01- Creating Models

```
class Product(models.Model):
    title = models.CharField(max_length=255)
    slug = models.SlugField()
    description = models.TextField(null=True, blank=True)
    unit_price = models.DecimalField(max_digits=6, decimal_places=2,
validators=[MinValueValidator(1)])
    inventory = models.IntegerField()
    last_update = models.DateTimeField(auto_now=True)

# Exercise:
class Customer(models.Model):
    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    email = models.EmailField(unique=True)
    phone = models.CharField(max_length=255)
    birth_date = models.DateField(null=True)
```

##

02- Choice Fields

```
class Customer(models.Model):
    MEMBERSHIP_BRONZE = 'B'
    MEMBERSHIP_SILVER = 'S'
    MEMBERSHIP_GOLD = 'G'
```

```
    MEMBERSHIP_CHOICES = [
        (MEMBERSHIP_BRONZE, 'Bronze'),
        (MEMBERSHIP_SILVER, 'Silver'),
        (MEMBERSHIP_GOLD, 'Gold'),
    ]
    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    email = models.EmailField(unique=True)
    phone = models.CharField(max_length=255)
    birth_date = models.DateField(null=True)
    membership = models.CharField(
        max_length=1, choices=MEMBERSHIP_CHOICES, default=MEMBERSHIP_BRONZE)
```

Exercise:

```
class Order(models.Model):
    PAYMENT_STATUS_PENDING = 'P'
    PAYMENT_STATUS_COMPLETE = 'C'
    PAYMENT_STATUS_FAILED = 'F'
    PAYMENT_STATUS_CHOICES = [
        (PAYMENT_STATUS_PENDING, 'Pending'),
        (PAYMENT_STATUS_COMPLETE, 'Complete'),
        (PAYMENT_STATUS_FAILED, 'Failed')
    ]

    placed_at = models.DateTimeField(auto_now_add=True)
    payment_status = models.CharField(
        max_length=1, choices=PAYMENT_STATUS_CHOICES,
default=PAYMENT_STATUS_PENDING)
```

```
# ############################################################
# 03- Defining One-to-one Relationships

class Address(models.Model):
```

```
    street = models.CharField(max_length=255)
    city = models.CharField(max_length=255)
    customer = models.OneToOneField(
        Customer, on_delete=models.CASCADE, primary_key=True)




# ############################################################
# 04- Defining a One-to-many Relationship

class Address(models.Model):
    street = models.CharField(max_length=255)
    city = models.CharField(max_length=255)
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)

Exercise:

class Product(models.Model):
    title = models.CharField(max_length=255)
    slug = models.SlugField()
    description = models.TextField(null=True, blank=True)
    unit_price = models.DecimalField(max_digits=6, decimal_places=2,
validators=[MinValueValidator(1)])
    inventory = models.IntegerField()
    last_update = models.DateTimeField(auto_now=True)
    collection = models.ForeignKey(Collection, on_delete=models.PROTECT)

class Order(models.Model):
    PAYMENT_STATUS_PENDING = 'P'
    PAYMENT_STATUS_COMPLETE = 'C'
    PAYMENT_STATUS_FAILED = 'F'
    PAYMENT_STATUS_CHOICES = [
        (PAYMENT_STATUS_PENDING, 'Pending'),
        (PAYMENT_STATUS_COMPLETE, 'Complete'),
        (PAYMENT_STATUS_FAILED, 'Failed')
```

```
    ]

    placed_at = models.DateTimeField(auto_now_add=True)
    payment_status = models.CharField(
        max_length=1, choices=PAYMENT_STATUS_CHOICES,
default=PAYMENT_STATUS_PENDING)
    customer = models.ForeignKey(Customer, on_delete=models.PROTECT)


class OrderItem(models.Model):
    order = models.ForeignKey(Order, on_delete=models.PROTECT)
    product = models.ForeignKey(Product, on_delete=models.PROTECT)
    quantity = models.PositiveSmallIntegerField()
    unit_price = models.DecimalField(max_digits=6, decimal_places=2)


class CartItem(models.Model):
    cart = models.ForeignKey(Cart, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    quantity = models.PositiveSmallIntegerField()




# ############################################################
# 05- Defining Many-to-many Relationships

class Promotion(models.Model):
    description = models.CharField(max_length=255)
    discount = models.FloatField()
```

```
class Product(models.Model):
    title = models.CharField(max_length=255)
    slug = models.SlugField()
    description = models.TextField(null=True, blank=True)
    unit_price = models.DecimalField(max_digits=6, decimal_places=2,
validators=[MinValueValidator(1)])
    inventory = models.IntegerField()
    last_update = models.DateTimeField(auto_now=True)
    collection = models.ForeignKey(Collection, on_delete=models.PROTECT)
    promotions = models.ManyToManyField(Promotion, blank=True)
```

```
# ############################################################
# 06- Resolving Circular Relationships

Circular Dependency Happens when 2 classes depend on each other at the same time

class Collection(models.Model):
    title = models.CharField(max_length=255)
    featured_product = models.ForeignKey(
        'Product', on_delete=models.SET_NULL, null=True, related_name='+')


class Product(models.Model):
    title = models.CharField(max_length=255)
    slug = models.SlugField()
    description = models.TextField(null=True, blank=True)
```

```
    unit_price = models.DecimalField(max_digits=6, decimal_places=2,
validators=[MinValueValidator(1)])
    inventory = models.IntegerField()
    last_update = models.DateTimeField(auto_now=True)
    collection = models.ForeignKey(Collection, on_delete=models.PROTECT)
    promotions = models.ManyToManyField(Promotion, blank=True)




# ############################################################
# 07- Generic Relationships

class Tag(models.Model):
    label = models.CharField(max_length=255)


class TaggedItem(models.Model):
    objects = TaggedItemManager()
    tag = models.ForeignKey(Tag, on_delete=models.CASCADE)
    # Content Type(Type)
    content_type = models.ForeignKey(ContentType, on_delete=models.CASCADE)
    # Object Id(ID)
    object_id = models.PositiveIntegerField()
    # Content Object(Actual Object)
    content_object = GenericForeignKey()

# Exercise:
class LikedItem(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    content_type = models.ForeignKey(ContentType, on_delete=models.CASCADE)
    object_id = models.PositiveIntegerField()
    content_object = GenericForeignKey()
```