CRUD COMMANDS – DJANGO

this can ***ALL*** be done in the shell (**python manage.py shell**) which can be faster to check data than writing new route, linking in views, creating html, etc

**to use shell:** must first import models!!! Once in shell: **from your\_app\_name.models import \***

1. Creating a new record
   1. ClassName.objects.**create**(field1="value for field1", field2="value for field2", etc.)
2. Reading existing records
   1. Methods that return a single instance of a class
      1. ClassName.objects.**first**() - gets the first record in the table
      2. ClassName.objects.**last**() - gets the last record in the table
      3. ClassName.objects.**get**(id=1) - gets the record in the table with the specified id
         1. **this method will throw an error unless *only and exactly one record matches* the query**
   2. Methods that return a list of instances of a class
      1. ClassName.objects.**all**() - gets all the records in the table
      2. ClassName.objects.**filter**(field1="value for field1", etc.) - gets any records matching the query provided
      3. ClassName.objects.**exclude**(field1="value for field1", etc.) - gets any records *not* matching the query provided
3. Updating an existing record
   1. c = ClassName.objects.**get**(id=1)  
      c.field\_name = "some new value for field\_name"  
      c.**save**()
4. Deleting an existing record
   1. c = ClassName.objects.**get**(id=1)  
      c.**delete**()
5. Other helpful methods
   1. Displaying records
      1. ClassName.objects.get(id=1).**\_\_dict\_\_** - shows all the values of a single record as a dictionary
      2. ClassName.objects.all().**values**() - shows all the values of a QuerySet (i.e. multiple instances)
   2. Ordering records
      1. ClassName.objects.all().**order\_by**("field\_name") - orders by field provided, ascending
      2. ClassName.objects.all().**order\_by**("**-**field\_name") - orders by field provided, descending