These are the commands for the follow-along exercises in Lesson 2.3.

2.304 Writing models

Commands

```
# workon advanced_web_dev
# cd topic2
# django-admin startproject bioweb
# cd bioweb
# python manage.py startapp genedata
```

Code

```
class Gene(models.Model):
    gene_id = models.CharField(max_length=256, null=False,
                               blank=False)
    entity = models.CharField(max_length=256, null=False,
                               blank=False)
    start = models.IntegerField(null=False, blank=True)
    stop = models.IntegerField(null=False, blank=True)
    sense = models.CharField(max_length=1)
    start_codon = models.CharField(max_length=1, default="M")
    sequencing = models.ForeignKey(Sequencing, on_delete=models.DO_NOTHING)
    ec = models.ForeignKey(EC, on_delete=models.DO_NOTHING)

    def __str__(self):
        return self.gene_id

class EC(models.Model):
    ec_name = models.CharField(max_length=256, null=False, blank=False)

    def __str__(self):
        return self.ec_name

class Sequencing(models.Model):
    sequencing_factory = models.CharField(max_length=256, null=False,
blank=False)
    factory_location = models.CharField(max_length=256, null=False, blank=False)

    def __str__(self):
        return self.factory_location


class Product(models.Model):
    type = models.CharField(max_length=256, null=False,
                               blank=False)
    product = models.CharField(max_length=256, null=False, blank=False)
    gene = models.ForeignKey(Gene, on_delete=models.CASCADE)

class Attribute(models.Model):
    key = models.CharField(max_length=256, null=False, blank=False)
    value = models.CharField(max_length=256, null=False, blank=False)
    gene = models.ManyToManyField(Gene, through='GeneAttributeLink')

    def __str__(self):
       return self.key+":"+self.value

class GeneAttributeLink(models.Model):
    gene = models.ForeignKey(Gene, on_delete=models.DO_NOTHING)
    attribute = models.ForeignKey(Attribute, on_delete=models.DO_NOTHING)
```

2.306 Migrations

Code

```
'genedata.apps.GenedataConfig',
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'bioweb_db',
        'USER': 'coder',
        'PASSWORD': '',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Commands

```
# psql -d postgres -h localhost -U coder
# CREATE DATABASE bioweb_db;
# \q

# python manage.py showmigrations
# python manage.py makemigrations
# python manage.py showmigrations
# ls genedata/migrations/0001_initial.py
# python manage.py sqlmigrate genedata 0001_initial
# python manage.py migrate

# psql -d postgres
# /c bioweb_db
# \d
```

Code

```
gene_id = models.CharField(max_length=256, null=False,
                        blank=False, db_index=True)
```

Commands

```
# python manage.py makemigrations
# python manage.py showmigrations
# python manage.py migrate
# \d genedata_gene
# select * from django_migrations where app='genedata';'
```

2.308 Adding database contents: Django admin

Commands

```
python manage.py createsuperuser
```

Code

```
ALLOWED_HOSTS = ['.coursera-apps.org',]
```

Commands

```
python manage.py runserver
```

Code admin.py

```
from .models import *
```

```
class GeneAttributeLinkInline(admin.TabularInline):
    model = GeneAttributeLink
    extra = 3

class GeneAdmin(admin.ModelAdmin):
    list_display = ('gene_id', 'entity', 'start', 'stop', 'sense')
    inlines = [GeneAttributeLinkInline]

class ECAdmin(admin.ModelAdmin):
    list_display = ('ec_name', )

class SequencingAdmin(admin.ModelAdmin):
    list_display = ('sequencing_factory', 'factory_location')

admin.site.register(Gene, GeneAdmin)
admin.site.register(EC, ECAdmin)
admin.site.register(Sequencing, SequencingAdmin)
```

2.310 Adding database contents: Writing a script

Code

```
import os

import sys
import django
import csv
from collections import defaultdict
sys.path.append("/home/coder/project/topic2/bioweb")
os.environ.setdefault('DJANGO_SETTINGS_MODULE',
                      'bioweb.settings')
django.setup()
from genedata.models import *
data_file = '/home/coder/project/topic2/scripts/example_data_to_load.csv'

genes = defaultdict(list)
sequencing = set()
ec = set()
products = defaultdict(dict)
attributes = defaultdict(dict)

with open(data_file) as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    header = csv_reader.__next__()
    for row in csv_reader:
      product_pairs = row[9].split(';')
      attribute_pairs = row[10].split(';')
      for pair in product_pairs:
          tupple = pair.split(":")
          products[row[0]][tupple[0]] = tupple[1]
      for pair in attribute_pairs:
          tupple = pair.split(":")
          attributes[row[0]][tupple[0]] = tupple[1]
      ec.add(row[8])
      sequencing.add((row[4], row[5]))
      genes[row[0]] = row[1:4]+row[6:9]

GeneAttributeLink.objects.all().delete()

Gene.objects.all().delete()
EC.objects.all().delete()
Sequencing.objects.all().delete()
```

```
Attribute.objects.all().delete()
Product.objects.all().delete()

ec_rows = {}
sequencing_rows = {}
gene_rows = {}

for entry in ec:
    row = EC.objects.create(ec_name=entry)
    row.save()
    ec_rows[entry] = row
for seq_centre in sequencing:
    row = Sequencing.objects.create(sequencing_factory=seq_centre[0],
                                    factory_location=seq_centre[1])
    row.save()
    sequencing_rows[seq_centre[0]] = row

for gene_id, data in genes.items():
    row = Gene.objects.create(gene_id=gene_id, entity = data[0],
                              start=data[1], stop=data[2],
                              sense=data[3], start_codon=data[4],
                              sequencing = sequencing_rows['Sanger'],
                              ec=ec_rows[data[5]])
    row.save()
    gene_rows[gene_id] = row

for gene_id, data_dict in products.items():
    for key in data_dict.keys():
        row = Product.objects.create(type=key, product=data_dict[key],
                                     gene=gene_rows[gene_id])
        row.save()

for gene_id, data_dict in attributes.items():
    for key in data_dict.keys():
        row = Attribute.objects.create(key=key, value=data_dict[key])
        row.gene.add(gene_rows[gene_id])
        row.save()
```