University of Windsor

School of Computer Science

Masters in Applied Computing (M.A.C)

COMP8347

Internet Applications and Distributed Systems

Professor Dr Adel Abusitta

Lab Assignment

by

Hamza Baig (110089314)

Harsimran Singh (110090200)

Vivek Verma (110094354)

Talha Muhammad Shamoon Chaudhry (110087321)

Vidya Sehgal (110090510)

Komalben Kishorbhai Patel (110091595)

Abhishek Singh (110089745)

Table of Contents

I - The new model **OrderVehicle** would look like: 4
1. Vehicle serving as a foreign key: 4
2. Buyer serving as a foreign key: 4
3. Field that indicates number of vehicles being ordered: 4
4. Field that indicates status: 4
5. Field that indicates date order was updated: 4

II – Add optional field in vehicles that describes product in few words: 4
III – Add a phone number field in buyer: 4
IV – Update default area of Buyer to Chatham: 4
V- Remove fullname attribute from Buyer: 4
VI – Add __str__ method in all models: 5
VII – Write a method in OrderVehicle that returns the total price for all vehicles: 6
Answer 1b: 7
I – Admin.py 7
II and III – Admin: 7
IV - 7
Answer 2: 10
a. List the buyers having last name ‘Smith’: 10
b. List the buyers whose addresses start with ‘444’: 10
c. List the buyers who live on a ‘street’ in Windsor area: 10
d. List the buyers who live in Chatham and Toronto: 10
e. List the buyers who do not live in Windsor: 10
f. List the buyers who are interested in CarType ‘Toyota’: 10
g. List the vehicles that cost less than $30000: 10
h. List the vehicles which are not available at the moment: 10
i. List all the cartypes in which a buyer with username lara is interested in: 10
j. List all the vehicles with a car_price > $25000 and inventory <10: 10
k. Get the first name of the buyer of the order having primary key (or pk) equal to 2: 10
l. Write a query that first stores all cartypes in a variable called ‘all_cartypes’, then calculates the length of ‘all_cartypes’, and displays the third element of ‘all_cartypes’: 11
m. Write a query with a ‘for loop’ and a ‘print statement’ in it. You can choose any model (CarType, Vehicle, Buyer, OrderVehicle) you like: 11
Answer 3: 11
Model Description with title, text, and created_at: 11
Perform Migrations: 12

Admin.py: 12

Add data for description table: 12

Python Console: 12

1. Get the first description from the description model (this answer should return a query set): 13

2. Get the title of the first description Get the first description (this answer should return the text): 13

3. Query all the database objects: 13

4. From the Python Console, create a new description with a title and a description: 13

5. Filter the description title based on the starting letters (ex. “this”): 13

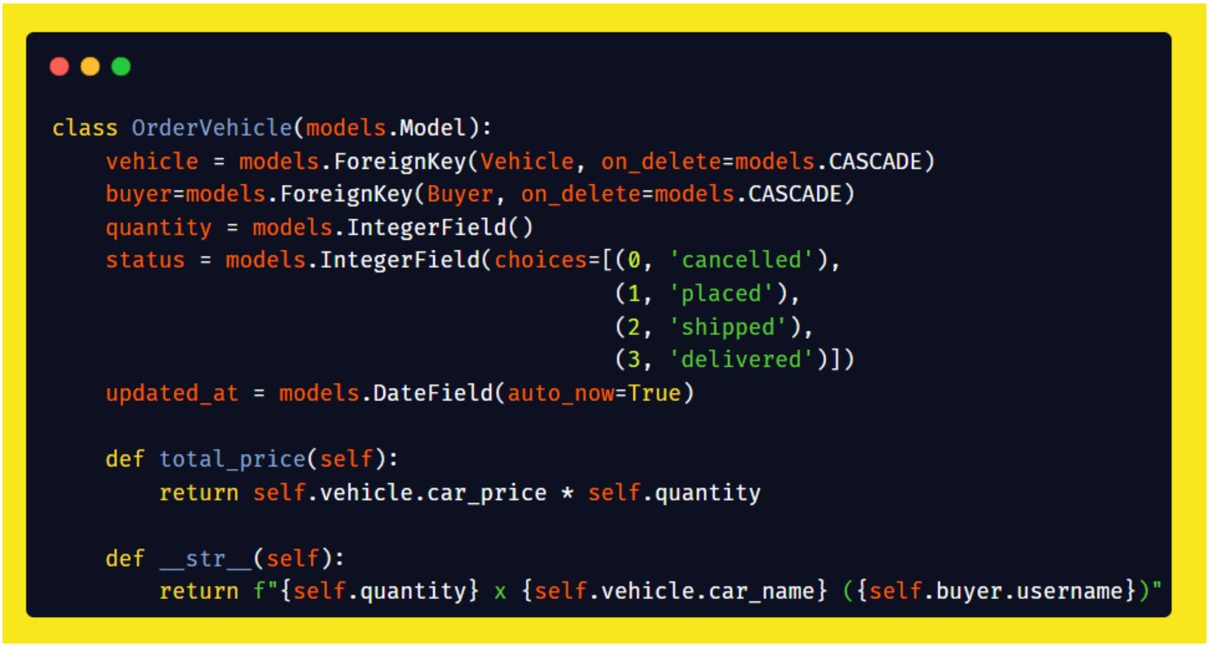
6. Filter the description that contains any word (ex. “Django”): 13

7. Filter the description that does not contain any word (ex. “Django”): 13

8. Filter the description that contains any word (ex. “Django”) but the title not having the same word: 14

Answer 1a

I - The new model **OrderVehicle** would look like:



```
class OrderVehicle(models.Model):
    vehicle = models.ForeignKey(Vehicle, on_delete=models.CASCADE)
    buyer=models.ForeignKey(Buyer, on_delete=models.CASCADE)
    quantity = models.IntegerField()
    status = models.IntegerField(choices=[(0, 'cancelled'),
                                          (1, 'placed'),
                                          (2, 'shipped'),
                                          (3, 'delivered')])
    updated_at = models.DateField(auto_now=True)

    def total_price(self):
        return self.vehicle.car_price * self.quantity

    def __str__(self):
        return f"{self.quantity} x {self.vehicle.car_name} ({self.buyer.username})"
```

1. Vehicle serving as a foreign key:
   vehicle = models.ForeignKey(Vehicle, on_delete=models.CASCADE)

2. Buyer serving as a foreign key:
   buyer=models.ForeignKey(Buyer, on_delete=models.CASCADE)

3. Field that indicates number of vehicles being ordered:
   quantity = models.IntegerField()

4. Field that indicates status:
   status = models.IntegerField(choices=[(0, 'cancelled'), (1, 'placed'), (2, 'shipped'), (3, 'delivered')])

5. Field that indicates date order was updated:
   updated_at = models.DateField(auto_now=True)

II – Add optional field in vehicles that describes product in few words:

description = models.TextField(blank=True)

III – Add a phone number field in buyer:

phone_number = models.CharField(max_length=20, null=True,blank=True)

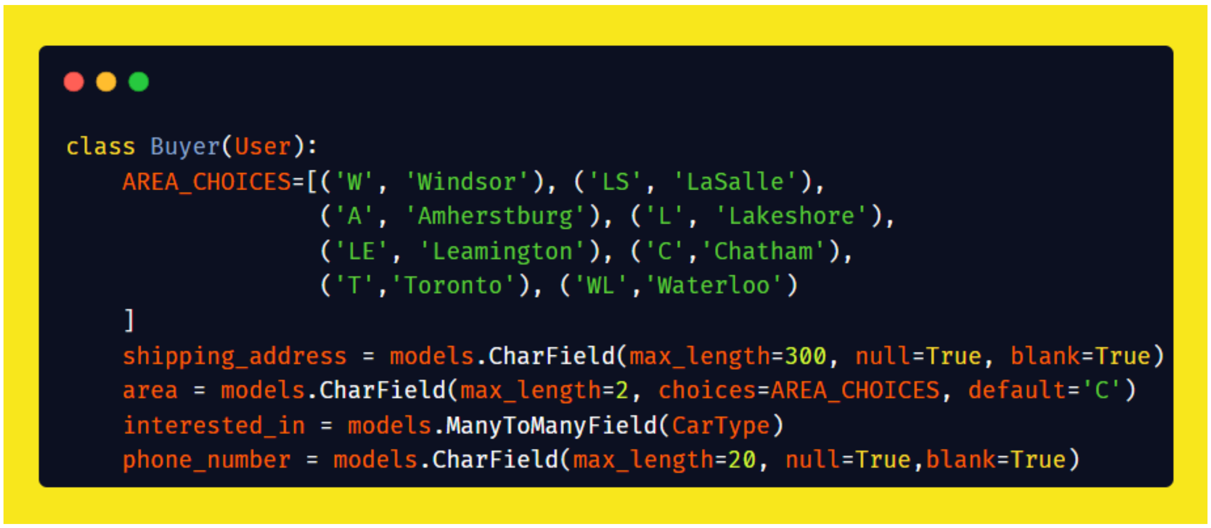
IV – Update default area of Buyer to Chatham:

AREA_CHOICES=[('W', 'Windsor'), ('LS', 'LaSalle'), ('A', 'Amherstburg'), ('L', 'Lakeshore'), ('LE', 'Leamington'), ('C','Chatham'), ('T','Toronto'), ('WL','Waterloo')]

area = models.CharField(max_length=2, choices=AREA_CHOICES, default='C')

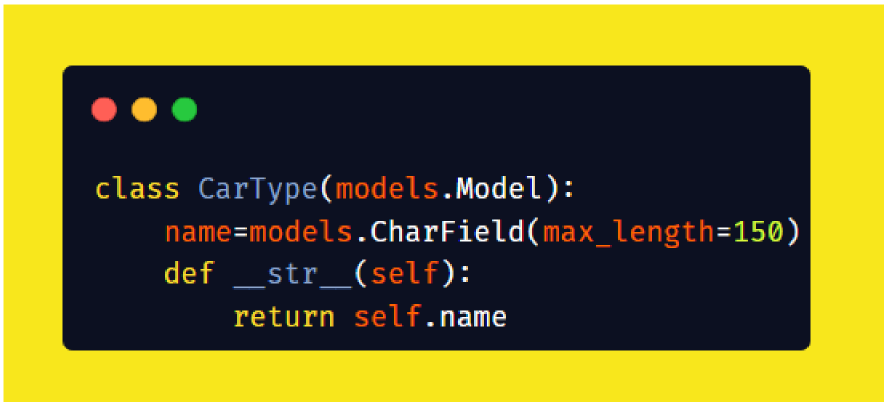
V- Remove fullname attribute from Buyer:

Updated Buyer

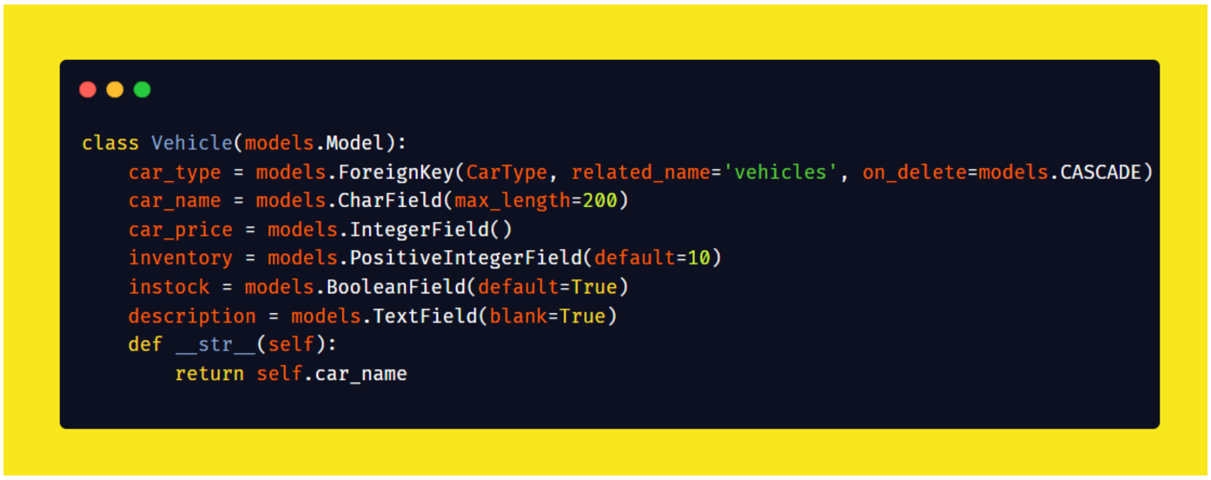


```
class Buyer(User):
    AREA_CHOICES=[('W', 'Windsor'), ('LS', 'LaSalle'),
                  ('A', 'Amherstburg'), ('L', 'Lakeshore'),
                  ('LE', 'Leamington'), ('C','Chatham'),
                  ('T','Toronto'), ('WL','Waterloo')
    ]
    shipping_address = models.CharField(max_length=300, null=True, blank=True)
    area = models.CharField(max_length=2, choices=AREA_CHOICES, default='C')
    interested_in = models.ManyToManyField(CarType)
    phone_number = models.CharField(max_length=20, null=True,blank=True)
```

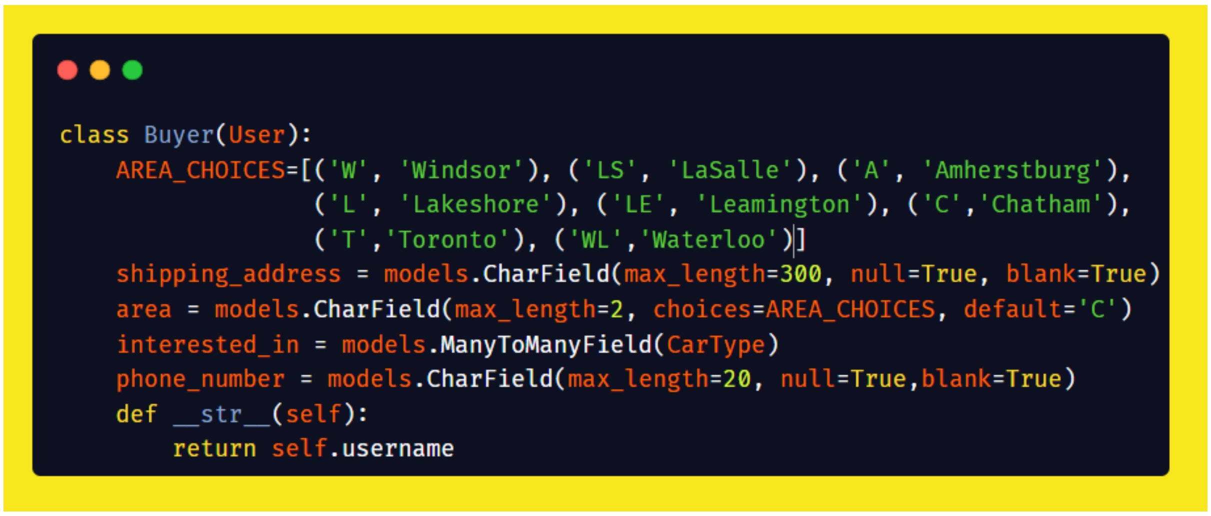
VI – Add __str__ method in all models:



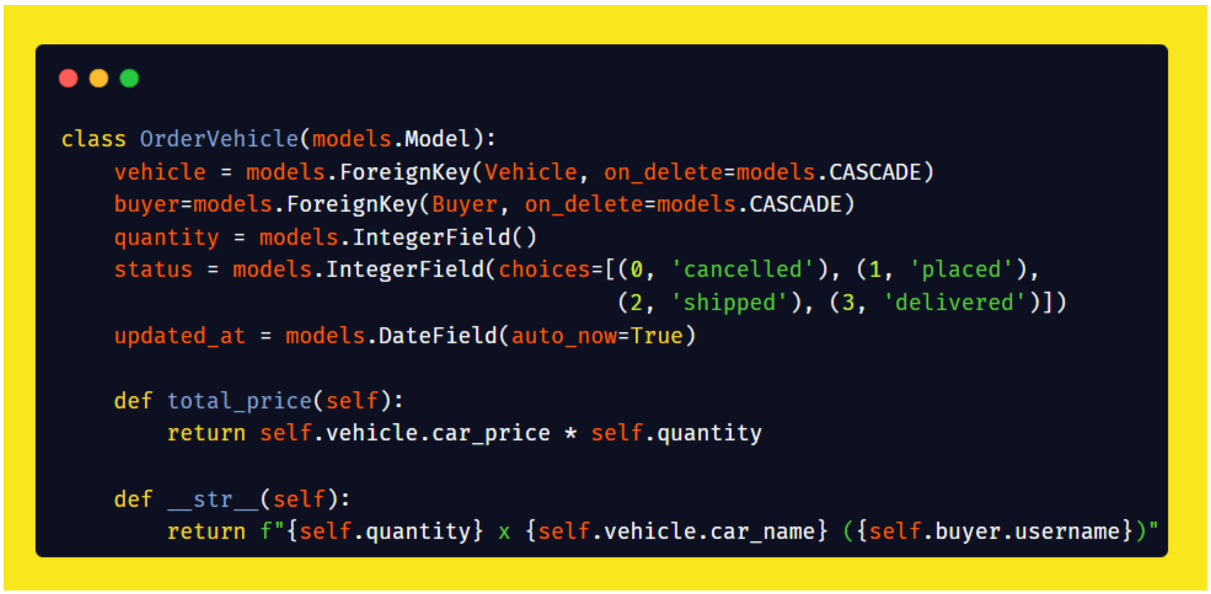
```
class CarType(models.Model):
    name=models.CharField(max_length=150)
    def __str__(self):
        return self.name
```



```
class Vehicle(models.Model):
    car_type = models.ForeignKey(CarType, related_name='vehicles', on_delete=models.CASCADE)
    car_name = models.CharField(max_length=200)
    car_price = models.IntegerField()
    inventory = models.PositiveIntegerField(default=10)
    instock = models.BooleanField(default=True)
    description = models.TextField(blank=True)
    def __str__(self):
        return self.car_name
```



```python
class Buyer(User):
    AREA_CHOICES=[('W', 'Windsor'), ('LS', 'LaSalle'), ('A', 'Amherstburg'),
                  ('L', 'Lakeshore'), ('LE', 'Leamington'), ('C','Chatham'),
                  ('T','Toronto'), ('WL','Waterloo')]
    shipping_address = models.CharField(max_length=300, null=True, blank=True)
    area = models.CharField(max_length=2, choices=AREA_CHOICES, default='C')
    interested_in = models.ManyToManyField(CarType)
    phone_number = models.CharField(max_length=20, null=True,blank=True)
    def __str__(self):
        return self.username
```



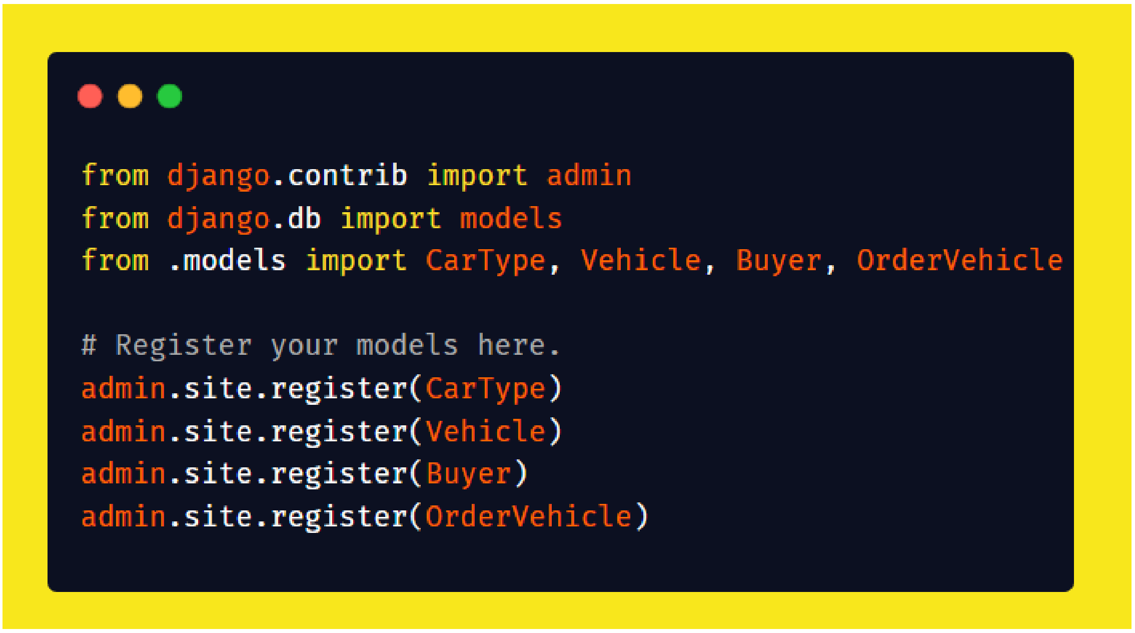
```python
class OrderVehicle(models.Model):
    vehicle = models.ForeignKey(Vehicle, on_delete=models.CASCADE)
    buyer=models.ForeignKey(Buyer, on_delete=models.CASCADE)
    quantity = models.IntegerField()
    status = models.IntegerField(choices=[(0, 'cancelled'), (1, 'placed'),
                                          (2, 'shipped'), (3, 'delivered')])
    updated_at = models.DateField(auto_now=True)

    def total_price(self):
        return self.vehicle.car_price * self.quantity

    def __str__(self):
        return f"{self.quantity} x {self.vehicle.car_name} ({self.buyer.username})"
```

VII – Write a method in OrderVehicle that returns the total price for all vehicles:

```python
def total_price(self):
    return self.vehicle.car_price * self.quantity
```

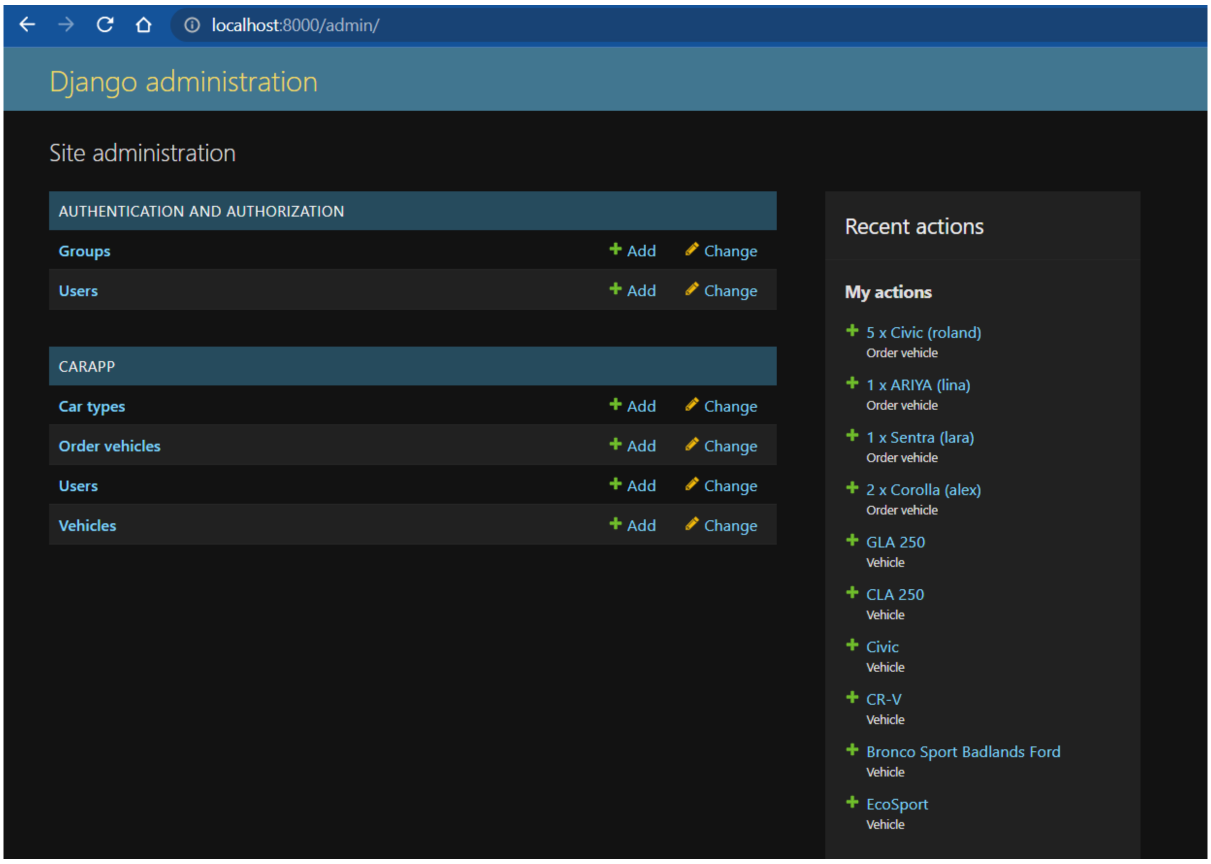
Answer 1b:

I – Admin.py



```
from django.contrib import admin
from django.db import models
from .models import CarType, Vehicle, Buyer, OrderVehicle

# Register your models here.
admin.site.register(CarType)
admin.site.register(Vehicle)
admin.site.register(Buyer)
admin.site.register(OrderVehicle)
```

II and III – Admin:



localhost:8000/admin/

Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	Add	Change
Users	Add	Change

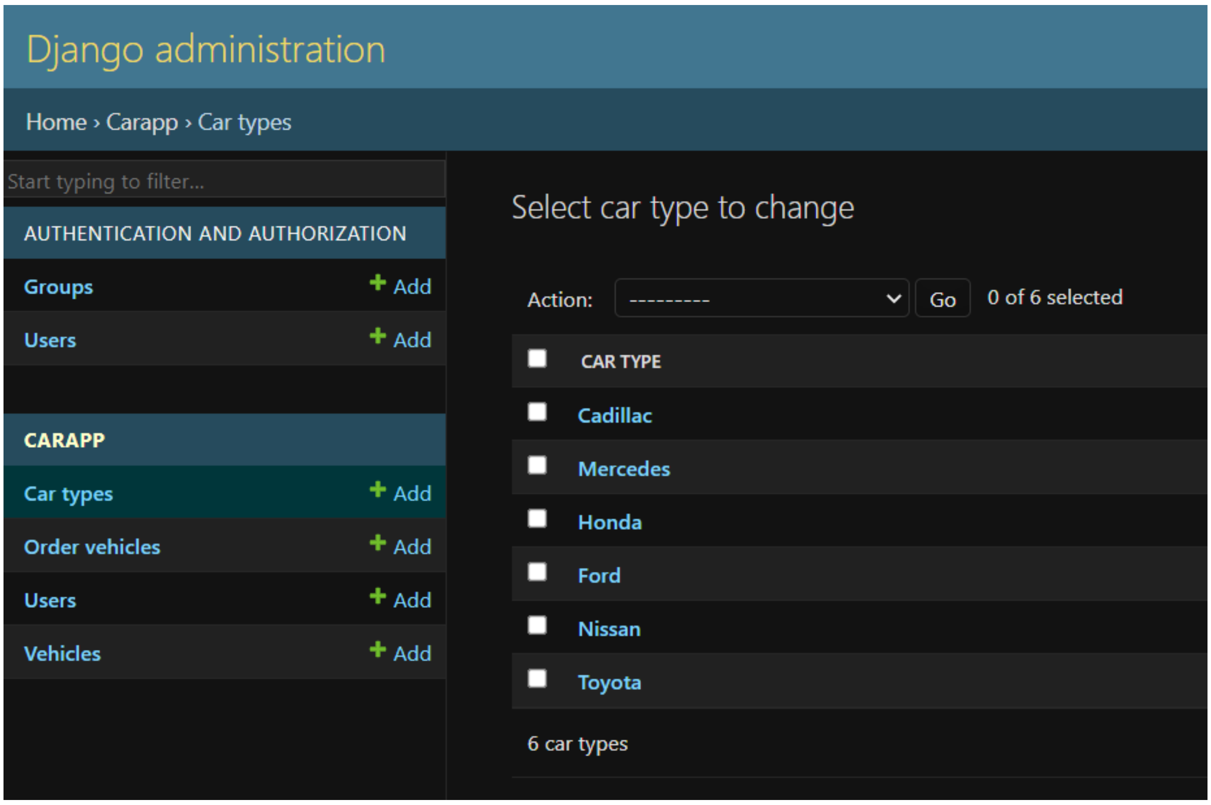
CARAPP		
Car types	Add	Change
Order vehicles	Add	Change
Users	Add	Change
Vehicles	Add	Change

Recent actions

My actions

- 5 x Civic (roland) — Order vehicle
- 1 x ARIYA (lina) — Order vehicle
- 1 x Sentra (lara) — Order vehicle
- 2 x Corolla (alex) — Order vehicle
- GLA 250 — Vehicle
- CLA 250 — Vehicle
- Civic — Vehicle
- CR-V — Vehicle
- Bronco Sport Badlands Ford — Vehicle
- EcoSport — Vehicle

IV -


Django administration

Home › Carapp › Car types

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

CARAPP

Car types + Add

Order vehicles + Add

Users + Add

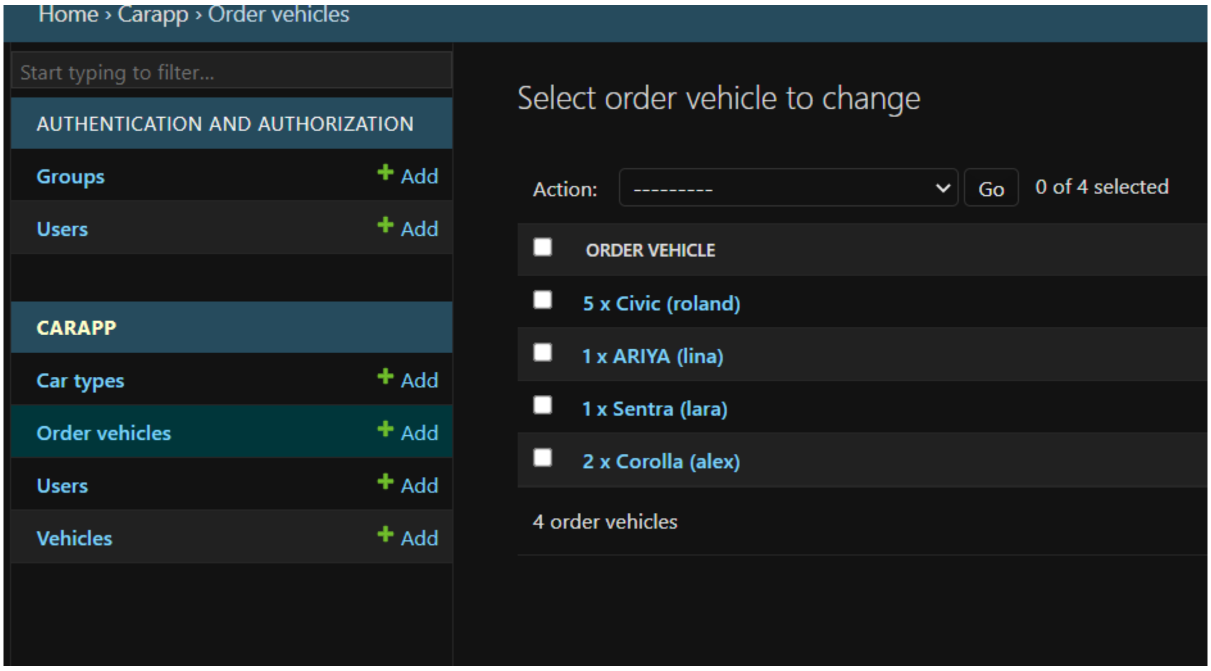
Vehicles + Add

Select car type to change

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

CAR TYPE
Cadillac
Mercedes
Honda
Ford
Nissan
Toyota

6 car types


Home › Carapp › Order vehicles

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

CARAPP

Car types + Add

Order vehicles + Add

Users + Add

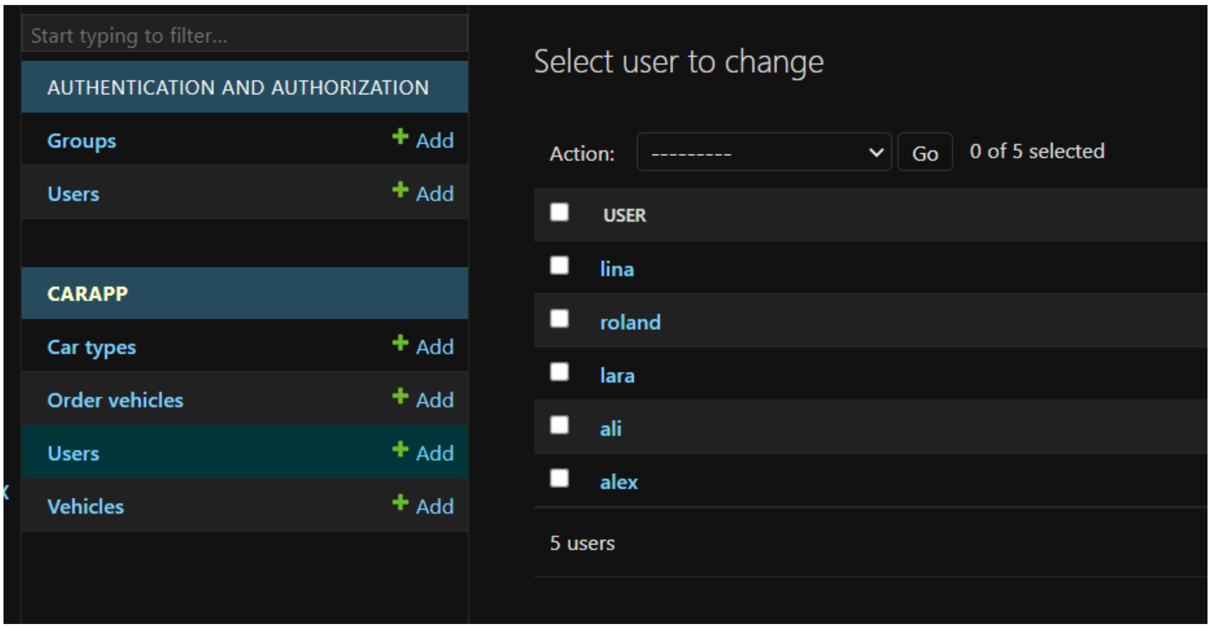
Vehicles + Add

Select order vehicle to change

Action: --------- Go 0 of 4 selected

ORDER VEHICLE
5 x Civic (roland)
1 x ARIYA (lina)
1 x Sentra (lara)
2 x Corolla (alex)

4 order vehicles

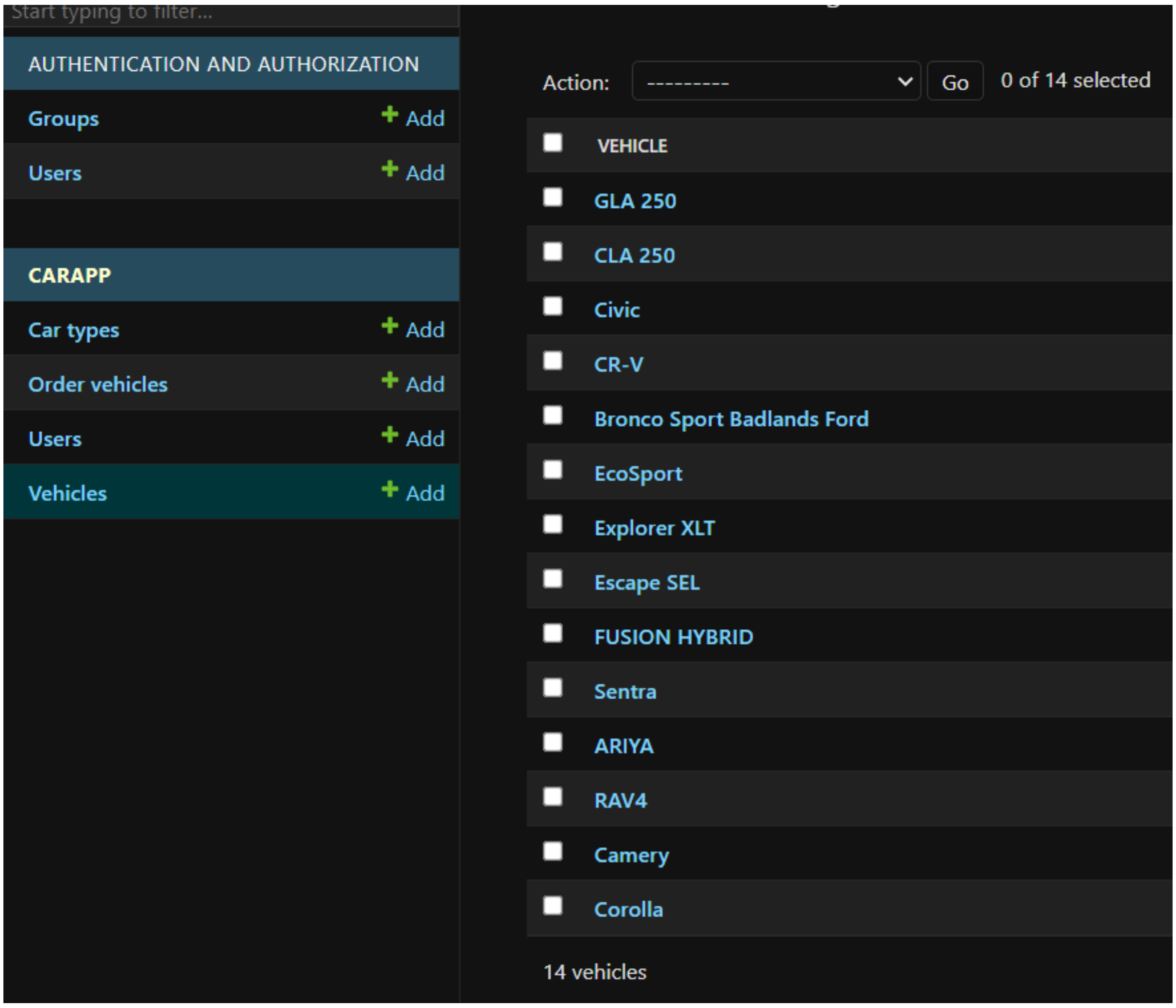


Select user to change

Action: --------- Go 0 of 5 selected

USER
lina
roland
lara
ali
alex

5 users



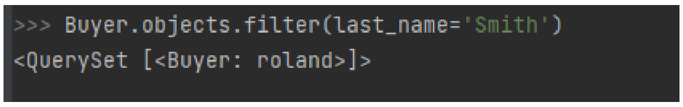
Action: --------- Go 0 of 14 selected

VEHICLE
GLA 250
CLA 250
Civic
CR-V
Bronco Sport Badlands Ford
EcoSport
Explorer XLT
Escape SEL
FUSION HYBRID
Sentra
ARIYA
RAV4
Camery
Corolla

14 vehicles

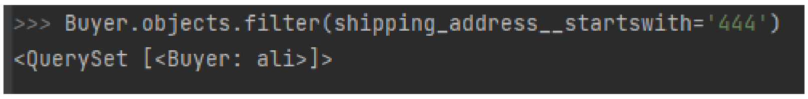
Answer 2:

a. List the buyers having last name ‘Smith’:



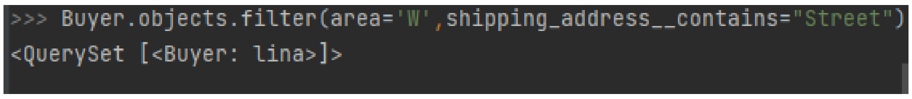
```
>>> Buyer.objects.filter(last_name='Smith')
<QuerySet [<Buyer: roland>]>
```

b. List the buyers whose addresses start with ‘444’:



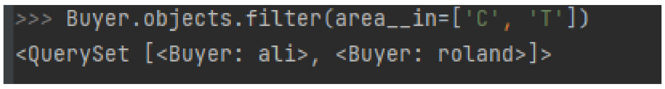
```
>>> Buyer.objects.filter(shipping_address__startswith='444')
<QuerySet [<Buyer: ali>]>
```

c. List the buyers who live on a ‘street’ in Windsor area:



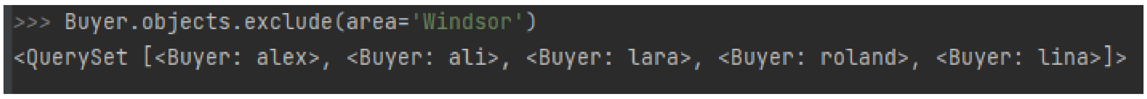
```
>>> Buyer.objects.filter(area='W',shipping_address__contains="Street")
<QuerySet [<Buyer: lina>]>
```

d. List the buyers who live in Chatham and Toronto:



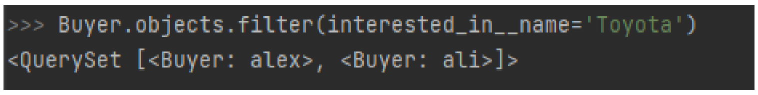
```
>>> Buyer.objects.filter(area__in=['C', 'T'])
<QuerySet [<Buyer: ali>, <Buyer: roland>]>
```

e. List the buyers who do not live in Windsor:



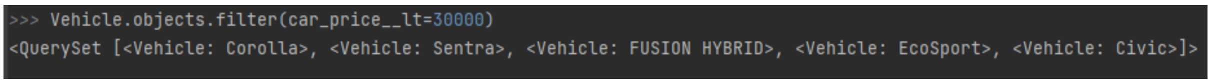
```
>>> Buyer.objects.exclude(area='Windsor')
<QuerySet [<Buyer: alex>, <Buyer: ali>, <Buyer: lara>, <Buyer: roland>, <Buyer: lina>]>
```

f. List the buyers who are interested in CarType ‘Toyota’:



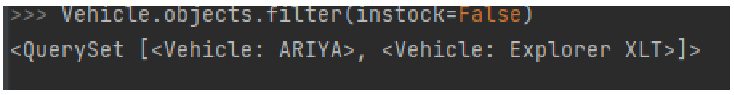
```
>>> Buyer.objects.filter(interested_in__name='Toyota')
<QuerySet [<Buyer: alex>, <Buyer: ali>]>
```

g. List the vehicles that cost less than $30000:



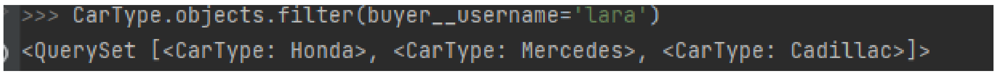
```
>>> Vehicle.objects.filter(car_price__lt=30000)
<QuerySet [<Vehicle: Corolla>, <Vehicle: Sentra>, <Vehicle: FUSION HYBRID>, <Vehicle: EcoSport>, <Vehicle: Civic>]>
```

h. List the vehicles which are not available at the moment:



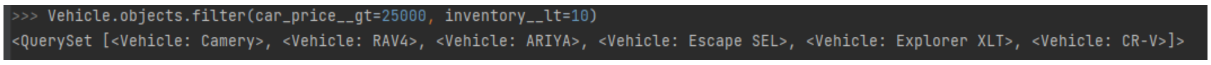
```
>>> Vehicle.objects.filter(instock=False)
<QuerySet [<Vehicle: ARIYA>, <Vehicle: Explorer XLT>]>
```

i. List all the cartypes in which a buyer with username lara is interested in:



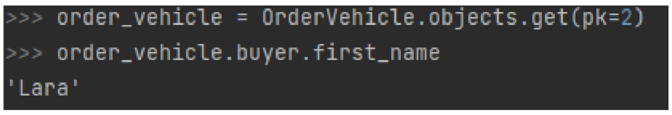
```
>>> CarType.objects.filter(buyer__username='lara')
<QuerySet [<CarType: Honda>, <CarType: Mercedes>, <CarType: Cadillac>]>
```

j. List all the vehicles with a car_price > $25000 and inventory <10:



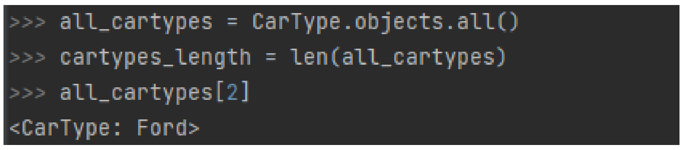
```
>>> Vehicle.objects.filter(car_price__gt=25000, inventory__lt=10)
<QuerySet [<Vehicle: Camery>, <Vehicle: RAV4>, <Vehicle: ARIYA>, <Vehicle: Escape SEL>, <Vehicle: Explorer XLT>, <Vehicle: CR-V>]>
```

k. Get the first name of the buyer of the order having primary key (or pk) equal to 2:



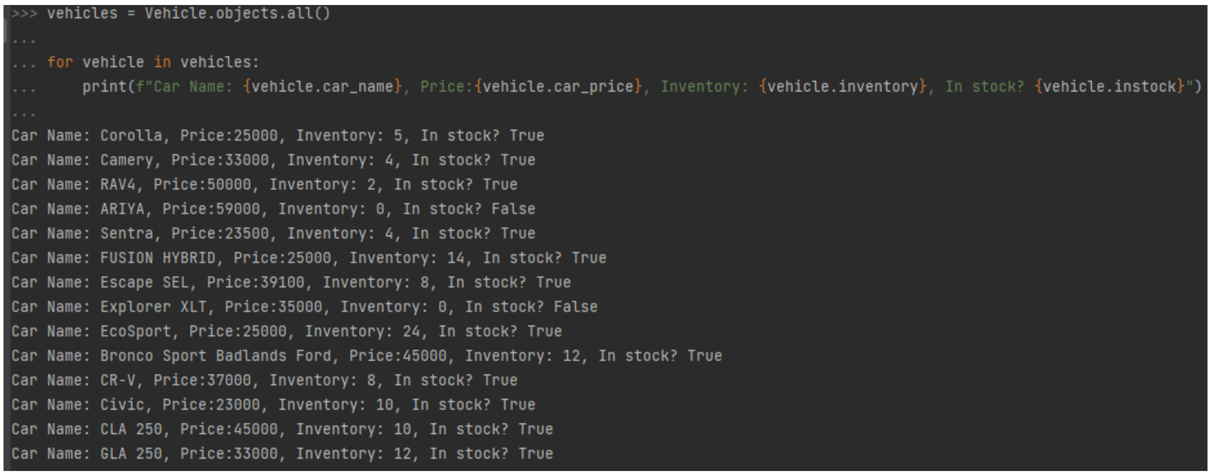
```
>>> order_vehicle = OrderVehicle.objects.get(pk=2)
>>> order_vehicle.buyer.first_name
'Lara'
```

l. Write a query that first stores all cartypes in a variable called 'all_cartypes', then calculates the length of 'all_cartypes', and displays the third element of 'all_cartypes':



```
>>> all_cartypes = CarType.objects.all()
>>> cartypes_length = len(all_cartypes)
>>> all_cartypes[2]
<CarType: Ford>
```

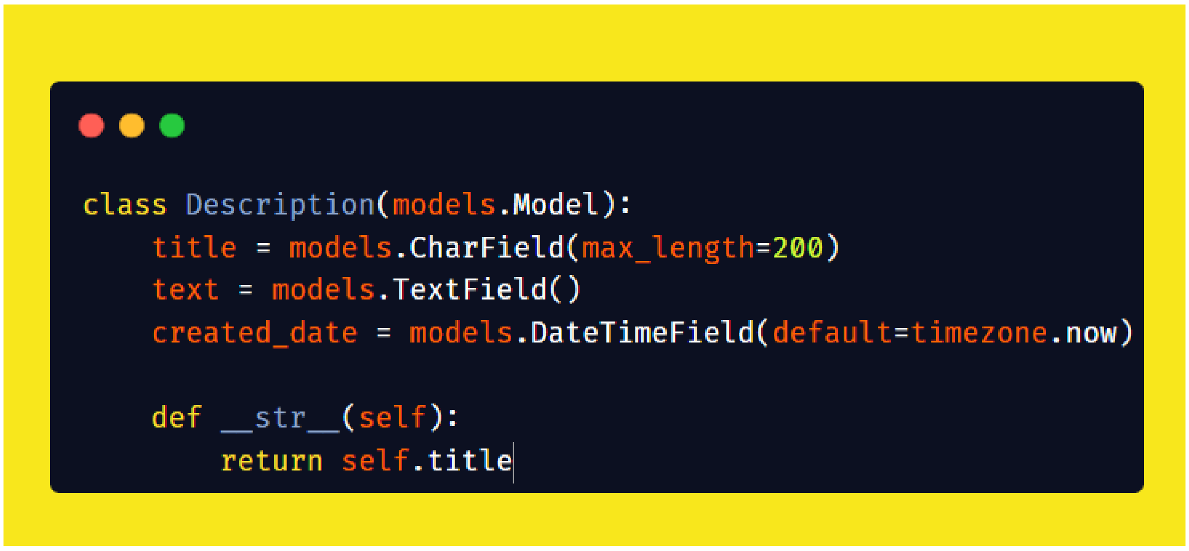
m. Write a query with a 'for loop' and a 'print statement' in it. You can choose any model (CarType, Vehicle, Buyer, OrderVehicle) you like:



```
>>> vehicles = Vehicle.objects.all()
...
... for vehicle in vehicles:
...     print(f"Car Name: {vehicle.car_name}, Price:{vehicle.car_price}, Inventory: {vehicle.inventory}, In stock? {vehicle.instock}")
...
Car Name: Corolla, Price:25000, Inventory: 5, In stock? True
Car Name: Camery, Price:33000, Inventory: 4, In stock? True
Car Name: RAV4, Price:50000, Inventory: 2, In stock? True
Car Name: ARIYA, Price:59000, Inventory: 0, In stock? False
Car Name: Sentra, Price:23500, Inventory: 4, In stock? True
Car Name: FUSION HYBRID, Price:25000, Inventory: 14, In stock? True
Car Name: Escape SEL, Price:39100, Inventory: 8, In stock? True
Car Name: Explorer XLT, Price:35000, Inventory: 0, In stock? False
Car Name: EcoSport, Price:25000, Inventory: 24, In stock? True
Car Name: Bronco Sport Badlands Ford, Price:45000, Inventory: 12, In stock? True
Car Name: CR-V, Price:37000, Inventory: 8, In stock? True
Car Name: Civic, Price:23000, Inventory: 10, In stock? True
Car Name: CLA 250, Price:45000, Inventory: 10, In stock? True
Car Name: GLA 250, Price:33000, Inventory: 12, In stock? True
```

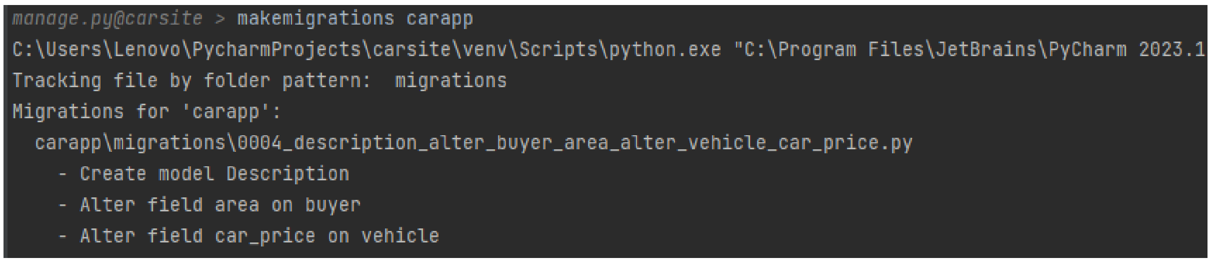
Answer 3:

Model Description with title, text, and created_at:

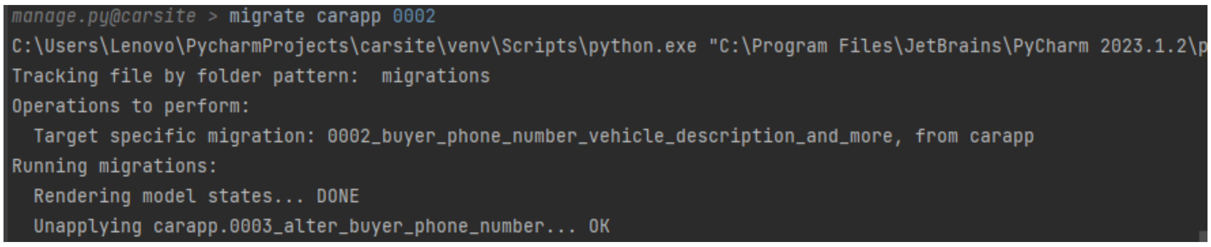


```python
class Description(models.Model):
    title = models.CharField(max_length=200)
    text = models.TextField()
    created_date = models.DateTimeField(default=timezone.now)

    def __str__(self):
        return self.title
```

Perform Migrations:

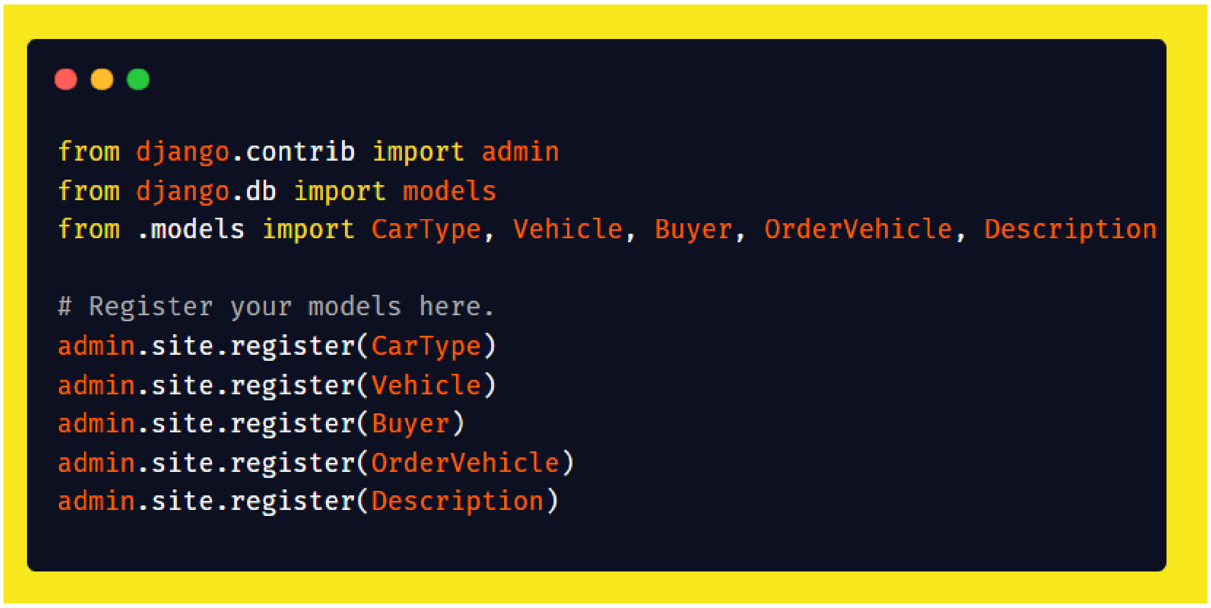


```
manage.py@carsite > makemigrations carapp
C:\Users\Lenovo\PycharmProjects\carsite\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2023.1
Tracking file by folder pattern:  migrations
Migrations for 'carapp':
  carapp\migrations\0004_description_alter_buyer_area_alter_vehicle_car_price.py
    - Create model Description
    - Alter field area on buyer
    - Alter field car_price on vehicle
```



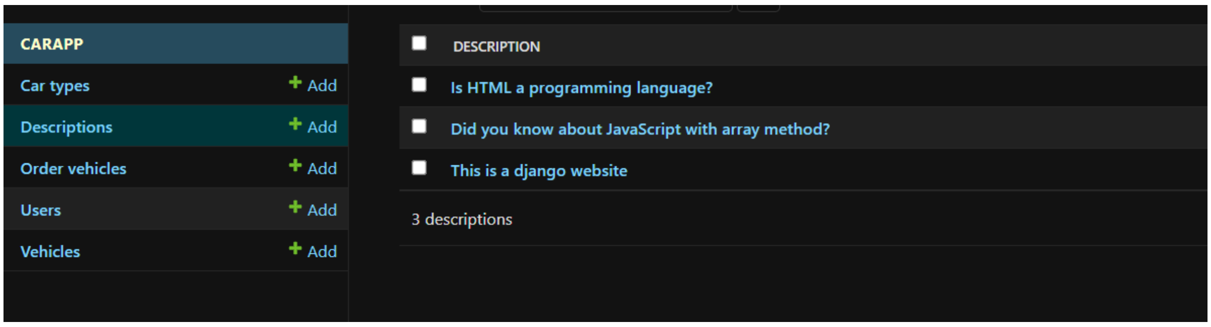
```
manage.py@carsite > migrate carapp 0002
C:\Users\Lenovo\PycharmProjects\carsite\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2023.1.2\p
Tracking file by folder pattern:  migrations
Operations to perform:
  Target specific migration: 0002_buyer_phone_number_vehicle_description_and_more, from carapp
Running migrations:
  Rendering model states... DONE
  Unapplying carapp.0003_alter_buyer_phone_number... OK
```

Admin.py:



```python
from django.contrib import admin
from django.db import models
from .models import CarType, Vehicle, Buyer, OrderVehicle, Description

# Register your models here.
admin.site.register(CarType)
admin.site.register(Vehicle)
admin.site.register(Buyer)
admin.site.register(OrderVehicle)
admin.site.register(Description)
```

Add data for description table:

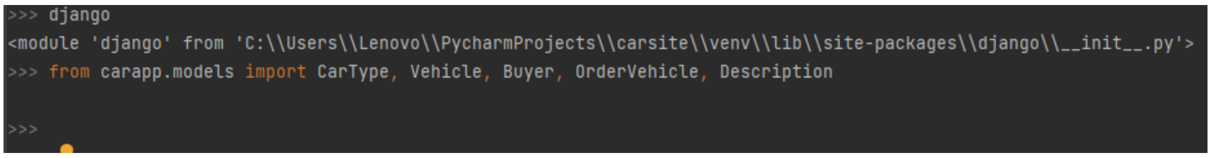


CARAPP	
Car types	+ Add
Descriptions	+ Add
Order vehicles	+ Add
Users	+ Add
Vehicles	+ Add

DESCRIPTION
Is HTML a programming language?
Did you know about JavaScript with array method?
This is a django website

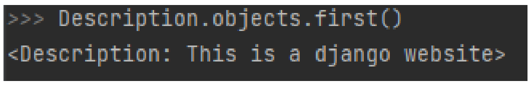
3 descriptions

Python Console:



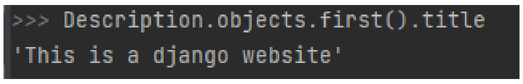
```
>>> django
<module 'django' from 'C:\\Users\\Lenovo\\PycharmProjects\\carsite\\venv\\lib\\site-packages\\django\\__init__.py'>
>>> from carapp.models import CarType, Vehicle, Buyer, OrderVehicle, Description

>>>
```

1. Get the first description from the description model (this answer should return a query set):

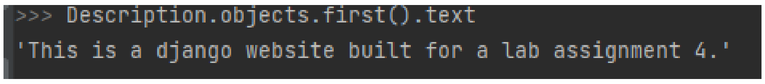


```
>>> Description.objects.first()
<Description: This is a django website>
```

2. Get the title of the first description Get the first description (this answer should return the text):

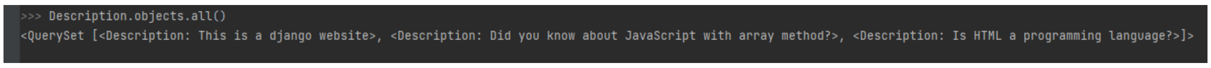


```
>>> Description.objects.first().title
'This is a django website'
```



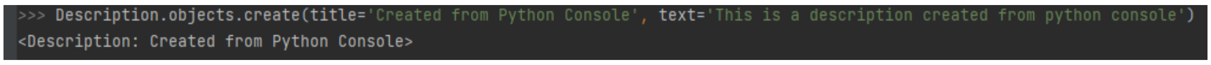
```
>>> Description.objects.first().text
'This is a django website built for a lab assignment 4.'
```

3. Query all the database objects:



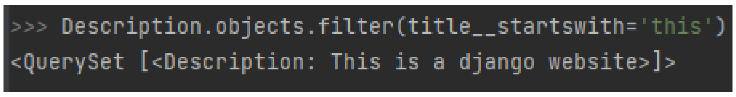
```
>>> Description.objects.all()
<QuerySet [<Description: This is a django website>, <Description: Did you know about JavaScript with array method?>, <Description: Is HTML a programming language?>]>
```

4. From the Python Console, create a new description with a title and a description:



```
>>> Description.objects.create(title='Created from Python Console', text='This is a description created from python console')
<Description: Created from Python Console>
```

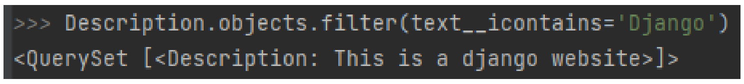
5. Filter the description title based on the starting letters (ex. “this”):



```
>>> Description.objects.filter(title__startswith='this')
<QuerySet [<Description: This is a django website>]>
```

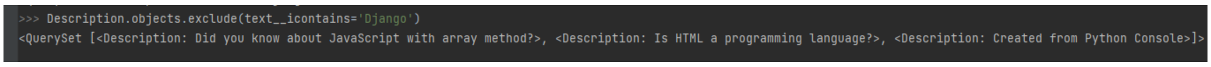
6. Filter the description that contains any word (ex. “Django”):

icontains will not be case sensitive.



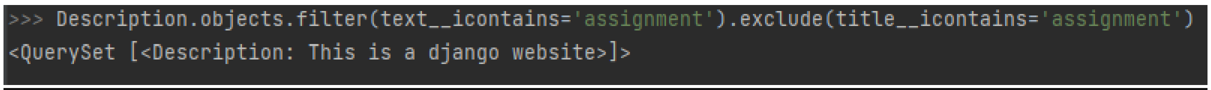
```
>>> Description.objects.filter(text__icontains='Django')
<QuerySet [<Description: This is a django website>]>
```

7. Filter the description that does not contain any word (ex. “Django”):

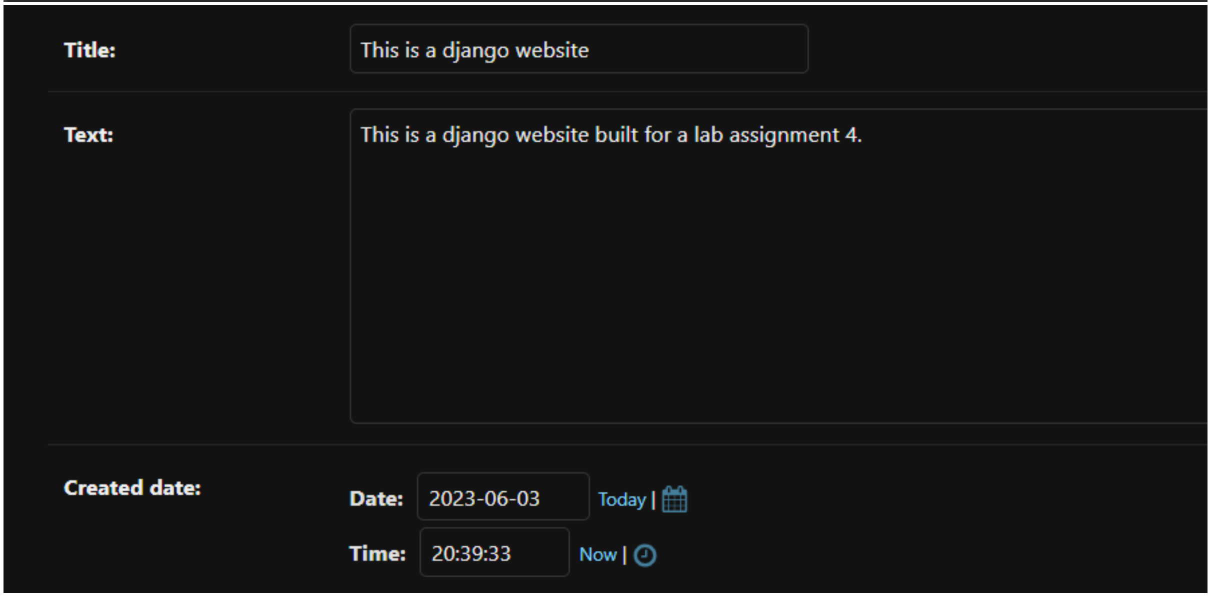


```
>>> Description.objects.exclude(text__icontains='Django')
<QuerySet [<Description: Did you know about JavaScript with array method?>, <Description: Is HTML a programming language?>, <Description: Created from Python Console>]>
```

8. Filter the description that contains any word (ex. “Django”) but the title not having the same word:



```
>>> Description.objects.filter(text__icontains='assignment').exclude(title__icontains='assignment')
<QuerySet [<Description: This is a django website>]>
```



Title:	This is a django website
Text:	This is a django website built for a lab assignment 4.
Created date:	**Date:** 2023-06-03 Today | **Time:** 20:39:33 Now |