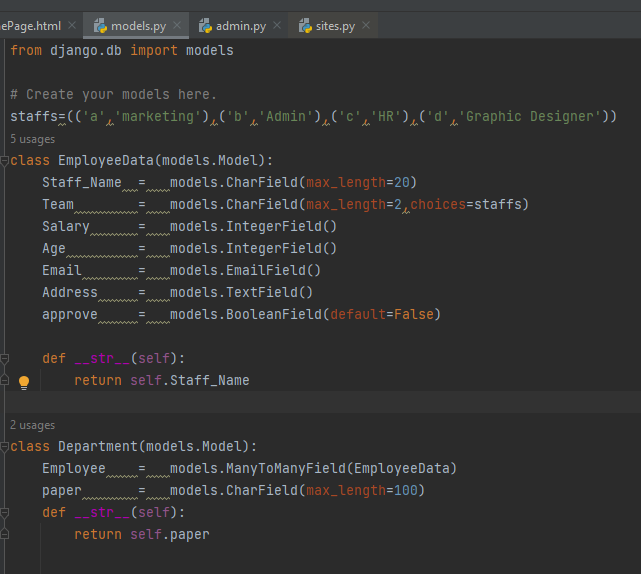
* class LeaveApplication(models.Model):: This line defines a new Django model named LeaveApplication which is a subclass of models.Model. This indicates that it's meant to be a database model.
* Leave\_Date = models.DateField(): This line creates a field named Leave\_Date in the model, which is of type DateField. This field is likely intended to store the date of the leave.
* Reason = models.CharField(max\_length=100): This line creates a field named Reason in the model, which is of type CharField. The max\_length parameter is set to 100, indicating that the maximum length of the text for this field is 100 characters. This field is meant to store the reason for the leave.
* Name = models.ForeignKey(EmployeeData, on\_delete=models.CASCADE): This line creates a ForeignKey field named Name in the model. It establishes a many-to-one relationship with another model named EmployeeData using a foreign key. The on\_delete=models.CASCADE parameter means that if the referenced EmployeeData object is deleted, then also delete the related LeaveApplication objects.
* def \_\_str\_\_(self):: This method is defined to represent a string representation of the model instance. In this case, it returns the value of the Reason field, which means that when you print an instance of this model, it will display the reason for the leave.

So, in summary, this code defines a Django model for leave applications with fields for leave date, reason, and a foreign key relationship with an EmployeeData model. The \_\_str\_\_ method provides a human-readable representation of the model instances when they are printed.

* Many-to-One Relationship: This is a specific type of ForeignKey relationship where multiple instances of one model (in this case, LeaveApplication) can be related to a single instance of another model (EmployeeData). This implies that many leave applications (LeaveApplication) can be associated with a single employee (EmployeeData).
* on\_delete=models.CASCADE: This parameter specifies the behavior when the referenced object (EmployeeData) is deleted. In this case, it's set to CASCADE, meaning that if an EmployeeData object is deleted, all related LeaveApplication objects will be deleted as well.

**Many-to-Many Relationship**

Many-to-Many Relationship: In a many-to-many relationship, instances of one model can be associated with multiple instances of another model, and vice versa. In this case, each department (Department model) can have multiple employees (EmployeeData model), and each employee can belong to multiple departments.



* class Department(models.Model):: This line defines a Django model named Department that inherits from models.Model, indicating that it's meant to be a database model.
* Employee = models.ManyToManyField(EmployeeData): This line creates a ManyToManyField named Employee in the Department model. A ManyToManyField is used to represent a many-to-many relationship between instances of the current model (Department) and another model (EmployeeData in this case). It implies that each department can be associated with multiple employees, and each employee can belong to multiple departments.
* paper = models.CharField(max\_length=100): This line defines a CharField named paper in the Department model. It's a character field with a maximum length of 100 characters, presumably meant to store information about the department.
* def \_\_str\_\_(self):: This method is defined to provide a string representation of the model instance. In this case, it returns the value of the paper field. Therefore, when you print an instance of this model, it will display the value of the paper field as a string.

So, in summary, this code defines a Django model for a department, where each department can have multiple employees (many-to-many relationship with EmployeeData), and it has a paper field to store information about the department. The \_\_str\_\_ method is implemented to display the value of the paper field when instances of this model are printed.