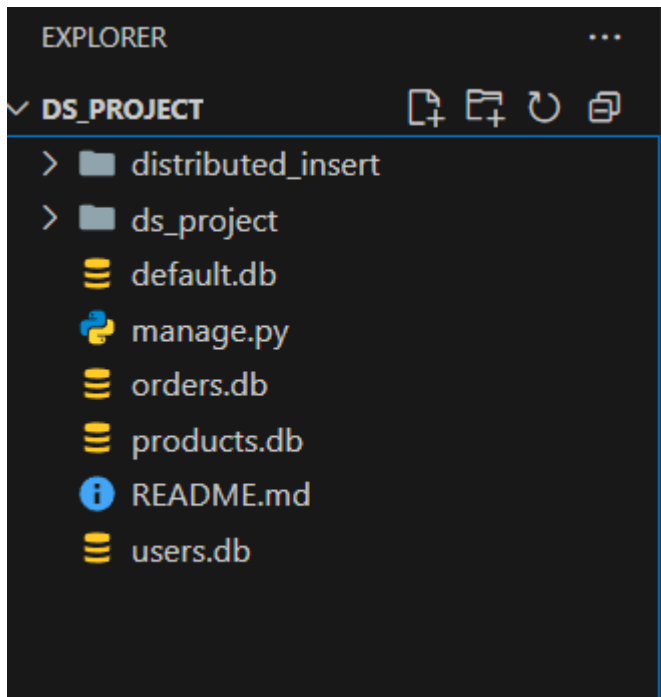
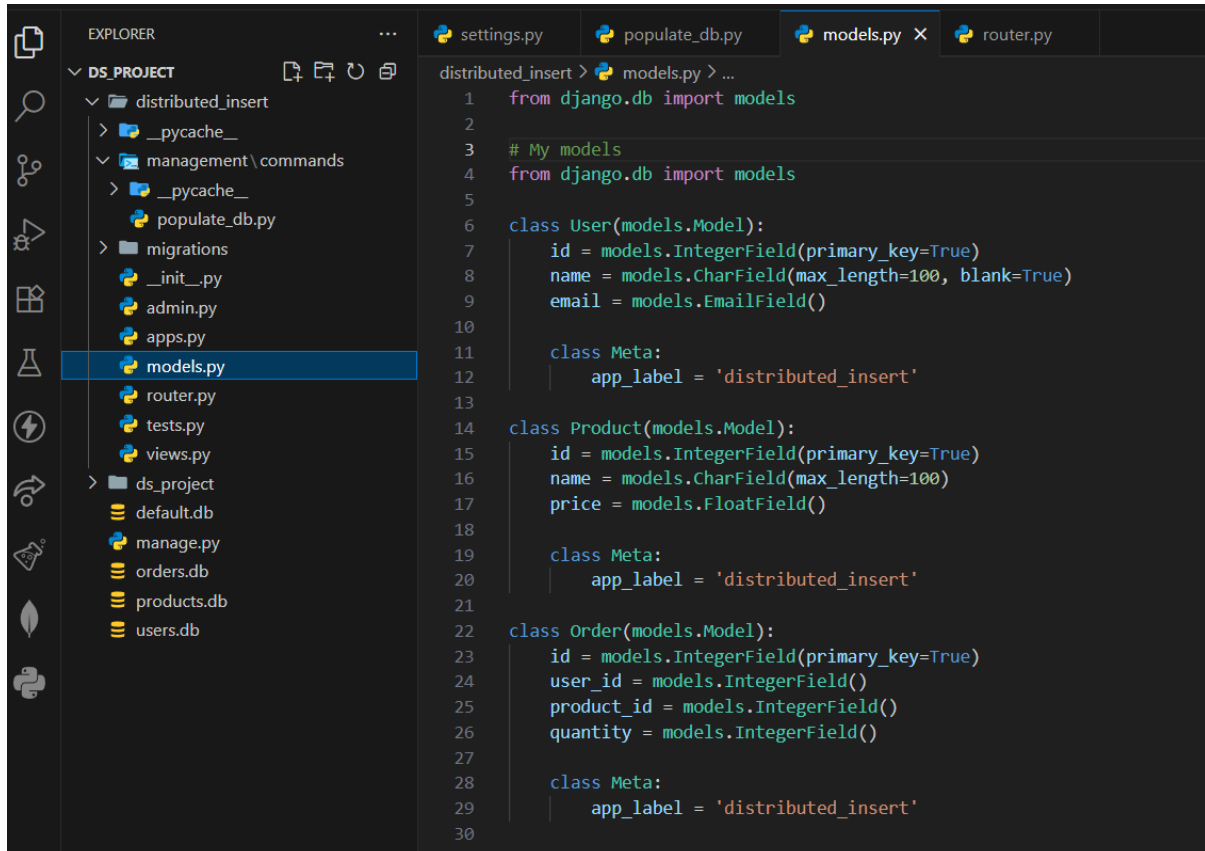


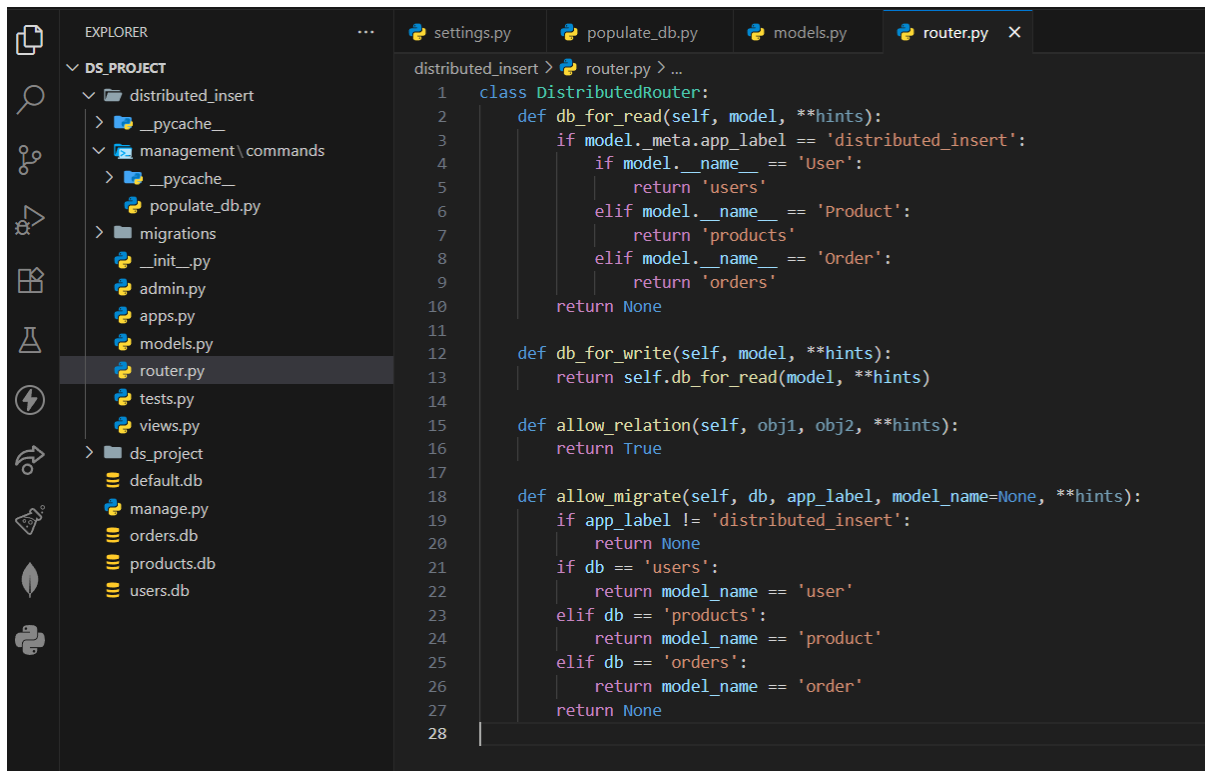
## 1. Project Structure:



## 2. Django Models Definition:



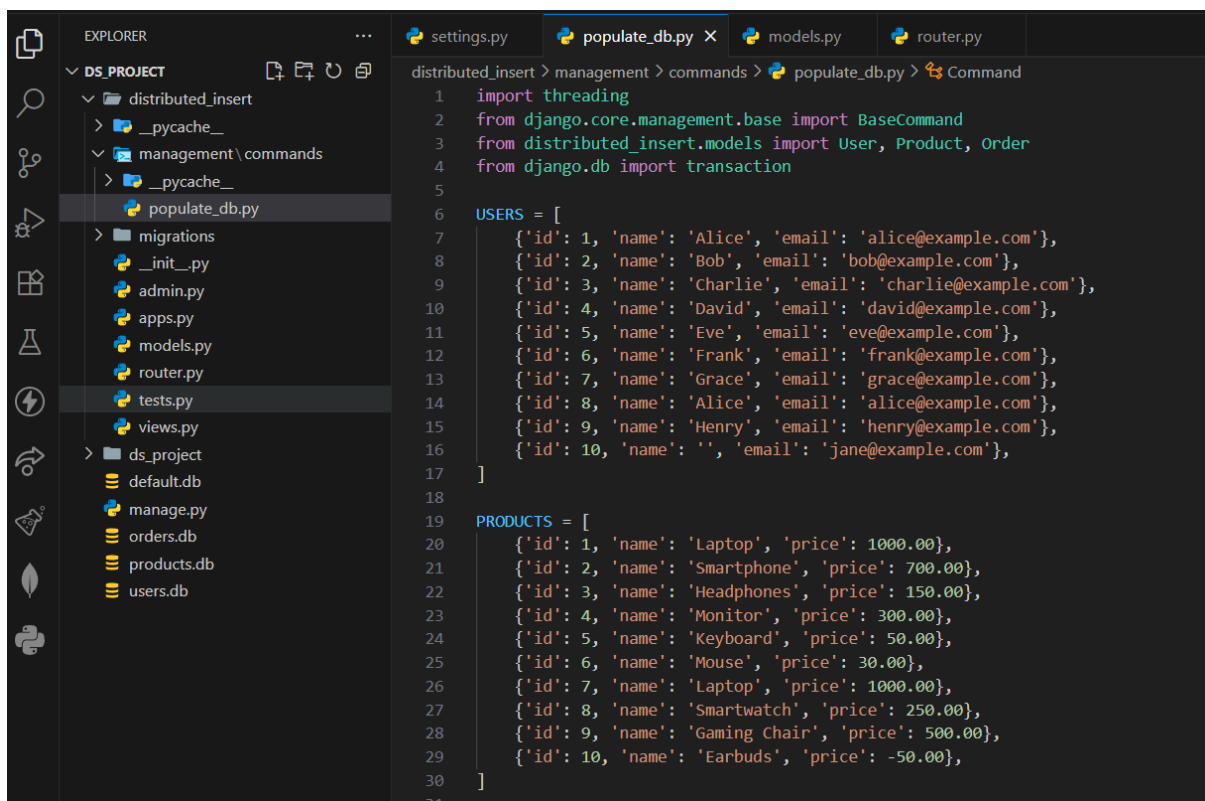
### 3. Database Router Implementation:



The screenshot shows the VS Code interface with the Explorer on the left and the Code editor on the right. The Explorer shows a project structure with a 'distributed\_insert' app. The Code editor shows the 'router.py' file with the following code:

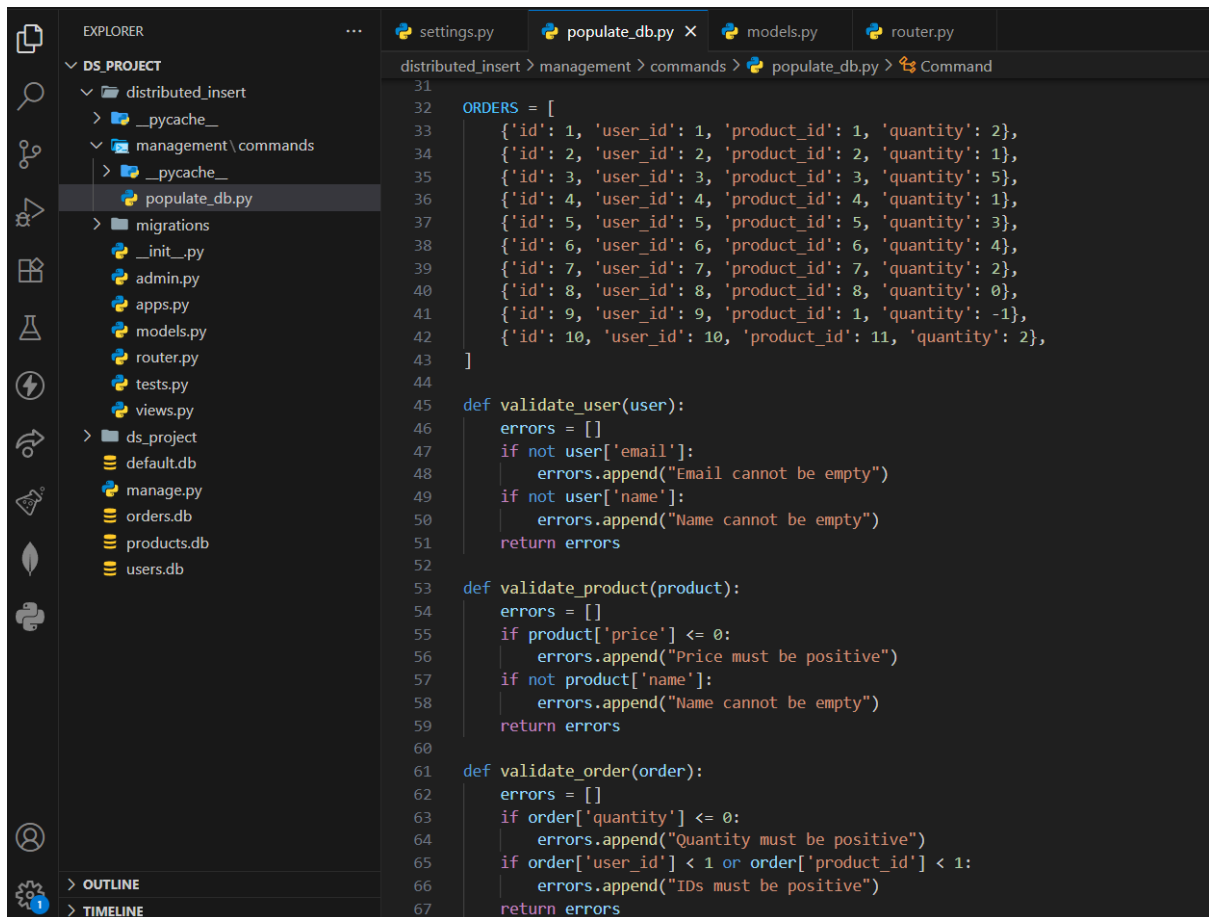
```
distributed_insert > router.py > ...
1 class DistributedRouter:
2     def db_for_read(self, model, **hints):
3         if model._meta.app_label == 'distributed_insert':
4             if model._name == 'User':
5                 return 'users'
6             elif model._name == 'Product':
7                 return 'products'
8             elif model._name == 'Order':
9                 return 'orders'
10            return None
11
12     def db_for_write(self, model, **hints):
13         return self.db_for_read(model, **hints)
14
15     def allow_relation(self, obj1, obj2, **hints):
16         return True
17
18     def allow_migrate(self, db, app_label, model_name=None, **hints):
19         if app_label != 'distributed_insert':
20             return None
21         if db == 'users':
22             return model_name == 'user'
23         elif db == 'products':
24             return model_name == 'product'
25         elif db == 'orders':
26             return model_name == 'order'
27         return None
28
```

### 4. Sample Data Definition & Validate Logic:



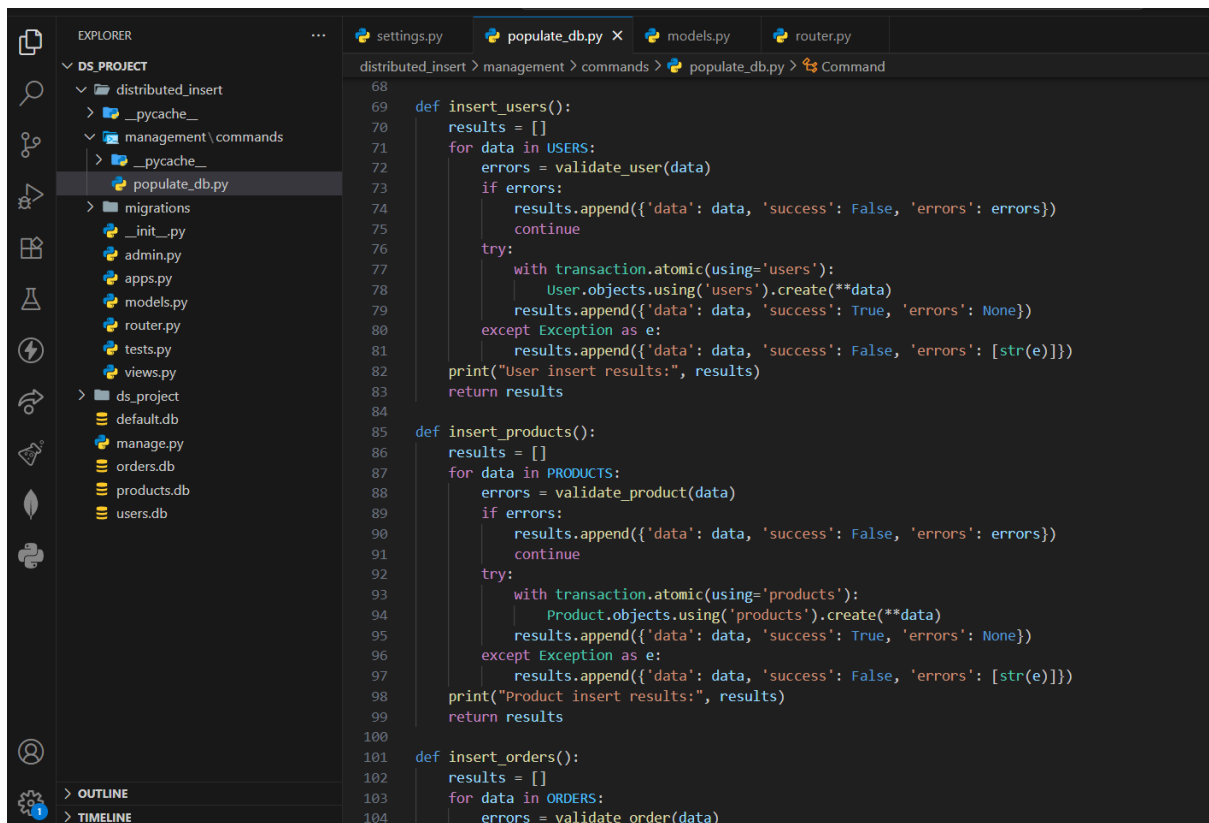
The screenshot shows the VS Code interface with the Explorer on the left and the Code editor on the right. The Explorer shows the same project structure as before. The Code editor shows the 'populate\_db.py' file with the following code:

```
distributed_insert > management > commands > populate_db.py > Command
1 import threading
2 from django.core.management.base import BaseCommand
3 from distributed_insert.models import User, Product, Order
4 from django.db import transaction
5
6 USERS = [
7     {'id': 1, 'name': 'Alice', 'email': 'alice@example.com'},
8     {'id': 2, 'name': 'Bob', 'email': 'bob@example.com'},
9     {'id': 3, 'name': 'Charlie', 'email': 'charlie@example.com'},
10    {'id': 4, 'name': 'David', 'email': 'david@example.com'},
11    {'id': 5, 'name': 'Eve', 'email': 'eve@example.com'},
12    {'id': 6, 'name': 'Frank', 'email': 'frank@example.com'},
13    {'id': 7, 'name': 'Grace', 'email': 'grace@example.com'},
14    {'id': 8, 'name': 'Alice', 'email': 'alice@example.com'},
15    {'id': 9, 'name': 'Henry', 'email': 'henry@example.com'},
16    {'id': 10, 'name': '', 'email': 'jane@example.com'},
17 ]
18
19 PRODUCTS = [
20     {'id': 1, 'name': 'Laptop', 'price': 1000.00},
21     {'id': 2, 'name': 'Smartphone', 'price': 700.00},
22     {'id': 3, 'name': 'Headphones', 'price': 150.00},
23     {'id': 4, 'name': 'Monitor', 'price': 300.00},
24     {'id': 5, 'name': 'Keyboard', 'price': 50.00},
25     {'id': 6, 'name': 'Mouse', 'price': 30.00},
26     {'id': 7, 'name': 'Laptop', 'price': 1000.00},
27     {'id': 8, 'name': 'Smartwatch', 'price': 250.00},
28     {'id': 9, 'name': 'Gaming Chair', 'price': 500.00},
29     {'id': 10, 'name': 'Earbuds', 'price': -50.00},
30 ]
31
```



```
31
32 ORDERS = [
33     {'id': 1, 'user_id': 1, 'product_id': 1, 'quantity': 2},
34     {'id': 2, 'user_id': 2, 'product_id': 2, 'quantity': 1},
35     {'id': 3, 'user_id': 3, 'product_id': 3, 'quantity': 5},
36     {'id': 4, 'user_id': 4, 'product_id': 4, 'quantity': 1},
37     {'id': 5, 'user_id': 5, 'product_id': 5, 'quantity': 3},
38     {'id': 6, 'user_id': 6, 'product_id': 6, 'quantity': 4},
39     {'id': 7, 'user_id': 7, 'product_id': 7, 'quantity': 2},
40     {'id': 8, 'user_id': 8, 'product_id': 8, 'quantity': 0},
41     {'id': 9, 'user_id': 9, 'product_id': 1, 'quantity': -1},
42     {'id': 10, 'user_id': 10, 'product_id': 11, 'quantity': 2},
43 ]
44
45 def validate_user(user):
46     errors = []
47     if not user['email']:
48         errors.append("Email cannot be empty")
49     if not user['name']:
50         errors.append("Name cannot be empty")
51     return errors
52
53 def validate_product(product):
54     errors = []
55     if product['price'] <= 0:
56         errors.append("Price must be positive")
57     if not product['name']:
58         errors.append("Name cannot be empty")
59     return errors
60
61 def validate_order(order):
62     errors = []
63     if order['quantity'] <= 0:
64         errors.append("Quantity must be positive")
65     if order['user_id'] < 1 or order['product_id'] < 1:
66         errors.append("IDs must be positive")
67     return errors
```

## 5. Threading Implementation

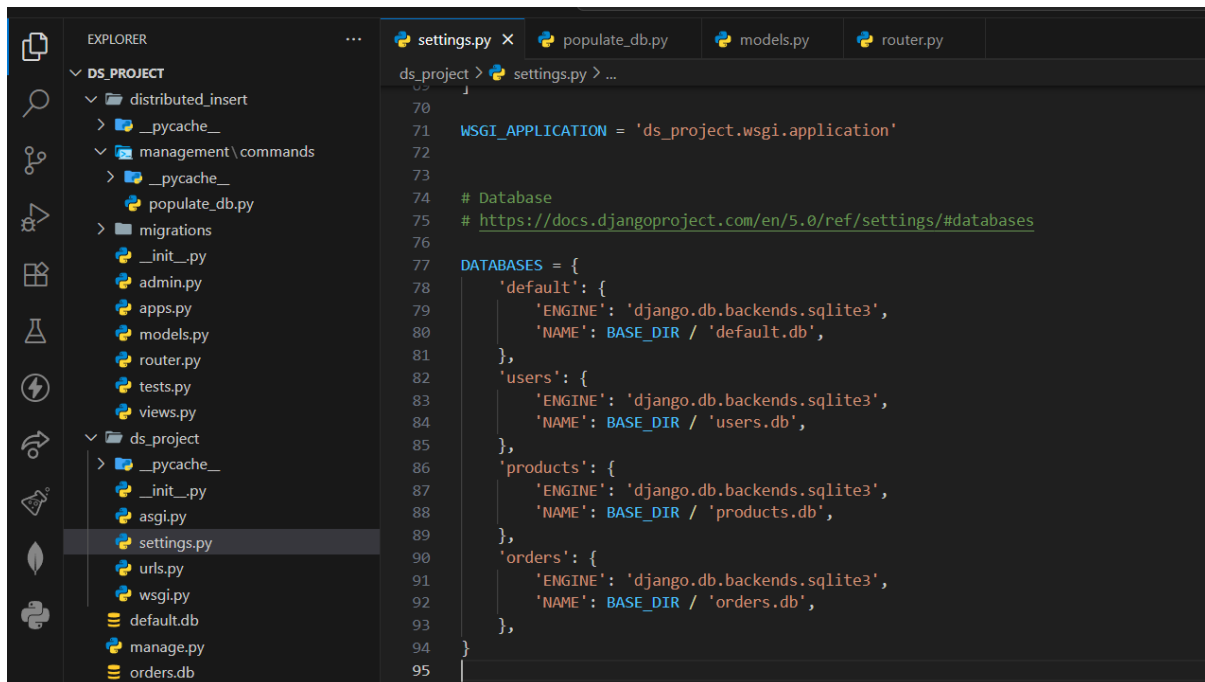


```
68
69 def insert_users():
70     results = []
71     for data in USERS:
72         errors = validate_user(data)
73         if errors:
74             results.append({'data': data, 'success': False, 'errors': errors})
75             continue
76         try:
77             with transaction.atomic(using='users'):
78                 User.objects.using('users').create(**data)
79             results.append({'data': data, 'success': True, 'errors': None})
80         except Exception as e:
81             results.append({'data': data, 'success': False, 'errors': [str(e)]})
82     print("User insert results:", results)
83     return results
84
85 def insert_products():
86     results = []
87     for data in PRODUCTS:
88         errors = validate_product(data)
89         if errors:
90             results.append({'data': data, 'success': False, 'errors': errors})
91             continue
92         try:
93             with transaction.atomic(using='products'):
94                 Product.objects.using('products').create(**data)
95             results.append({'data': data, 'success': True, 'errors': None})
96         except Exception as e:
97             results.append({'data': data, 'success': False, 'errors': [str(e)]})
98     print("Product insert results:", results)
99     return results
100
101 def insert_orders():
102     results = []
103     for data in ORDERS:
104         errors = validate_order(data)
```

## 5.1 Threading Strategy:

- Three separate threads for concurrent execution
- Each thread handles one model's data independently
- Thread.join() ensures all operations complete before exit
- Atomic transactions ensure data consistency

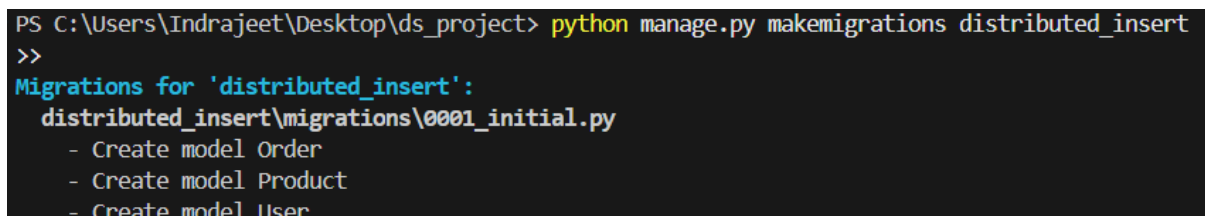
## 6. Database Configuration:



```
70
71 WSGI_APPLICATION = 'ds_project.wsgi.application'
72
73
74 # Database
75 # https://docs.djangoproject.com/en/5.0/ref/settings/#databases
76
77 DATABASES = {
78     'default': {
79         'ENGINE': 'django.db.backends.sqlite3',
80         'NAME': BASE_DIR / 'default.db',
81     },
82     'users': {
83         'ENGINE': 'django.db.backends.sqlite3',
84         'NAME': BASE_DIR / 'users.db',
85     },
86     'products': {
87         'ENGINE': 'django.db.backends.sqlite3',
88         'NAME': BASE_DIR / 'products.db',
89     },
90     'orders': {
91         'ENGINE': 'django.db.backends.sqlite3',
92         'NAME': BASE_DIR / 'orders.db',
93     },
94 }
95
```

## 7. Migration Process:

### Step 1: Creating Migrations



```
PS C:\Users\Indrajeet\Desktop\ds_project> python manage.py makemigrations distributed_insert
>>
Migrations for 'distributed_insert':
  distributed_insert\migrations\0001_initial.py
    - Create model Order
    - Create model Product
    - Create model User
```

## Step 2: Database-Specific Migrations

### Users Database Migration:

```
PS C:\Users\Indrajeet\Desktop\ds_project> python manage.py migrate --database=users
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, distributed_insert, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying distributed_insert.0001_initial... OK
  Applying sessions.0001_initial... OK
```

### Products Database Migration:

```
PS C:\Users\Indrajeet\Desktop\ds_project> python manage.py migrate --database=products
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, distributed_insert, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying distributed_insert.0001_initial... OK
  Applying sessions.0001_initial... OK
```

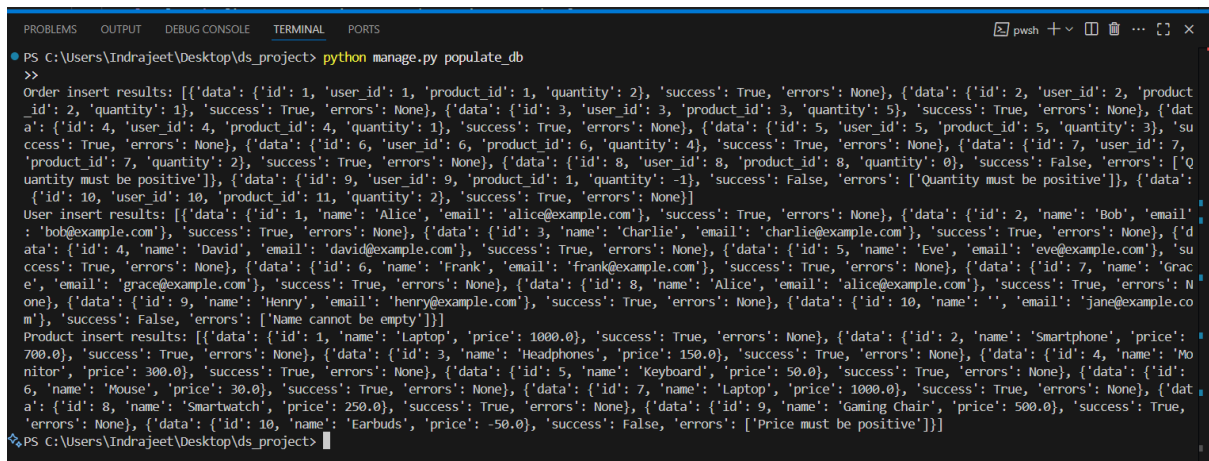
## Orders Database Migration:

```
PS C:\Users\Indrajeet\Desktop\ds_project> python manage.py migrate --database=orders
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, distributed_insert, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying distributed_insert.0001_initial... OK
```

## Migration Strategy:

- Each database receives full Django migration set
- Model-specific tables created in designated databases
- Database router ensures proper table placement

## Execution Results:



```
PS C:\Users\Indrajeet\Desktop\ds_project> python manage.py populate_db
>>
Order insert results: [{'data': {'id': 1, 'user_id': 1, 'product_id': 1, 'quantity': 2}, 'success': True, 'errors': None}, {'data': {'id': 2, 'user_id': 2, 'product_id': 2, 'quantity': 1}, 'success': True, 'errors': None}, {'data': {'id': 3, 'user_id': 3, 'product_id': 3, 'quantity': 5}, 'success': True, 'errors': None}, {'data': {'id': 4, 'user_id': 4, 'product_id': 4, 'quantity': 1}, 'success': True, 'errors': None}, {'data': {'id': 5, 'user_id': 5, 'product_id': 5, 'quantity': 3}, 'success': True, 'errors': None}, {'data': {'id': 6, 'user_id': 6, 'product_id': 6, 'quantity': 4}, 'success': True, 'errors': None}, {'data': {'id': 7, 'user_id': 7, 'product_id': 7, 'quantity': 2}, 'success': True, 'errors': None}, {'data': {'id': 8, 'user_id': 8, 'product_id': 8, 'quantity': 0}, 'success': False, 'errors': ['Quantity must be positive']}, {'data': {'id': 9, 'user_id': 9, 'product_id': 1, 'quantity': -1}, 'success': False, 'errors': ['Quantity must be positive']}, {'data': {'id': 10, 'user_id': 10, 'product_id': 11, 'quantity': 2}, 'success': True, 'errors': None}]
User insert results: [{'data': {'id': 1, 'name': 'Alice', 'email': 'alice@example.com'}, 'success': True, 'errors': None}, {'data': {'id': 2, 'name': 'Bob', 'email': 'bob@example.com'}, 'success': True, 'errors': None}, {'data': {'id': 3, 'name': 'Charlie', 'email': 'charlie@example.com'}, 'success': True, 'errors': None}, {'data': {'id': 4, 'name': 'David', 'email': 'david@example.com'}, 'success': True, 'errors': None}, {'data': {'id': 5, 'name': 'Eve', 'email': 'eve@example.com'}, 'success': True, 'errors': None}, {'data': {'id': 6, 'name': 'Frank', 'email': 'frank@example.com'}, 'success': True, 'errors': None}, {'data': {'id': 7, 'name': 'Grace', 'email': 'grace@example.com'}, 'success': True, 'errors': None}, {'data': {'id': 8, 'name': 'Alice', 'email': 'alice@example.com'}, 'success': True, 'errors': None}, {'data': {'id': 9, 'name': 'Henry', 'email': 'henry@example.com'}, 'success': True, 'errors': None}, {'data': {'id': 10, 'name': '', 'email': 'jane@example.com'}, 'success': False, 'errors': ['Name cannot be empty']}]
Product insert results: [{'data': {'id': 1, 'name': 'Laptop', 'price': 1000.0}, 'success': True, 'errors': None}, {'data': {'id': 2, 'name': 'Smartphone', 'price': 700.0}, 'success': True, 'errors': None}, {'data': {'id': 3, 'name': 'Headphones', 'price': 150.0}, 'success': True, 'errors': None}, {'data': {'id': 4, 'name': 'Monitor', 'price': 300.0}, 'success': True, 'errors': None}, {'data': {'id': 5, 'name': 'Keyboard', 'price': 50.0}, 'success': True, 'errors': None}, {'data': {'id': 6, 'name': 'Mouse', 'price': 30.0}, 'success': True, 'errors': None}, {'data': {'id': 7, 'name': 'Laptop', 'price': 1000.0}, 'success': True, 'errors': None}, {'data': {'id': 8, 'name': 'Smartwatch', 'price': 250.0}, 'success': True, 'errors': None}, {'data': {'id': 9, 'name': 'Gaming Chair', 'price': 500.0}, 'success': True, 'errors': None}, {'data': {'id': 10, 'name': 'Earbuds', 'price': -50.0}, 'success': False, 'errors': ['Price must be positive']}]
PS C:\Users\Indrajeet\Desktop\ds_project>
```

## Final Output Analysis:

### Concurrent Execution Results:

#### Orders Processing (8/10 successful):

- Orders 1-7: Successful insertions
- Order 8: Failed - Quantity = 0 (validation error)
- Order 9: Failed - Negative quantity (validation error)
- Order 10: Successful insertion

#### Users Processing (9/10 successful):

- Users 1-9: Successful insertions
- User 10: Failed - Empty name field (validation error)

#### Products Processing (9/10 successful):

- Products 1-9: Successful insertions
- Product 10: Failed - Negative price (validation error)

### Performance Analysis:

- Total Records Processed: 30 records
- Successful Insertions: 26 records (86.7% success rate)
- Validation Failures: 4 records (13.3% - as expected)
- Concurrent Execution: All three threads ran simultaneously
- Zero Database Errors: All failures were application-level validation