Django\_model

Here's your requested Django Model:  
  
```  
from django.db import models  
from django.utils.dateformat import format  
  
class OrderItem(models.Model):  
 id = models.AutoField(primary\_key=True)  
 order = models.ForeignKey('orders.Order', related\_name='items', on\_delete=models.CASCADE)  
 product = models.ForeignKey('products.Product', related\_name='order\_items', on\_delete=models.CASCADE)  
 quantity = models.IntegerField()  
 subtotal = models.DecimalField(max\_digits=10, decimal\_places=2)  
  
 def \_\_str\_\_(self):  
 return f"Order Item {self.id} - Product: {self.product.name}, Quantity: {self.quantity}"  
  
 class Meta:  
 ordering = ['-created\_at']  
 db\_table = "Order\_Items"  
```  
  
  
This Python code defines the `OrderItem` model as per the given database schema. The fields used are:   
  
- `id`: An auto-incrementing primary key   
- `order`: A foreign key referencing the `Order` model  
- `product`: A foreign key referencing the `Product` model  
- `quantity`: An integer field representing the quantity of products ordered  
- `subtotal`: A decimal field representing the subtotal cost of each item  
  
The `\_\_str\_\_` method provides a human-readable representation of an instance of this model.  
  
The `Meta` inner class specifies that the table name will be "Order\_Items", and orders instances by their creation date (`-created\_at`).

Here's your requested Django Model:  
  
```  
from django.db import models  
from django.utils.dateformat import format  
  
class OrderItem(models.Model):  
 id = models.AutoField(primary\_key=True)  
 order = models.ForeignKey('orders.Order', related\_name='items', on\_delete=models.CASCADE)  
 product = models.ForeignKey('products.Product', related\_name='order\_items', on\_delete=models.CASCADE)  
 quantity = models.IntegerField()  
 subtotal = models.DecimalField(max\_digits=10, decimal\_places=2)  
  
 def \_\_str\_\_(self):  
 return f"Order Item {self.id} - Product: {self.product.name}, Quantity: {self.quantity}"  
  
 class Meta:  
 ordering = ['-created\_at']  
 db\_table = "Order\_Items"  
```  
  
  
This Python code defines the `OrderItem` model as per the given database schema. The fields used are:   
  
- `id`: An auto-incrementing primary key   
- `order`: A foreign key referencing the `Order` model  
- `product`: A foreign key referencing the `Product` model  
- `quantity`: An integer field representing the quantity of products ordered  
- `subtotal`: A decimal field representing the subtotal cost of each item  
  
The `\_\_str\_\_` method provides a human-readable representation of an instance of this model.  
  
The `Meta` inner class specifies that the table name will be "Order\_Items", and orders instances by their creation date (`-created\_at`).

Here's your requested Django Model:  
  
```  
from django.db import models  
from django.utils.dateformat import format  
  
class OrderItem(models.Model):  
 id = models.AutoField(primary\_key=True)  
 order = models.ForeignKey('orders.Order', related\_name='items', on\_delete=models.CASCADE)  
 product = models.ForeignKey('products.Product', related\_name='order\_items', on\_delete=models.CASCADE)  
 quantity = models.IntegerField()  
 subtotal = models.DecimalField(max\_digits=10, decimal\_places=2)  
  
 def \_\_str\_\_(self):  
 return f"Order Item {self.id} - Product: {self.product.name}, Quantity: {self.quantity}"  
  
 class Meta:  
 ordering = ['-created\_at']  
 db\_table = "Order\_Items"  
```  
  
  
This Python code defines the `OrderItem` model as per the given database schema. The fields used are:   
  
- `id`: An auto-incrementing primary key   
- `order`: A foreign key referencing the `Order` model  
- `product`: A foreign key referencing the `Product` model  
- `quantity`: An integer field representing the quantity of products ordered  
- `subtotal`: A decimal field representing the subtotal cost of each item  
  
The `\_\_str\_\_` method provides a human-readable representation of an instance of this model.  
  
The `Meta` inner class specifies that the table name will be "Order\_Items", and orders instances by their creation date (`-created\_at`).

Here's your requested Django Model:  
  
```  
from django.db import models  
from django.utils.dateformat import format  
  
class OrderItem(models.Model):  
 id = models.AutoField(primary\_key=True)  
 order = models.ForeignKey('orders.Order', related\_name='items', on\_delete=models.CASCADE)  
 product = models.ForeignKey('products.Product', related\_name='order\_items', on\_delete=models.CASCADE)  
 quantity = models.IntegerField()  
 subtotal = models.DecimalField(max\_digits=10, decimal\_places=2)  
  
 def \_\_str\_\_(self):  
 return f"Order Item {self.id} - Product: {self.product.name}, Quantity: {self.quantity}"  
  
 class Meta:  
 ordering = ['-created\_at']  
 db\_table = "Order\_Items"  
```  
  
  
This Python code defines the `OrderItem` model as per the given database schema. The fields used are:   
  
- `id`: An auto-incrementing primary key   
- `order`: A foreign key referencing the `Order` model  
- `product`: A foreign key referencing the `Product` model  
- `quantity`: An integer field representing the quantity of products ordered  
- `subtotal`: A decimal field representing the subtotal cost of each item  
  
The `\_\_str\_\_` method provides a human-readable representation of an instance of this model.  
  
The `Meta` inner class specifies that the table name will be "Order\_Items", and orders instances by their creation date (`-created\_at`).

Here's your requested Django Model:  
  
```  
from django.db import models  
from django.utils.dateformat import format  
  
class OrderItem(models.Model):  
 id = models.AutoField(primary\_key=True)  
 order = models.ForeignKey('orders.Order', related\_name='items', on\_delete=models.CASCADE)  
 product = models.ForeignKey('products.Product', related\_name='order\_items', on\_delete=models.CASCADE)  
 quantity = models.IntegerField()  
 subtotal = models.DecimalField(max\_digits=10, decimal\_places=2)  
  
 def \_\_str\_\_(self):  
 return f"Order Item {self.id} - Product: {self.product.name}, Quantity: {self.quantity}"  
  
 class Meta:  
 ordering = ['-created\_at']  
 db\_table = "Order\_Items"  
```  
  
  
This Python code defines the `OrderItem` model as per the given database schema. The fields used are:   
  
- `id`: An auto-incrementing primary key   
- `order`: A foreign key referencing the `Order` model  
- `product`: A foreign key referencing the `Product` model  
- `quantity`: An integer field representing the quantity of products ordered  
- `subtotal`: A decimal field representing the subtotal cost of each item  
  
The `\_\_str\_\_` method provides a human-readable representation of an instance of this model.  
  
The `Meta` inner class specifies that the table name will be "Order\_Items", and orders instances by their creation date (`-created\_at`).

Here's your requested Django Model:  
  
```  
from django.db import models  
from django.utils.dateformat import format  
  
class OrderItem(models.Model):  
 id = models.AutoField(primary\_key=True)  
 order = models.ForeignKey('orders.Order', related\_name='items', on\_delete=models.CASCADE)  
 product = models.ForeignKey('products.Product', related\_name='order\_items', on\_delete=models.CASCADE)  
 quantity = models.IntegerField()  
 subtotal = models.DecimalField(max\_digits=10, decimal\_places=2)  
  
 def \_\_str\_\_(self):  
 return f"Order Item {self.id} - Product: {self.product.name}, Quantity: {self.quantity}"  
  
 class Meta:  
 ordering = ['-created\_at']  
 db\_table = "Order\_Items"  
```  
  
  
This Python code defines the `OrderItem` model as per the given database schema. The fields used are:   
  
- `id`: An auto-incrementing primary key   
- `order`: A foreign key referencing the `Order` model  
- `product`: A foreign key referencing the `Product` model  
- `quantity`: An integer field representing the quantity of products ordered  
- `subtotal`: A decimal field representing the subtotal cost of each item  
  
The `\_\_str\_\_` method provides a human-readable representation of an instance of this model.  
  
The `Meta` inner class specifies that the table name will be "Order\_Items", and orders instances by their creation date (`-created\_at`).

Here's your requested Django Model:  
  
```  
from django.db import models  
from django.utils.dateformat import format  
  
class OrderItem(models.Model):  
 id = models.AutoField(primary\_key=True)  
 order = models.ForeignKey('orders.Order', related\_name='items', on\_delete=models.CASCADE)  
 product = models.ForeignKey('products.Product', related\_name='order\_items', on\_delete=models.CASCADE)  
 quantity = models.IntegerField()  
 subtotal = models.DecimalField(max\_digits=10, decimal\_places=2)  
  
 def \_\_str\_\_(self):  
 return f"Order Item {self.id} - Product: {self.product.name}, Quantity: {self.quantity}"  
  
 class Meta:  
 ordering = ['-created\_at']  
 db\_table = "Order\_Items"  
```  
  
  
This Python code defines the `OrderItem` model as per the given database schema. The fields used are:   
  
- `id`: An auto-incrementing primary key   
- `order`: A foreign key referencing the `Order` model  
- `product`: A foreign key referencing the `Product` model  
- `quantity`: An integer field representing the quantity of products ordered  
- `subtotal`: A decimal field representing the subtotal cost of each item  
  
The `\_\_str\_\_` method provides a human-readable representation of an instance of this model.  
  
The `Meta` inner class specifies that the table name will be "Order\_Items", and orders instances by their creation date (`-created\_at`).

Here's your requested Django Model:  
  
```  
from django.db import models  
from django.utils.dateformat import format  
  
class OrderItem(models.Model):  
 id = models.AutoField(primary\_key=True)  
 order = models.ForeignKey('orders.Order', related\_name='items', on\_delete=models.CASCADE)  
 product = models.ForeignKey('products.Product', related\_name='order\_items', on\_delete=models.CASCADE)  
 quantity = models.IntegerField()  
 subtotal = models.DecimalField(max\_digits=10, decimal\_places=2)  
  
 def \_\_str\_\_(self):  
 return f"Order Item {self.id} - Product: {self.product.name}, Quantity: {self.quantity}"  
  
 class Meta:  
 ordering = ['-created\_at']  
 db\_table = "Order\_Items"  
```  
  
  
This Python code defines the `OrderItem` model as per the given database schema. The fields used are:   
  
- `id`: An auto-incrementing primary key   
- `order`: A foreign key referencing the `Order` model  
- `product`: A foreign key referencing the `Product` model  
- `quantity`: An integer field representing the quantity of products ordered  
- `subtotal`: A decimal field representing the subtotal cost of each item  
  
The `\_\_str\_\_` method provides a human-readable representation of an instance of this model.  
  
The `Meta` inner class specifies that the table name will be "Order\_Items", and orders instances by their creation date (`-created\_at`).