سوال ۱) منیجر چیست و برای کی مدل یک نمونه پیاده سازی کنید؟

منیجر یک رابط است که از طریق آن کو‍‍يریهای دیتابیس برای ارتباط با مدل های جنگو آماده میشوند.

سوال ۲)

1. Multi-table inheritance چیست؟

وقتی که همه ی مدلهایی که داریم به تنهایی خودشان یک تیبل جداگانه هستند.هرمدل مربوط به تیبل مخصوص به خودش است و می تواند به صورت جداگونه ساخته شود و روی آن کويری زده شود. رابطه ارث بری به صورت اتوماتیک یک لینک بین مدل چیلد و پرنت ان برقرار میکند.(یک رابطه ی یک به یک)

1. Abstract base classesچیست؟

این نوع از کلاس ها مواقعی مفید هستند که شما میخواهید یکسری اطلاعات مشترک را برای تعدادی از مدل های دیگر استفاده کنید. ابتدا کلاس بیس را مینویسیم و در کلاس متا **abstract=True**  قرار می دهیم. این مدل برای ساختن تیبل جدید در دیتابیس استفاده نمیشود. به جای آن وقتی که به عنوان یک کلاس بیس برای سایر کلاس ها مورد استفاده قرار فیلدهای آن برای کلاس چایلد استفاده میشوند.

1. Proxy Modelچیست؟

وراثت مدل پروکسی برای ایجاد یک پروکسی برای مدل اصلی آن است. شما می توانید نمونه هایی از مدل پروکسی را ایجاد، حذف و به روز کنید و تمام داده ها به گونه ای ذخیره می شوند که گویی از مدل اصلی (غیر پروکسی) استفاده می کنید. تفاوت این است که می‌توانید مواردی مانند سفارش مدل پیش‌فرض یا مدیر پیش‌فرض در پروکسی را بدون نیاز به تغییر نسخه اصلی تغییر دهید.

سوال ۳)

Product.objects.create(product\_name='ipod', category=mycategory, supplier=mysupplier, unit\_price=60, is\_discontinued=False)

Part1

1. from shopUsers.models import Product
2. from django.db.models import Avg
3. price\_average = Product.objects.aggregate(Avg('unit\_price'))
4. Product.objects.filter(unit\_price\_\_gt=price\_average['unit\_price\_\_avg'])

Part2

1. from shopUsers.models import User, Customer, Supplier, Comment, EmailToSupplier, EmailToCustomer, Category, Product, Order, OrderItem
2. from django.db.models import Count, Max
3. customer\_order\_count\_query = Customer.objects.annotate(order\_count=Count('order'))
4. max\_orders\_count= customer\_order\_count\_query.aggregate(Max('order\_count'))
5. customer\_order\_count\_query.filter(order\_count= max\_orders\_count['order\_count\_\_max'])

part3

1. query\_product\_quantity\_sold = Product.objects.annotate(count\_item=Sum('orderitem\_\_quantity'))
2. query\_product\_quantity\_sold.order\_by('-count\_item')

part4

1. query\_category\_quantity\_sold = Category.objects.annotate(count\_item=Sum('product\_\_orderitem\_\_quantity'))
2. query\_category\_quantity\_sold.filter(count\_item=max\_quantity\_sold\_category['count\_item\_\_max'])
3. max\_quantity\_sold\_category = query\_category\_quantity\_sold.aggregate(Max('count\_item'))

part5

1. query\_category\_quantity\_sold = Category.objects.annotate(count\_item=Sum('product\_\_orderitem\_\_quantity'))
2. query\_category\_quantity\_sold.order\_by('-count\_item')[:3]

part6

1. Customer.objects.filter(order\_\_order\_date\_\_week\_day=2).distinct()

Part7

1. count\_tag\_product = Product.objects.annotate(count\_tags=Count('tag'))
2. count\_tag\_product.aggregate(Avg('count\_tags'))