University of Windsor

School of Computer Science
https://cs.uwindsor.ca

Master of Applied Computing
COMP-8347 – Summer 2023
Internet Applications and Distributed Systems

LAB 4 – Updating Model and Admin Files and Querying the DB

Part 1(a) – Updating Database Tables

I. Add a new model ***Order*Vehicle** in "models.py" with fields:

1. *vehicle* serving as a Foreign Key (FK) for the *Vehicle* table
   - this should indicate the ordered vehicles
2. *buyer* serving as a Foreign Key (FK) for the *Buyer* table
   - this should indicate the buyer who has ordered the vehicle
3. A field indicating number of *vehicles* being ordered
4. A field indicating the status of an order. These statuses can be with choices {0,1,2,3} such as:
   a. 0 is for a cancelled order
   b. 1 is for a placed order
   c. 2 is for a shipped order
   d. 3 is for a delivered order
5. A date field indicating the date when an order was last updated

II. Add an optional field in the **Vehicle** table which describes the product in a few words. This should be a text field

III. Add a phone number field in the **Buyer** table. The value of this field can be NULL

IV. Change the default area value in the **Buyer** table from Windsor to Chatham

V. Remove 'fullname' attribute from the the **Buyer** table

VI. Write a **__str__** method discussed in Lecture 4 for each model

VII. For the ***Order*Vehicle** model, write a method def total_price(self). This method should return the total price for all the **Vehicles** in the order

Run **makemigrations**, **sqlmigrate,** and **migrate** again until there are no errors. What is the latest file in migrations directory? Open it and check its contents.

Part 1(b) – Adding data in the Admin File

I. Add the following data in the admin.py file:

```
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. Start your server (**Run → Run 'carsite**) and navigate to admin site (127.0.0.1:8000/admin)

III. Login using *superuser* name and password (similarly as you did in Lab 3)

IV. Enter the information for each CarType, Vehicle, Buyer, and OrderVehicle as given in ***lab4data*** through the admin interface. Add additional data as you see appropriate

Part 2 – Querying the Database

In **Python console** import Django and then the models from *models.py* as follows:

```
import django

from carapp1.models import CarType, Vehicle, Buyer, OrderVehicle
```

After the above imports, write queries below to obtain the related information. Compare your query answers with ***lab4data*** file and rewrite the queries until there are no mismatches**.**

a. List the buyers having last name 'Smith'
b. List the buyers whose addresses start with '444'
c. List the buyers who live on a 'street' in Windsor area
d. List the buyers who live in Chatham and Toronto
e. List the buyers who do not live in Windsor
f. List the buyers who are interested in CarType 'Toyota'
g. List the vehicles that cost less than $30000
h. List the vehicles which are not available at the moment
i. List all the cartypes in which a buyer with username *lara* is interested in
j. List all the vehicles with a car_price > $25000 and inventory <10
k. Get the first name of the buyer of the order having primary key (or pk) equal to 2
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'
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

<u>Part 3 – Creating a model that describes our carsite website.</u>

- Create a new model called "Description" – you may also give it any name of your choice
- Create a title field for your model. This field should provide a short title for the project. Ex. A carsite website for the buyers. You can provide multiple titles to the project
- Create a text field which describes the project. We can add multiple descriptions for the project
- Create a datetime field which shows when the description was given. This field should be updated automatically whenever a new description is added for the project
- Perform migrations and register this model with the admin
- Go to Django admin page and add multiple titles and descriptions for the project
- Open the Python Console and perform the following queries to the description model:
    - Get the first description from the description model (this answer should return a query set)
    - Get the title of the first description

      Get the first description (this answer should return the text)
    - Query all the database objects
    - From the Python Console, create a new description with a title and a description
    - Filter the description title based on the starting letters (ex. "this")
    - Filter the description that contains any word (ex. "Django")
    - Filter the description that does not contain any word (ex. "Django")
    - Filter the description that contains any word (ex. "Django") but the title not having the same word