Name: Athanasios Memtsas

Student ID: f3352115

1) For ERD I didn't choose arrows to show the (one to one ,one to many) but with 0:N which shows that each value of this relation can be accomannied by N values on the other relation.

   Some assumptions I made:

   - Each customer can be assigned only at one dealer at a time and reverse.The customer isn't making the order by himself but always through a dealer.
   - Cars are assigned to one dealer at a time,or unassigned if they are available to sale.
   - A vehicle and a model can have many options,but an option is assigned to one vehicle and one model,the option must change (We consider that normally the number of options would be highly increased).
   - Each model has one brand ,but a brand can have many models as in reality
   - A vehicle may have no options and an option may not be included by any vehicle

2) Next i just implemented everything and made the (one to many etc) with foreign key and one to one with specifign the fk as unique. I included every thing for the DDL on createsql.txt
   Filled the values in the fillsql.txt

3) The queries were typical questions that would by asked by the boss or the dealers of the Automobile Company.I implemented an index about the names of the customers that would be asked a lot and it dropped the search time of the 4th query from 60 msec to 48msec.

4) The code is in Jupiter notebook form:

The examples below are not trolls they worked so i didn't want to change them (excuse me for referring to a team ☺)

Firstly we choose a selection for management search etc.
Then if selection =1 or 2 I make a dictionary and save the chosen entity on a dictionary to make my life easier.Next to find the fields I executed a select * from the entity chosen .I didn't know the use of sys in python so I tried to make insert an query by placeholders %s and it was extremely difficult.I made a string %s,%s… as many times as the fields.
Then the user only has to specify the value of the field (If integer press exactly Yes or the query will be made with a string in that place.
Commit to save it permanently!

If we specify it is not an integer then we don't need to type the: ' France ' for the non integers just France

Example for insert:
We see that the row has been added to my database

```
 Press 1 to  insert entity
 Press 2  to delete entity
 Press 3  to update entity  : 1
Entity fields are:  brand_id , b_name , fame , country
Is input int?  Type Yes else insert any key: a
type input parameter
Type the parameter: 342124
Is input int?  Type Yes else insert any key: a
type input parameter
Type the parameter: Paok
Is input int?  Type Yes else insert any key: e
type input parameter
Type the parameter: little
Is input int?  Type Yes else insert any key: e
type input parameter
Type the parameter: Pakistan
```

```
In [101]: cu=con.cursor()
          cu.execute("Select * from brand")
          cu.fetchall()

Out[101]: [(12345, 'Audi', 'great', 'Germany'),
           (12346, 'Volvo', 'small', 'Sweden'),
           (12347, 'Ferrari', 'word class', 'Italia'),
           (100000001, 'Lamborghini', 'word class', 'Italia'),
           (12312883, 'volgswagen', 'good', 'Spain'),
           (111111223, 'Rover', 'average', 'France'),
           (127755347, 'Jeep', 'good', 'Zambia'),
           (1213120045, 'Noes', 'NONE', 'USA'),
           (342124, 'Paok', 'little', 'Pakistan')]
```

Next for the delete: I used the sys and my life became a lot easier.

We observe that it has been removed.

```
 Press 1 to  insert entity
 Press 2  to delete entity
 Press 3  to update entity  : 2
Entity fields are:  brand_id , b_name , fame , country
Type something like Country='France'
 b_name='Paok'
```

```
In [104]: cu=con.cursor()
          cu.execute("select * from brand")
          cu.fetchall()
```

```
Out[104]: [(12345, 'Audi', 'great', 'Germany'),
           (12346, 'Volvo', 'small', 'Sweden'),
           (12347, 'Ferrari', 'word class', 'Italia'),
           (100000001, 'Lamborghini', 'word class', 'Italia'),
           (12312883, 'volgswagen', 'good', 'Spain'),
           (111111223, 'Rover', 'average', 'France'),
           (127755347, 'Jeep', 'good', 'Zambia'),
           (1213120045, 'Noes', 'NONE', 'USA')]
```

And for update:

(Keep in mind do not type SET I already have included it)

```
 Press 1 to  insert entity
 Press 2  to delete entity
 Press 3  to update entity  : 3
Entity fields are:  brand_id , b_name , fame , country
Type something like SET column1 = value1, column2 = value2, ... WHERE condition
 b_name='Opel' where country='Spain'
```

```
In [113]: cu=con.cursor()
          cu.execute("select * from brand")
          cu.fetchall()
```

```
Out[113]: [(12345, 'Audi', 'great', 'Germany'),
           (12346, 'Volvo', 'small', 'Sweden'),
           (12347, 'Ferrari', 'word class', 'Italia'),
           (100000001, 'Lamborghini', 'word class', 'Italia'),
           (111111223, 'Rover', 'average', 'France'),
           (127755347, 'Jeep', 'good', 'Zambia'),
           (1213120045, 'Noes', 'NONE', 'USA'),
           (12312883, 'Opel', 'good', 'Spain')]
```

For the search query :

It was easy to understand and implement the logic.I understood that the search would be select * from (entity chosen) where (field) = (something) .I didn't know if u wanted '=' but for the other operators it's the same and we have the sys library too.

For specified query :

Again I created a dictionary to save the 4 queries I made in Question 3 and im asking the user for input , then I make it an integer and insert it to the dictionary to specify which query is going to be executed. Fetchall() again to show results.

Thank you for you time!