📋 Тестовое задание: “Мини-сервис аналитики заказов”

Задача:
Реализовать Django-приложение orders, которое принимает данные о заказах в формате JSON, сохраняет их в базу, и формирует простую агрегированную статистику по пользователю.

1️⃣ Модели

Создай три модели:

```
class User(models.Model):
    username = models.CharField(max_length=50, unique=True)

class Order(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE,
related_name="orders")
    order_number = models.CharField(max_length=50,
unique=True)
    created_at = models.DateTimeField()
    total_amount = models.DecimalField(max_digits=12,
decimal_places=2)
    status = models.CharField(max_length=20)

class OrderItem(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE,
related_name="items")
    sku = models.CharField(max_length=50)
    name = models.CharField(max_length=255)
    quantity = models.PositiveIntegerField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
```

2️⃣ API-эндпоинты

Реализуй два простых view (можно DRF или обычный Django View):

а) POST /api/orders/upload

Принимает JSON вида:

```
{
  "user": "test_seller",
  "orders": [
    {
      "order_number": "12345",
      "created_at": "2025-11-12T10:00:00Z",
      "total_amount": 2500.00,
      "status": "delivered",
      "items": [
        {"sku": "tea01", "name": "Чай зелёный", "quantity": 2, "price": 500},
        {"sku": "cup01", "name": "Чашка фарфоровая", "quantity": 1, "price": 1500}
      ]
    }
  ]
}
```

Требования:

при повторной загрузке заказа по order_number должен происходить **update** (не дубликат);

записи сохраняются атомарно (transaction.atomic);

использовать bulk_create/bulk_update где целесообразно;

валидировать базовые типы данных.

b) GET /api/orders/stats?user=test_seller
Возвращает JSON статистику:

```
{
  "user": "test_seller",
  "orders_count": 10,
  "total_revenue": 12345.67,
  "avg_order_value": 1234.56
}
```

3️⃣ Дополнительные задания (по желанию)
⚙️ Добавить Celery-таску, которая ежедневно считает статистику (daily_order_stats)

🧾 Добавить кастомный Django Admin для Order с мини-таблицей товаров

🪵 Добавить logger в upload-метод