

1. Create project and application
  - a. Django>django-admin startproject orm\_and\_frontend
  - b. Django>cd orm\_and\_frontend
  - c. Django\orm\_and\_frontend>python manage.py startapp product
2. Open project in vs code
3. Register application in settings.py
4. Create database in workbench  
create database orm\_and\_frontend;  
use orm\_and\_frontend;
5. Register database in settings.py

```

settings.py X
orm_and_frontend > settings.py > ...
76
77 DATABASES = {
78     'default': {
79         'ENGINE': 'django.db.backends.mysql',
80         'NAME': 'orm_and_frontend',
81         'USER': 'root',
82         'PASSWORD': 'root',
83         'HOST': 'localhost',
84         'PORT': '3306'
85     }
86 }

```

6. Create model class for Products, so in models.py

```

models.py X
product > models.py > ProductTable
1 from django.db import models
2
3 # Create your models here.
4 class ProductTable(models.Model):
5     name = models.CharField(max_length=50)
6     price = models.FloatField()
7     details=models.CharField(max_length=150)
8     category = models.IntegerField()
9     is_active= models.BooleanField()
10    rating = models.FloatField()
11
12    def __str__(self) :
13        return self.name + " added to table"

```

7. Register model class in admin.py

```

admin.py X
product > admin.py > ...
1 from django.contrib import admin
2 from product.models import ProductTable
3
4 # Register your models here.
5 class ProductAdmin(admin.ModelAdmin):
6     list_display = ['id', 'name', 'price', 'details', 'category', 'is_active', 'rating']
7
8     admin.site.register(ProductTable, ProductAdmin)
9

```

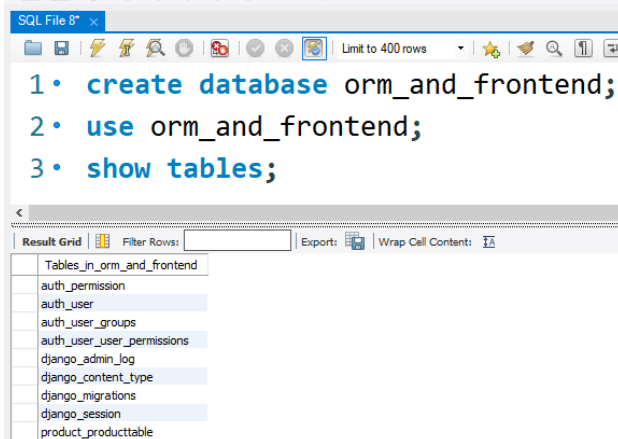
## 8. Makemigrations

- Django\orm\_and\_frontend>python manage.py makemigrations

## 9. Migrate

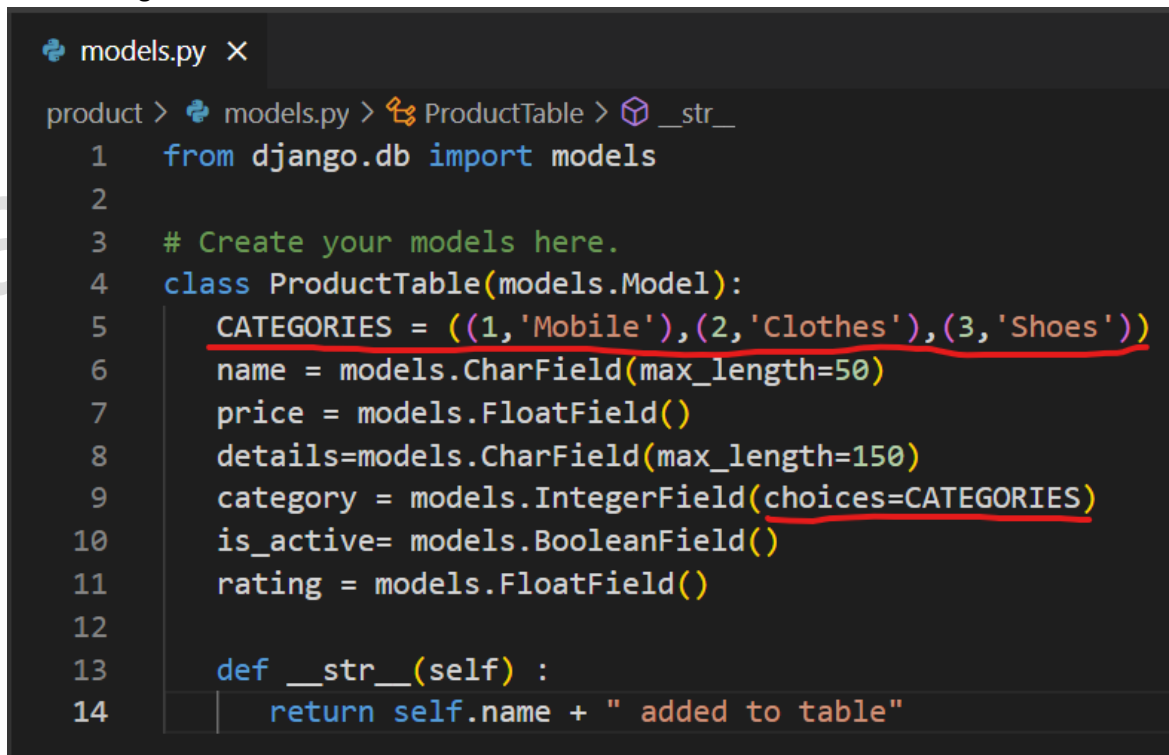
- Django\orm\_and\_frontend>python manage.py migrate

## 10. Check tables in orm\_and\_frontend database (in workbench)

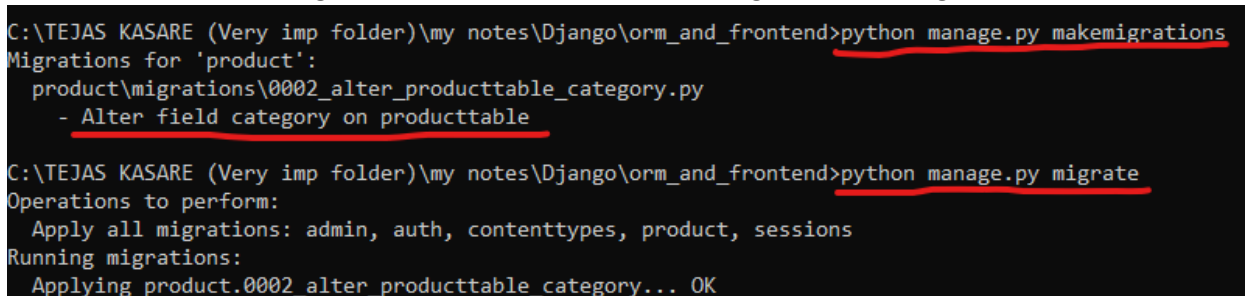


## 11. Providing categories and drop down option

- Make change in ProductTable model



- Since we have made changes in model, we have to do makemigrations and migrate



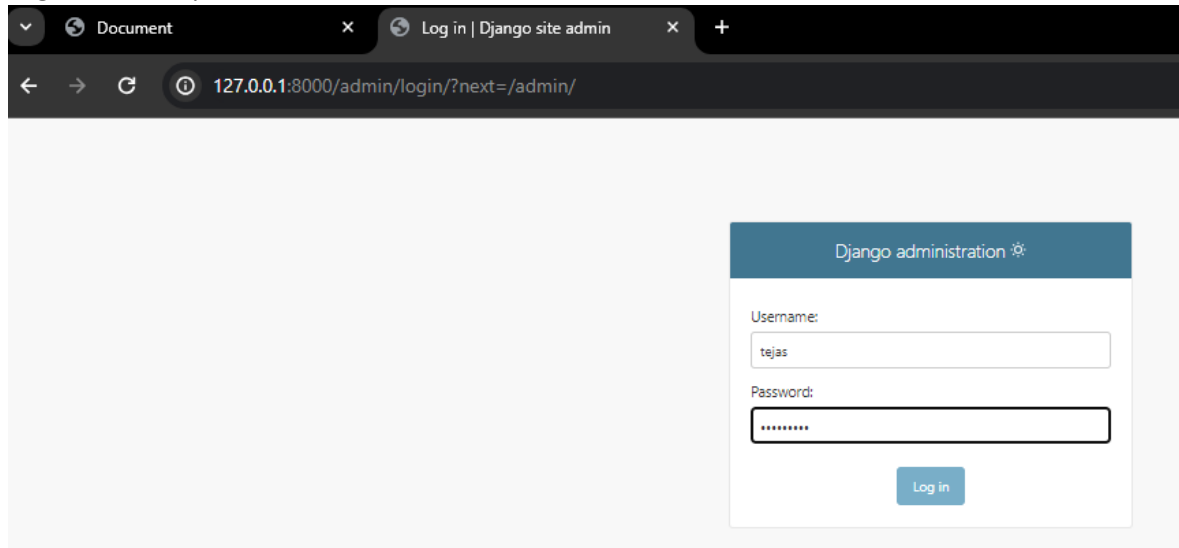
## 12. Adding some products into product table from admin panel

### a. Create superuser

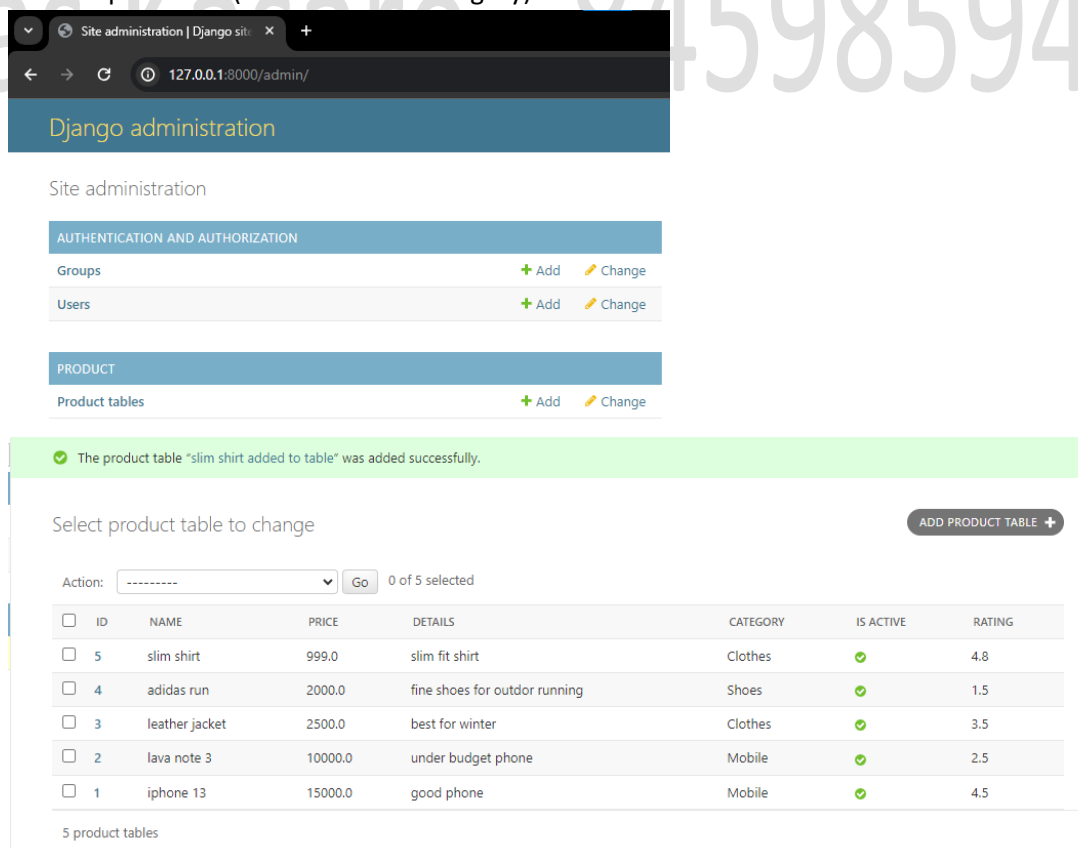
```
C:\TEJAS KASARE (Very imp folder)\my notes\Django\orm_and_frontend>python manage.py createsuperuser
Username (leave blank to use 'admin'): tejas
Email address: tejas@gmail.com
Password:
Password (again):
Superuser created successfully.
```

### b. runserver

### c. Login to admin panel

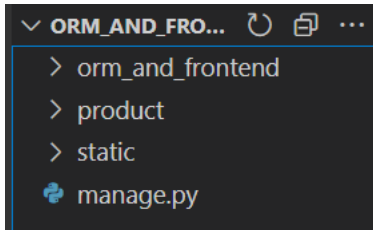


### d. Add 4-5 products (min 1 of each category)



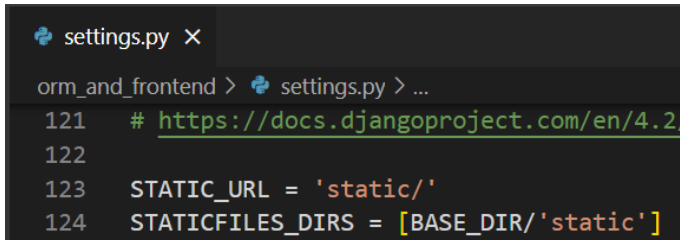
ID	NAME	PRICE	DETAILS	CATEGORY	IS ACTIVE	RATING
5	slim shirt	999.0	slim fit shirt	Clothes	✓	4.8
4	adidas run	2000.0	fine shoes for outdoor running	Shoes	✓	1.5
3	leather jacket	2500.0	best for winter	Clothes	✓	3.5
2	lava note 3	10000.0	under budget phone	Mobile	✓	2.5
1	iphone 13	15000.0	good phone	Mobile	✓	4.5

13. Creating static folder to store css file



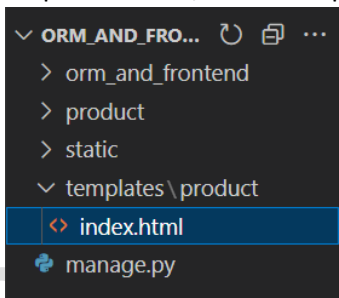
a.

14. Register static folder in settings.py



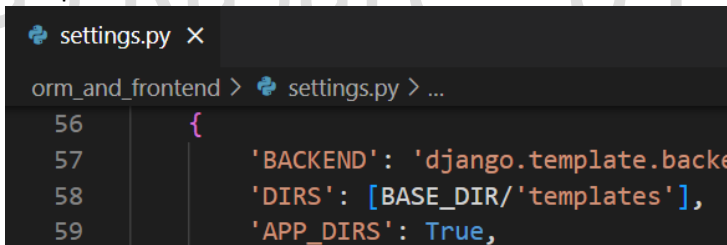
a.

15. Create templates folder, inside templates folder create product folder, create index.html in product folder



a.

16. Register templates folder



a.

17. In static folder do following :

- Create css folder > create product\_style.css
- Create images folder > add one shirt image in it (200x200 size)
- add following code in it

```
.container
{
    width: 100%;
    display: flex;
    flex-direction: row;
}

.filter_area
{
    width: 20%;
    display: flex;
    /* background-color: aqua; */
    border: 1px dashed black;
    flex-direction: column;
    padding: 10px;
}
```

```

.product_area
{
  width: 80%;
  display: flex;
  margin-left: 2px;
  /* border: 1px dashed black; */
  flex-wrap: wrap;
}

.card
{
  display: flex;
  flex-direction: column;
  width: 25%;
}

.card .card-items
{
  border: 1px solid black;
  padding: 10px;
  border-radius: 10px;
  display: flex;
  flex-direction: column;
  margin: 10px;
  align-items: center;
  justify-content: center;
  box-shadow: 3px 5px 5px grey;
}

.card .card-items img
{
  border-radius: 5%;
  height: 150px;
  width: 150px;
  padding: 5px;
}

.card .card-items button
{
  border-radius: 20px;
  padding: 12px;
  border: none;
}

.card .card-items #add_to_cart_btn
{
  background-color: #F7CA00;
}

.card .card-items #buy_now_btn
{
  background-color: #FFA41C;
}

.card .card-items button a
{
  text-decoration: none;
  color: black;
}

.card .card-items .card-text
{
  margin-left: 10px;
}

```

18. in templates folder do following

- create product folder > create index.html
- add following code in it

```
<!DOCTYPE html>
<html lang="en">
  {% load static %}
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="{% static 'css/product_style.css' %}">
</head>
<body>
  <div class="navbar"></div>
  <div class="container">
    <div class="filter_area">
      <div class="by_category">
        <h2>Filter By Category</h2>
        <ul>
          <li><a href="">All</a></li>
          <li><a href="">By Mobile</a></li>
          <li><a href="">By Clothes</a></li>
          <li><a href="">By Shoes</a></li>
        </ul>
      </div>
    </div>
    <div>-----</div>
    <div class="by_price">
      <h2>Filter By Price</h2>
      <form action="">
        <label for="">Min:</label>
        <input type="number"> <br><br>
        <label for="">Max:</label>
        <input type="number"> <br><br>
        <input type="submit">
      </form>
    </div>
    <div>-----</div>
    <div class="sort_by_price">
      <h2>Sort By Price</h2>
      <ul>
        <li><a href="">High to Low</a></li>
        <li><a href="">Low to High</a></li>
      </ul>
    </div>
    <div>-----</div>
    <div class="by_rating">
      <h2>Sort By Rating</h2>
      <ul>
        <li><a href="">3 and above</a></li>
        <li><a href="">4 and above</a></li>
      </ul>
    </div>
    <div>-----</div>
    <div class="product_area">
      <div class="card">
        <div class="card-items">
          
          <div class="card-text">
            <p>Cotton King</p>
            <p>499</p>
            <button id="add_to_cart_btn"><a href="">Add to Cart</a></button>
            <button id="buy_now_btn"><a href="">Buy Now</a></button>
          </div>
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```

19. create view to display index.html file

```
views.py X
product > views.py > ...
1 from django.shortcuts import render
2
3 # Create your views here.
4 def index(request):
5     return render(request, 'product/index.html')
6
```

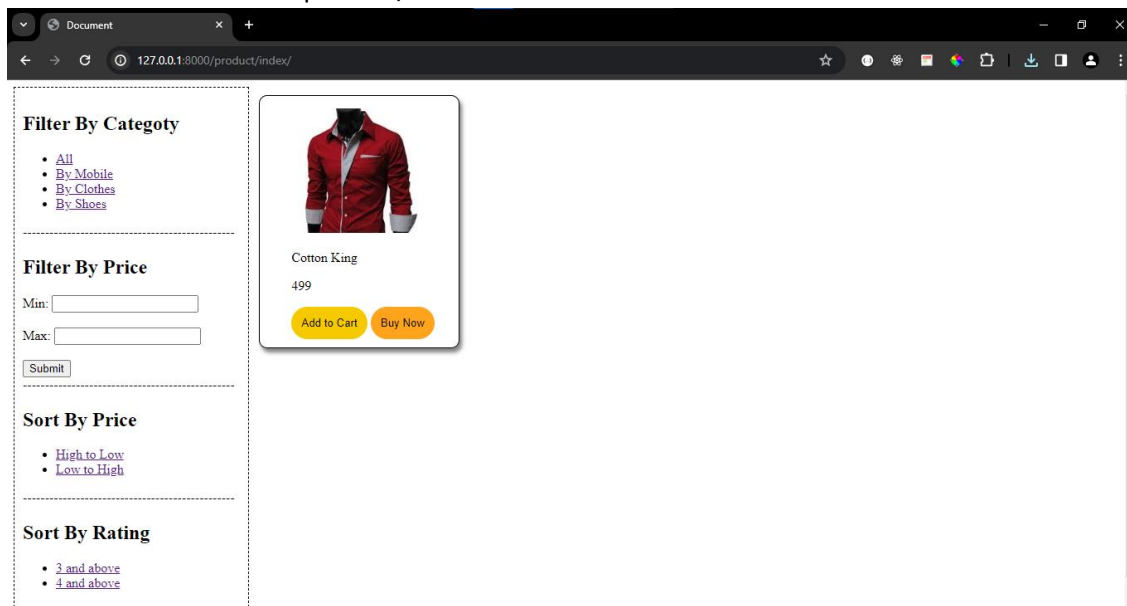
20. create product\_urls.py in product folder

```
product_urls.py X
product > product_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('index/', views.index),
6 ]
```

21. create url for above application level url in project level url (urls.py)

```
urls.py X
orm_and_frontend > urls.py > ...
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('product/', include('product.product_urls')),
23 ]
```

22. runserver and check for product/index url



# OPERATIONS

1. get all products and display in index.html
  - a. add logic to fetch data in index() view

```
views.py x
product > views.py > ...
1  from django.shortcuts import render
2  from product.models import ProductTable
3
4  # Create your views here.
5  def index(request):
6      data={}
7      #ProductTable.objects.all() this will fetch non active product also. so it is better to use filter
8      fetched_products=ProductTable.objects.filter(is_active=True)
9      data['products']=fetched_products
10     return render(request,'product/index.html',context=data)
11
```

- b. pass fetched data to index.html and display using loop

```
index.html x
templates > product > index.html
50  <div>-----</div>
51  </div>
52  <div class="product_area">
53      {% for product in products %}
54      <div class="card">
55          <div class="card-items">
56              
57              <div class="card-text">
58                  <p>{{product.name|upper}}</p>
59                  <p>{{product.price}}</p>
60                  <button id="add_to_cart_btn"><a href="">Add to Cart</a></button>
61                  <button id="buy_now_btn"><a href="">View More</a></button>
62              </div>
63          </div>
64      </div>
65      {% endfor %}
66  </div>
67  </div>
68  </body>
69  </html>
```

IMPORTANT : in above code I have changed Buy Now button to View More  
We will add buy now option when we show each product indivisibly (product details)



## 2. Implementing Filter by category logic

### a. Create view

```
views.py X
product > views.py > ...
12
13 def filter_by_category(request,category_value):
14     #select * from product where is_active=True and category=category_value;
15     #ProductTable.objects.filter(is_active=True , category=category_value)
16     #from django.db.models import Q
17     data={}
18     q1 = Q(is_active=True)
19     q2 = Q(category=category_value)
20     filtered_products=ProductTable.objects.filter(q1 & q2)
21     data['products']=filtered_products
22     return render(request,'product/index.html',context=data)
```

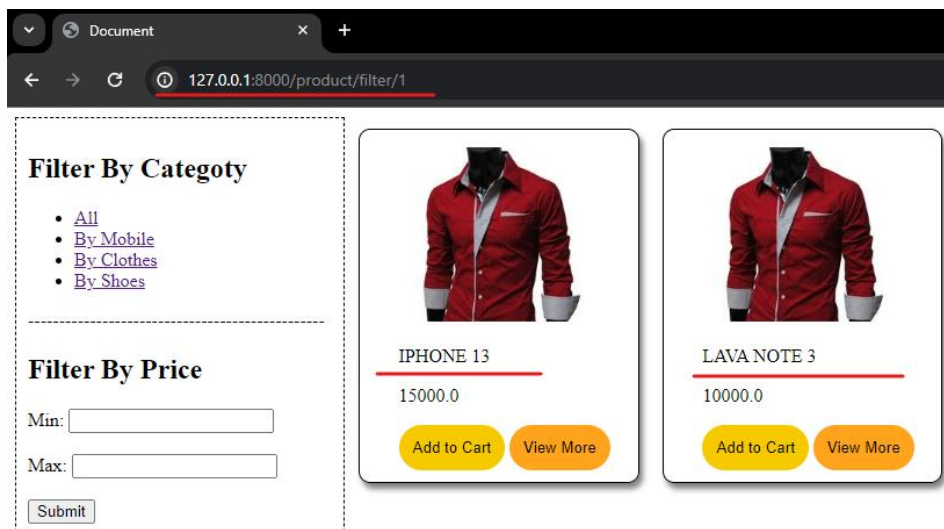
### b. Create url for view

```
product_urls.py X
product > product_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('index/', views.index),
6     path('filter/<category_value>', views.filter_by_category),
7 ]
```

### c. Use url in index.html

```
index.html X
templates > product > index.html
14 <div class="by_category">
15     <h2>Filter By Category</h2>
16     <ul>
17         <li><a href="/product/index">All</a></li>
18         <li><a href="/product/filter/1">By Mobile</a></li>
19         <li><a href="/product/filter/2">By Clothes</a></li>
20         <li><a href="/product/filter/3">By Shoes</a></li>
21     </ul>
22 </div>
```

Result :



### 3. Implementing sorting logic (high to low and low to high)

#### a. Create view

```
views.py
product > views.py > ...
23
24 def sort_by_price(request, sort_value):
25     #select * from product order by salary desc;
26     data={}
27     if sort_value=='asc':
28         price = 'price'
29     else:
30         price = '-price'
31     sorted_products=ProductTable.objects.filter(is_active=True).order_by(price)
32     data['products']=sorted_products
33     return render(request, 'product/index.html', context=data)
34
```

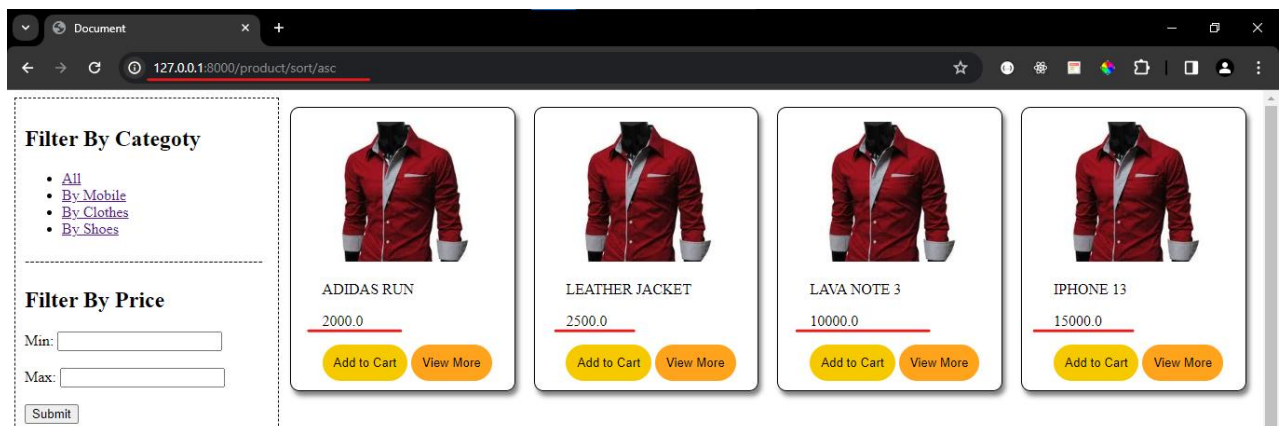
#### b. Create url for view

```
product_urls.py
product > product_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('index/', views.index),
6     path('filter/<category_value>', views.filter_by_category),
7     path('sort/<sort_value>', views.sort_by_price),
8 ]
```

#### c. Use url in index.html

```
index.html
templates > product > index.html
34 <div>-----</div>
35     <div class="sort_by_price">
36         <h2>Sort By Price</h2>
37         <ul>
38             <li><a href="/product/sort/asc">High to Low</a></li>
39             <li><a href="/product/sort/asc">Low to High</a></li>
40         </ul>
41     </div>
42 </div>-----</div>
```

#### d. Result

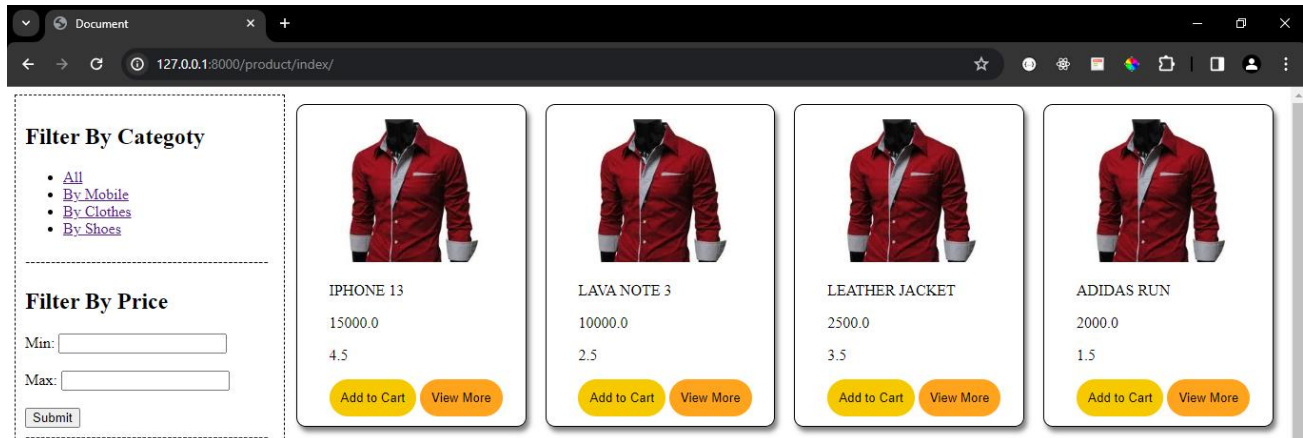


#### 4. Implementing filter by rating logic

- a. Since we are not displaying ratings in index.html, let's display first

```
div class="card-items">
  
  <div class="card-text">
    <p>{{product.name|upper}}</p>
    <p>{{product.price}}</p>
    <p>{{product.rating}}</p>
    <button id="add_to_cart_btn"><a href="">Add to Cart</a></button>
    <button id="buy_now_btn"><a href="">View More</a></button>
  </div>
```

- b. Result



- c. Create view

```
views.py
product > views.py > filter_by_price_range
35
36 def filter_by_rating(request, rating_value):
37     #select * from product where is_active=True and category=category_value;
38     #ProductTable.objects.filter(is_active=True, category=category_value)
39     #from django.db.models import Q
40     data={}
41     q1 = Q(is_active=True)
42     q2 = Q(rating__gt=rating_value)
43     filtered_products=ProductTable.objects.filter(q1 & q2)
44     data['products']=filtered_products
45     return render(request, 'product/index.html', context=data)
46
```

- d. Create url for view

```
product_urls.py
product > product_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('index/', views.index),
6     path('filter/<category_value>', views.filter_by_category),
7     path('sort/<sort_value>', views.sort_by_price),
8     path('rating/<rating_value>', views.filter_by_rating),
9 ]
```

- e. Use url in index.html

```
index.html X
templates > product > index.html
42 <div>-----</div>
43 <div class="by_rating">
44   <h2>Sort By Rating</h2>
45   <ul>
46     <li><a href="/product/rating/3">3 and above</a></li>
47     <li><a href="/product/rating/4">4 and above</a></li>
48   </ul>
49 </div>
50 <div>-----</div>
```

## 5. Filter by price range

- a. Create view

```
views.py X
product > views.py > filter_by_price_range
47
48 def filter_by_price_range(request):
49     data={}
50     min = request.GET['min']
51     max = request.GET['max']
52     q1 = Q(price__gte=min)
53     q2 = Q(price__lte=max)
54     q3 = Q(is_active=True)
55     filtered_products=ProductTable.objects.filter(q1 & q2 & q3)
56     data['products']=filtered_products
57     return render(request,'product/index.html',context=data)
```

- b. Create url for view

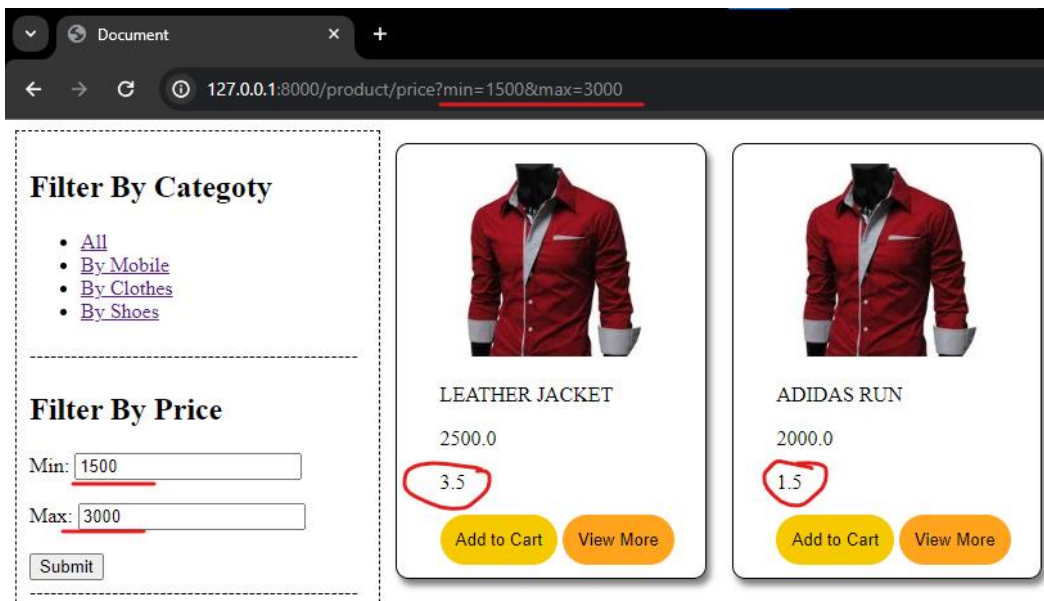
```
product_urls.py X
product > product_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('index/', views.index),
6     path('filter/<category_value>', views.filter_by_category),
7     path('sort/<sort_value>', views.sort_by_price),
8     path('rating/<rating_value>', views.filter_by_rating),
9     path('price', views.filter_by_price_range),
10 ]
```

IMPORTANT : there is no / after price

- c. User url in index.html

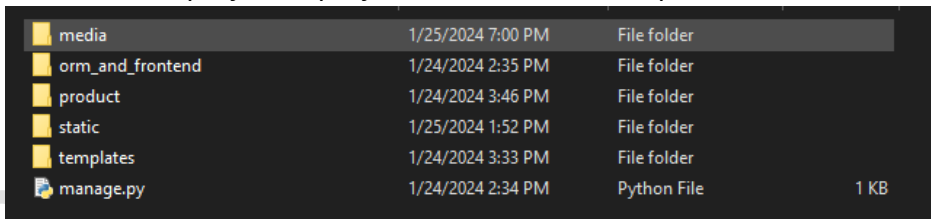
```
index.html X
templates > product > index.html
23 <div>-----</div>
24 <div class="by_price">
25   <h2>Filter By Price</h2>
26   <form action="/product/price">
27     <label for="">Min:</label>
28     <input type="number" name="min"> <br><br>
29     <label for="">Max:</label>
30     <input type="number" name="max"> <br><br>
31     <input type="submit">
32   </form>
33 </div>
34 <div>-----</div>
```

Result :

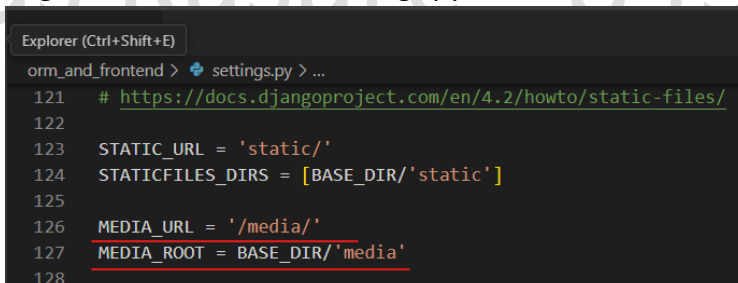


## 6. Uploading Image

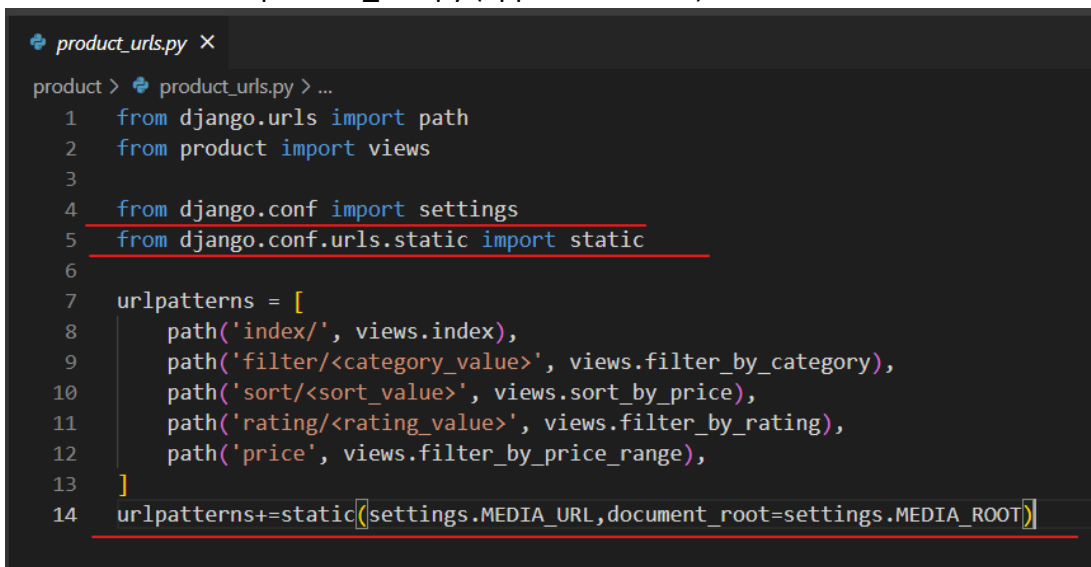
- Create media project in project like we create templates folder



- Register media folder in settings.py



- Create media url in product\_urls.py (application level)



- d. Since we are not having image field in our Product model, so lets add image field in it

```
models.py X
product > models.py > ProductTable > __str__
1 from django.db import models
2
3 # Create your models here.
4 class ProductTable(models.Model):
5     CATEGORIES = ((1,'Mobile'),(2,'Clothes'),(3,'Shoes'))
6     name = models.CharField(max_length=50)
7     price = models.FloatField()
8     details=models.CharField(max_length=150)
9     category = models.IntegerField(choices=CATEGORIES)
10    is_active= models.BooleanField()
11    rating = models.FloatField()
12    image=models.ImageField(upload_to='image')
13
14    def __str__(self) :
15        return self.name + " added to table"
```

- e. Makemigrations

```
C:\TEJAS KASARE (Very imp folder)\my notes\Django\orm_and_frontend>python manage.py makemigrations
It is impossible to add a non-nullable field 'image' to producttable without specifying a default. This
is because the database needs something to populate existing rows.
Please select a fix:
1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
2) Quit and manually define a default value in models.py.
Select an option: 1
Please enter the default value as valid Python.
The datetime and django.utils.timezone modules are available, so it is possible to provide e.g. timezon
e.now as a value.
Type 'exit' to exit this prompt
>>> 0
Migrations for 'product':
  product\migrations\0003_producttable_image.py
    - Add field image to producttable

C:\TEJAS KASARE (Very imp folder)\my notes\Django\orm_and_frontend>
```

After running makemigrations, you will get this error because

- f. Migrate  
g. Add "image" in list\_display in admin.py

```
admin.py X
product > admin.py > ...
1 from django.contrib import admin
2 from product.models import ProductTable
3
4 # Register your models here.
5 class ProductAdmin(admin.ModelAdmin):
6     list_display = ['id','name','price','details','category','is_active','rating','image']
7
8     admin.site.register(ProductTable,ProductAdmin)
```

- h. Runserver

- i. Check table in workbench, you will get image filed with default value 0

[illegible]

- j. Download one shoe image (200x200) and add one shoe product with image


The screenshot shows the Django administration interface for adding a new product table. The browser address bar indicates the URL is `127.0.0.1:8000/admin/product/producttable/add/`. The left sidebar contains a navigation menu with the following sections:

- AUTHENTICATION AND AUTHORIZATION**
  - Groups [+ Add](#)
  - Users [+ Add](#)
- PRODUCT**
  - Product tables [+ Add](#)

The main content area displays the "Add product table" form with the following fields:

- Name:**
- Price:**
- Details:**
- Category:**  (dropdown menu)
- Is active:** ☒
- Rating:**
- Image:**  shoe1.jpg







At the bottom of the form, there are three buttons: **SAVE**, **Save and add another**, and **Save and continue editing**.

 The product table "nike fast x added to table" was added successfully.

Select product table to change

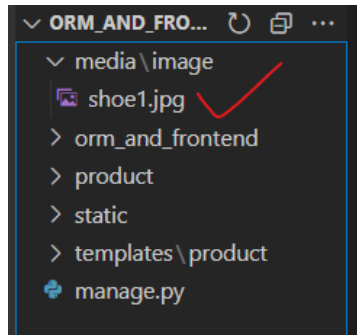
ADD PRODUCT TABLE +

Action: ----- Go 0 of 6 selected

<input type="checkbox"/>	ID	NAME	PRICE	DETAILS	CATEGORY	IS ACTIVE	RATING	IMAGE
<input type="checkbox"/>	6	nike fast x	3999.0	fine shoes for outdoor running	Shoes		4.5	<a href="image/shoe1.jpg">image/shoe1.jpg</a>
<input type="checkbox"/>	5	slim shirt	999.0	slim fit shirt	Clothes		4.8	0
<input type="checkbox"/>	4	adidas run	2000.0	fine shoes for outdoor running	Shoes		1.5	0
<input type="checkbox"/>	3	leather jacket	2500.0	best for winter	Clothes		3.5	0
<input type="checkbox"/>	2	lava note 3	10000.0	under budget phone	Mobile		2.5	0
<input type="checkbox"/>	1	iphone 13	15000.0	good phone	Mobile		4.5	0

6 product tables

- k. Check in media folder –
  - i. You will get image folder
  - ii. In image folder you will get your uploaded image



- l. Add image for other products
- m. Show images dynamically in index.html for each product(we will do it later)

7. DONE

Tejas Kasare - 8459859415