**Database migration and other database work…**

(venv) C:\Users\gmnai\Python\_Practice\Environments\FF\_web\_app>python manage.py makemigrations

Migrations for 'blog':

blog\migrations\0001\_initial.py

- Create model Post

(venv) C:\Users\gmnai\Python\_Practice\Environments\FF\_web\_app>python manage.py sqlmigrate blog 0001

BEGIN;

--

-- Create model Post

--

***After migrate the table is created:***

CREATE TABLE "blog\_post" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "title" varchar(100) NOT NULL, "content" text NOT NULL, "date\_posted" datetime NOT NULL, "author\_id" integer NOT NULL REFERENCES "auth\_user" ("id") DEFERRABLE INITIALLY DEFERRED);

CREATE INDEX "blog\_post\_author\_id\_dd7a8485" ON "blog\_post" ("author\_id");

COMMIT;

(venv) C:\Users\gmnai\Python\_Practice\Environments\FF\_web\_app>python manage.py migrate

Operations to perform:

Apply all migrations: admin, auth, blog, contenttypes, sessions

Running migrations:

Applying blog.0001\_initial... OK

From here working into shell:

(venv) C:\Users\gmnai\Python\_Practice\Environments\FF\_web\_app>python manage.py shell

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

(InteractiveConsole)

>>> from blog.models import Post

>>> from django.contrib.auth.models import User

>>> User.objects.all()

<QuerySet [<User: gmnaim>, <User: gmnai>]>

>>> User.objects.first()

<User: gmnaim>

>>> User.objects.last()

<User: gmnai>

>>> User.objects.filter(username='gmnaim')

<QuerySet [<User: gmnaim>]>

>>> User.objects.filter(username='gmnaim').first()

<User: gmnaim>

>>> User.objects.filter(username='gmnai').first()

<User: gmnai>

>>> user = User.objects.filter(username='gmnaim').first()

>>> user

<User: gmnaim>

>>> user.id

1

>>> user.pk

1

>>> user = User.objects.get(id=1)

>>> user

<User: gmnaim>

>>> user = User.objects.get(id=3)

Traceback (most recent call last):

File "<console>", line 1, in <module>

File "C:\Users\gmnai\Python\_Practice\Environments\venv\lib\site-packages\django\db\models\manager.py", line 82, in manager\_method

return getattr(self.get\_queryset(), name)(\*args, \*\*kwargs)

File "C:\Users\gmnai\Python\_Practice\Environments\venv\lib\site-packages\django\db\models\query.py", line 399, in get

self.model.\_meta.object\_name

django.contrib.auth.models.User.DoesNotExist: User matching query does not exist.

>>> user = User.objects.get(id=1)

>>> user

<User: gmnaim>

>>> Post.objects.all()

<QuerySet []>

***>>> post\_1 = Post(title='Blog 1', content='First Post Content', author=user)***

>>> Post.objects.all()

<QuerySet []>

***>>> post\_1.save()***

>>> Post.objects.all()

<QuerySet [<Post: Post object (1)>]>

>>> exit()

(venv) C:\Users\gmnai\Python\_Practice\Environments\FF\_web\_app>python manage.py shell

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

(InteractiveConsole)

>>> from blog.models import Post

>>> from django.contrib.auth.models import User

>>> Post.objects.all()

<QuerySet [<Post: Blog 1>]>

>>> user = User.objects.filter(username='gmnaim').first()

>>> user

<User: gmnaim>

***>>> post\_2 = Post(title='Blog 2', content='Second Post Content', author\_id=user.id)***

***>>> post\_2.save()***

>>> Post.objects.all()

<QuerySet [<Post: Blog 1>, <Post: Blog 2>]>

>>> post = Post.objects.first()

>>> post.content

'First Post Content'

>>> post.date\_posted

datetime.datetime(2018, 9, 25, 20, 19, 39, 174590, tzinfo=<UTC>)

>>> post.author

<User: gmnaim>

>>> post.author.email

'gmnaim3336@gmail.com'

>>> user

<User: gmnaim>

>>> user.post\_set

<django.db.models.fields.related\_descriptors.create\_reverse\_many\_to\_one\_manager.<locals>.RelatedManager object at 0x00921030>

>>> user.post\_set.all()

<QuerySet [<Post: Blog 1>, <Post: Blog 2>]>

***>>> user.post\_set.create(title='Blog 3', content='Third Post Content')***

<Post: Blog 3>

>>> Post.objects.all()

<QuerySet [<Post: Blog 1>, <Post: Blog 2>, <Post: Blog 3>]>

>>> exit()