**Module 2 Cheat Sheet: ORM: Bridging the Gap Between the Real World and Relational Model**

| **Package/Method** | **Description** | **Code Example** |
| --- | --- | --- |
| **django.db.models.Model** | Define a model. | from django.db import models  class MyModel(models.Model):      field1 = models.CharField(max\_length=100)      field2 = models.IntegerField() |
| **makemigrations/migrate** | Create database tables based on models. | python manage.py makemigrations  python manage.py migrate |
| **all()** | Retrieves all instances of the ‘MyModel’ model from the database. | MyModel.objects.all() |
| **filter()** | Filter objects using conditions. | MyModel.objects.filter(field1="value")  MyModel.objects.filter(field2\_\_gt=5) |
| **get()** | Retrieves a single instance of the ‘MyModel’ model from the database where the value of ‘field1’ is "value". | MyModel.objects.get(field1="value") |
| **obj = MyModel(field1="value", field2=5)**  **obj.save()** | Creates a new instance of the ‘MyModel’ model with the values "value" for ‘field1’ and 5 for ‘field2’, and then saves the instance to the database. | obj = MyModel(field1="value", field2=5)  obj.save() |
| **obj.field1 = "new value"**  **obj.save()** | Updates the value of ‘field1’ for the ‘obj’ instance to "new value" and saves the changes to the database. | obj.field1 = "new value"  obj.save() |
| **delete()** | Deletes an object. | obj.delete() |
| **obj.related\_model** | Retrieves the related model associated with the ‘obj’ instance. Access related objects (Foreign Key or OneToOneField) | obj.related\_model |
| **obj.model\_set.all()** | Fetches all related objects associated with the ‘obj’ instance. Access related objects in reverse (ForeignKey) | obj.model\_set.all() |
| **field** | Performs a filtering operation on the ‘MyModel’ model instances based on a related model's field value. | MyModel.objects.filter(related\_model\_\_field="value") |
| **exact** | Retrieves instances of the ‘MyModel’ model from the database where the value of the ‘field’ attribute is exactly equal to "value". | MyModel.objects.filter(field\_\_exact="value") |
| **iexact** | The iexact lookup is case-insensitive, meaning it will match values regardless of whether they are uppercase or lowercase and provide a case-insensitive match. | MyModel.objects.filter(field\_\_iexact="value") |
| **contains** | Checks if the value is a substring within the field. | MyModel.objects.filter(field\_\_contains="value") |
| **startswith** | Determines whether a string begins with the characters of a specified string. | MyModel.objects.filter(field\_\_startswith="value") |
| **endswith** | Determines whether a string ends with the specified suffix. | MyModel.objects.filter(field\_\_endswith="value") |
| **in** | Checks if the value of the field is present in the given list of values. | MyModel.objects.filter(field\_\_in=["value1", "value2"]) |
| **gt** | Checks if the value of ‘field’ is numerically greater than 5. | MyModel.objects.filter(field\_\_gt=5) |
| **lt** | Checks if the value of ‘field’ is numerically less than 10. | MyModel.objects.filter(field\_\_lt=10) |