The University of Texas at Arlington

Project 2: Part 3

Car Rental Database

Mohammed Ahmed (1001655176)

Hoang Ho (1001654608)

Shubhayu Shrestha (1001724804)

CSE 3330-004 Database Systems & File Structures

Nadra Guizani

November 29, 2021

HONOR CODE

I pledge, on my honor, to uphold UT Arlington's tradition of academic integrity, a tradition that values hard work and honest effort in the pursuit of academic excellence.

I promise that I will submit only work that I personally create or that I contribute to group collaborations, and I will appropriately reference any work from other sources. I will follow the highest standards of integrity and uphold the spirit of the Honor Code.

Introduction

This document holds the the execution of different queries including within a graphical user interface corresponding to the car rental database using SQLite3.

Task 1: Execute Queries on the CarRental2019 Database Tables

Query 1:

```
--Add an extra column ‘Returned’ to the RENTAL table. Values will be 0-for non-returned cars, and 1-for returned. Then update the ‘Returned’ column with '1' for all records that they have a payment date and with '0' for those that they do not have a payment date.
ALTER TABLE RENTAL ADD COLUMN Returned INTEGER DEFAULT 0;
UPDATE RENTAL
SET Returned = 1
WHERE PaymentDate <> 'NULL';
```

Output:

CustID	VehicleID	StartDate	OrderDate	RentalType	Qty	ReturnDate	TotalAmount	PaymentDate	Returned
203	JM3KE4DY4F0441471	2019-09-09	2019-05-22	1	4	2019-09-13	460	2019-09-09	1
210	19VDE1F3XEE414842	2019-11-01	2019-10-28	7	2	2019-11-15	1200	NULL	0
210	JTHFF2C26F135BX45	2019-05-01	2019-04-15	7	1	2019-05-08	600	2019-05-08	1
210	JTHFF2C26F135BX45	2019-11-01	2019-10-28	7	2	2019-11-15	1200	NULL	0
210	WAUTFAFH0E0010613	2019-11-01	2019-10-28	7	2	2019-11-15	1200	NULL	0
210	WBA3A9G51ENN73366	2019-11-01	2019-10-28	7	2	2019-11-15	1200	NULL	0
210	WBA3B9C59EP458859	2019-11-01	2019-10-28	7	2	2019-11-15	1200	NULL	0
210	WDCGG0EB0EG188709	2019-11-01	2019-10-28	7	2	2019-11-15	1200	NULL	0
212	19VDE1F3XEE414842	2019-06-10	2019-04-15	7	3	2019-07-01	1800	2019-06-10	1
216	1N6BF0KM0EN101134	2019-08-02	2019-03-15	7	4	2019-08-30	2740	2019-08-02	1
216	1N6BF0KM0EN101134	2019-08-30	2019-03-15	1	2	2019-09-01	230	2019-08-02	1
221	19VDE1F3XEE414842	2019-07-01	2019-06-12	7	1	2019-07-08	600	2019-07-01	1
221	19VDE1F3XEE414842	2019-07-09	2019-06-12	1	2	2019-07-11	200	2019-07-01	1
221	19VDE1F3XEE414842	2020-01-01	2019-12-15	7	4	2020-01-29	2400	NULL	0
221	JTHFF2C26F135BX45	2020-01-01	2019-12-15	7	4	2020-01-29	2400	NULL	0
221	WAUTFAFH0E0010613	2019-07-01	2019-06-12	7	1	2019-07-08	600	2019-07-01	1
221	WAUTFAFH0E0010613	2019-07-09	2019-06-12	1	2	2019-07-11	200	2019-07-01	1
221	WAUTFAFH0E0010613	2020-01-01	2019-12-15	7	4	2020-01-29	2400	NULL	0
221	WBA3A9G51ENN73366	2020-01-01	2019-12-15	7	4	2020-01-29	2400	NULL	0
221	WBA3B9C59EP458859	2020-01-01	2019-12-15	7	4	2020-01-29	2400	NULL	0
221	WDCGG0EB0EG188709	2020-01-01	2019-12-15	7	4	2020-01-29	2400	NULL	0
229	19VDE1F3XEE414842	2019-05-06	2019-04-12	1	4	2019-05-10	400	2019-05-06	1
229	WAUTFAFH0E0010613	2019-05-06	2019-04-12	1	4	2019-05-10	400	2019-05-06	1

Query 2:

```
--Create a view vRentalInfo that retrieves all information per rental.
CREATE VIEW vRentalInfo
AS SELECT R.OrderDate, R.StartDate, R.ReturnDate, CAST(JULIANDAY(R.ReturnDate)-
JULIANDAY(R.StartDate) AS INTEGER) AS TotalDays, R.VehicleID AS VIN, V.CarDescription AS Vehicle,
CASE
  WHEN V.CarType = 1 THEN 'Compact'
  WHEN V.CarType = 2 THEN 'Medium'
  WHEN V.CarType = 3 THEN 'Large'
  WHEN V.CarType = 4 THEN 'SUV'
  WHEN V.CarType = 5 THEN 'Truck'
  WHEN V.CarType = 6 THEN 'Van'
END AS Type,
CASE
  WHEN V.CarCategory = 0 THEN 'Basic'
  WHEN V.CarCategory = 1 THEN 'Luxury'
END AS Category,
R.CustID AS CustomerID, C.CustName AS CustomerName, R.TotalAmount AS OrderAmount,
```

```
CASE
  WHEN R.PaymentDate = 'NULL' THEN R.TotalAmount
  WHEN R.PaymentDate <> 'NULL' THEN 0
END AS RentalBalance
   FROM RENTAL AS R JOIN VEHICLE AS V ON R.VehicleID = V.VehicleID
        JOIN CUSTOMER AS C ON R.CustID = C.CustID
   ORDER BY StartDate ASC;

--Result
SELECT *
FROM vRentalInfo;

--How many rows returned
SELECT COUNT(OrderDate) AS Number_of_Rows
FROM vRentalInfo;
```

Output of vRentalInfo and Number of Rows Returned:

```
OrderDate   StartDate   ReturnDate  TotalDays  VIN                Vehicle      Type      Category  CustomerID  CustomerName  OrderAmount  RentalBalance
----------  ----------  ----------  ---------  -----------------  -----------  --------  --------  ----------  ------------  -----------  -------------
2019-04-15  2019-05-01  2019-05-08  7          JTHFF2C26F135BX45  Lexus IS 250C  Compact   Luxury    210         G. Clarkson   600          600
2019-04-12  2019-05-06  2019-05-10  4          19VDE1F3XEE414842  Acura ILX    Compact   Luxury    229         D. Kirkpatri  400          400
2019-04-12  2019-05-06  2019-05-10  4          WAUTFAFH0E0010613  Audi A5      Compact   Luxury    229         D. Kirkpatri  400          400
2019-04-15  2019-06-10  2019-07-01  21         19VDE1F3XEE414842  Acura ILX    Compact   Luxury    212         H. Gallegos   1800         1800
2019-06-12  2019-07-01  2019-07-08  7          19VDE1F3XEE414842  Acura ILX    Compact   Luxury    221         J. Brown      600          600
2019-06-12  2019-07-01  2019-07-08  7          WAUTFAFH0E0010613  Audi A5      Compact   Luxury    221         J. Brown      600          600
2019-06-12  2019-07-09  2019-07-11  2          19VDE1F3XEE414842  Acura ILX    Compact   Luxury    221         J. Brown      200          200
2019-06-12  2019-07-09  2019-07-11  2          WAUTFAFH0E0010613  Audi A5      Compact   Luxury    221         J. Brown      200          200
2019-03-15  2019-08-02  2019-08-30  28         1N6BF0KM0EN101134  Nissan NV    Van       Basic     216         A. Hess       2740         2740
2019-03-15  2019-08-30  2019-09-01  2          1N6BF0KM0EN101134  Nissan NV    Van       Basic     216         A. Hess       230          230
2019-05-22  2019-09-09  2019-09-13  4          JM3KE4DY4F0441471  Mazda CX5    SUV       Basic     203         A. Hernandez  460          460
2019-10-28  2019-11-01  2019-11-15  14         19VDE1F3XEE414842  Acura ILX    Compact   Luxury    210         G. Clarkson   1200         0
2019-10-28  2019-11-01  2019-11-15  14         JTHFF2C26F135BX45  Lexus IS 250C  Compact   Luxury    210         G. Clarkson   1200         0
2019-10-28  2019-11-01  2019-11-15  14         WAUTFAFH0E0010613  Audi A5      Compact   Luxury    210         G. Clarkson   1200         0
2019-10-28  2019-11-01  2019-11-15  14         WBA3A9G51ENN73366  BMW 3 Series  Compact   Luxury    210         G. Clarkson   1200         0
2019-10-28  2019-11-01  2019-11-15  14         WBA3B9C59EP458859  BMW 3 Series  Compact   Luxury    210         G. Clarkson   1200         0
2019-10-28  2019-11-01  2019-11-15  14         WDCGG0EB0EG188709  Mercedes_Benz  Compact   Luxury    210         G. Clarkson   1200         0
2019-12-15  2020-01-01  2020-01-29  28         19VDE1F3XEE414842  Acura ILX    Compact   Luxury    221         J. Brown      2400         0
2019-12-15  2020-01-01  2020-01-29  28         JTHFF2C26F135BX45  Lexus IS 250C  Compact   Luxury    221         J. Brown      2400         0
2019-12-15  2020-01-01  2020-01-29  28         WAUTFAFH0E0010613  Audi A5      Compact   Luxury    221         J. Brown      2400         0
2019-12-15  2020-01-01  2020-01-29  28         WBA3A9G51ENN73366  BMW 3 Series  Compact   Luxury    221         J. Brown      2400         0
2019-12-15  2020-01-01  2020-01-29  28         WBA3B9C59EP458859  BMW 3 Series  Compact   Luxury    221         J. Brown      2400         0
2019-12-15  2020-01-01  2020-01-29  28         WDCGG0EB0EG188709  Mercedes_Benz  Compact   Luxury    221         J. Brown      2400         0
> SELECT COUNT(OrderDate) AS Number_of_Rows
... FROM vRentalInfo;
Number_of_Rows
--------------
23
```

Task 2: Graphical User Interface for CarRental2019 Database

The graphical user interface, GUI was created using Python and its Tkinter and SQLite3 libraries. The following includes the execute queries Python for each requirement in Task 2 using the GUI. See the Readme.txt for installation details.

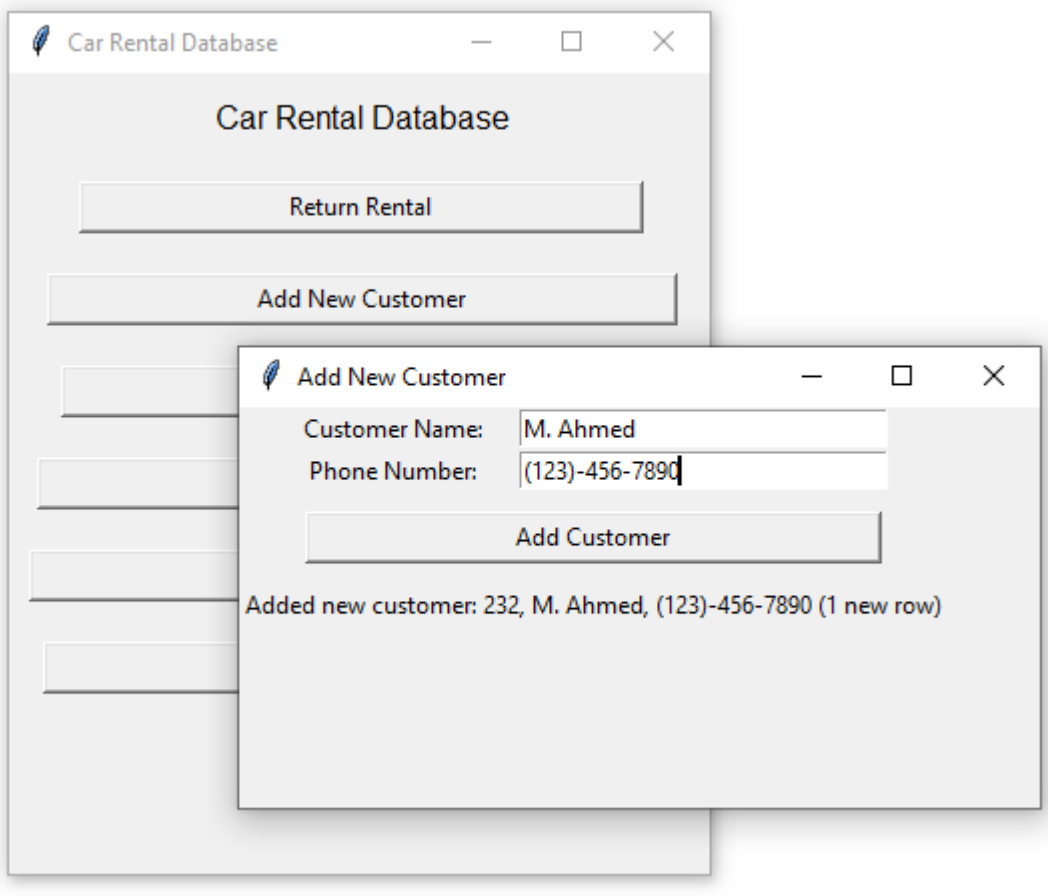
Requirement 1: Add information about a new customer. Customer ID is not provided in the query.

```
new_cust_cur.execute("INSERT INTO CUSTOMER VALUES(NULL, ?, ?)", (cust_name, phone))
```

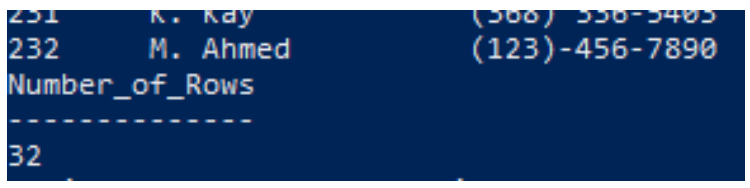
To display feedback to the user the customer information added:

```
new_cust_cur.execute("SELECT CustID FROM CUSTOMER WHERE CustName = ? AND Phone = ?",
(cust_name, phone))
```

Output:



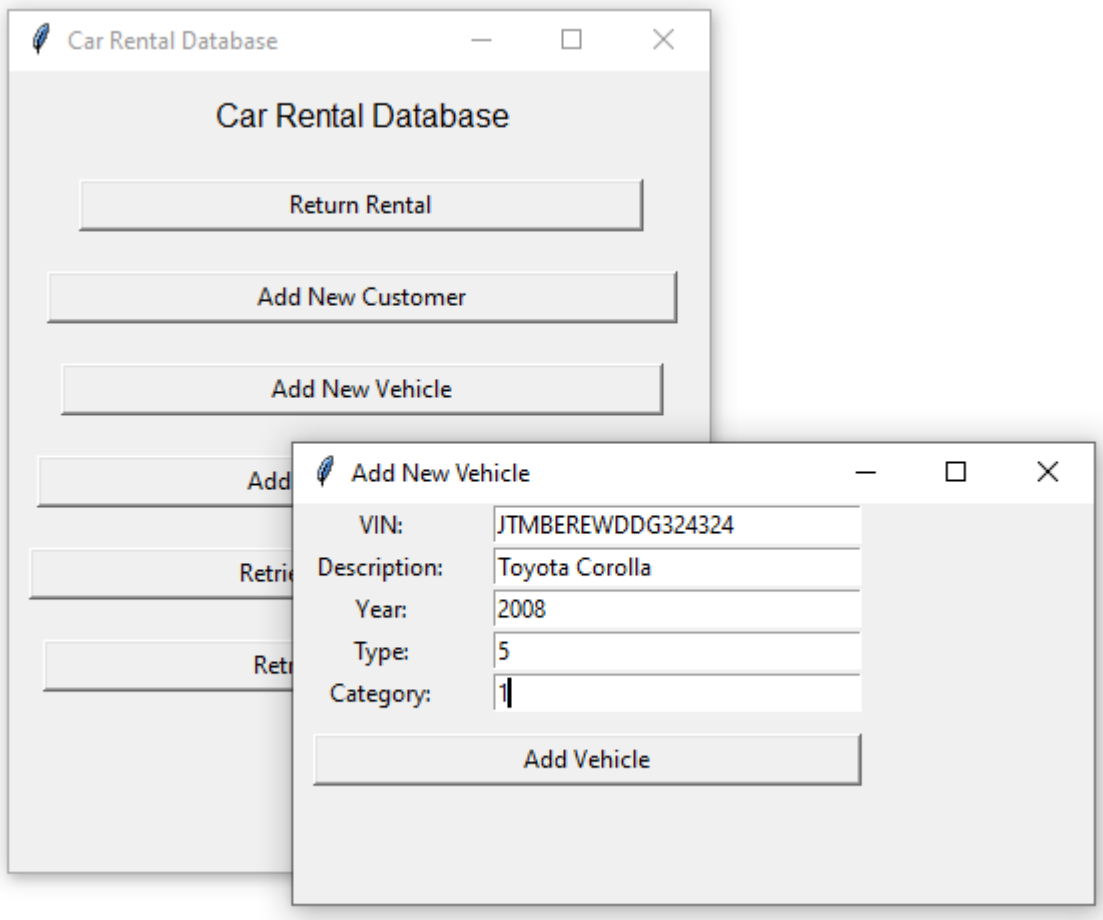
To add information about a new customer, the user clicks the “Add New Customer” button on the main “Car Rental Database” window and types the query’s input parameters and confirms by clicking the “Add Customer” button. Feedback to the user about the information added is displayed below the button.



Requirement 2: Add all the information about a new vehicle.

```
new_vehicle_cursor.execute("INSERT INTO VEHICLE VALUES(?, ?, ?, ?, ?)", (VIN, vehicle_description,
vehicle_year, vehicle_type, vehicle_category))
```

Output:



To add information about a new vehicle, the user clicks the “Add New Vehicle” button on the main “Car Rental Database” window and types the query’s input parameters and confirms by clicking the “Add Vehicle” button.

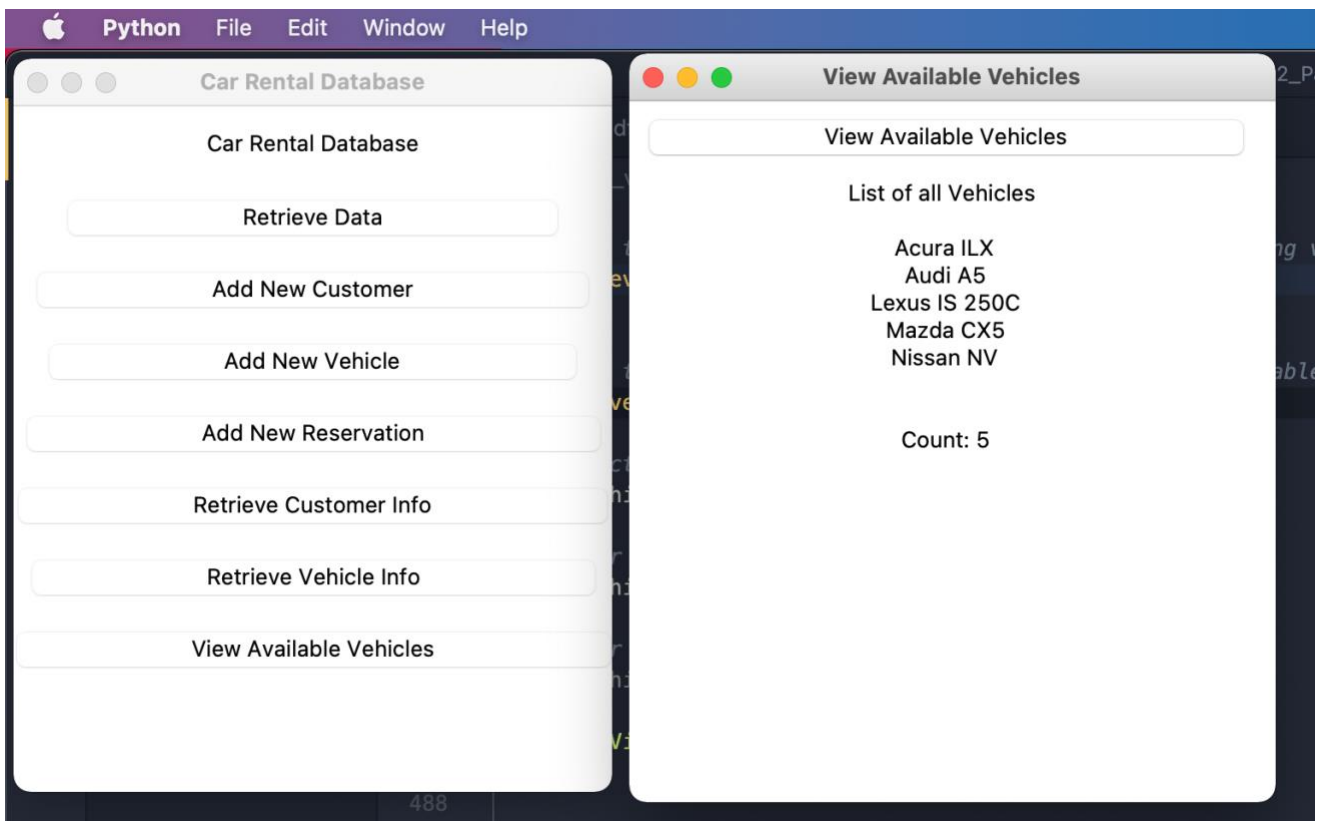
```
YV4940NB5F1191453  Volvo XC70                2015     4          1
JTMBEREWDDG324324  Toyota Corolla           2008     5          1
Number_of_Rows
--------------
61
```

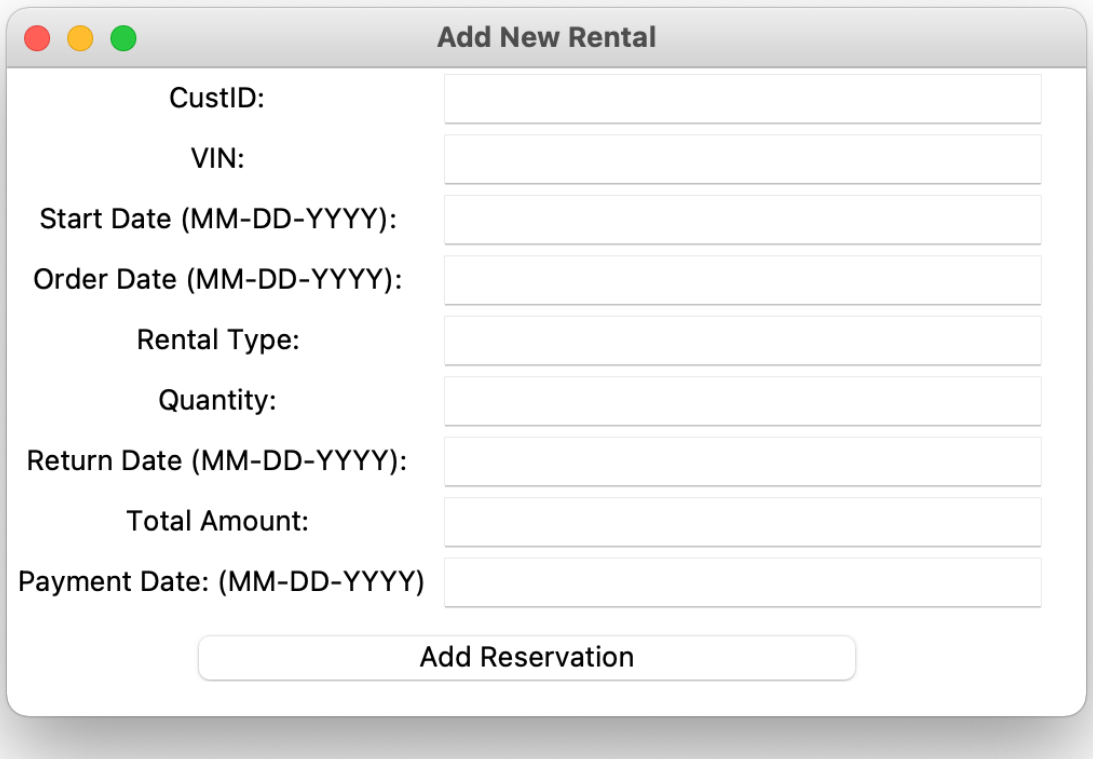
Requirement 3: Add all the information about a new rental reservation. This finds a free vehicle of the appropriate type and category for a specific rental period. Assumed that the customer has the right to either pay at the order or return date.

```
# execute
  rental_reservation_cur.execute(
      "INSERT INTO RENTAL VALUES(?,?,?,?,?,?,?,?,?,?)", (CustID, VehicleID, StartDate, OrderDate,
RentalType, Qty, ReturnDate, TotalAmount, PaymentDate, ReturnStatus))
```

```
  # List all available vehicles
  view_vehicle_cursor.execute("SELECT V.CarDescription FROM VEHICLE AS V, RENTAL AS R
WHERE R.VehicleID = V.VehicleID AND R.Returned = 1 GROUP BY V.CarDescription",)
```

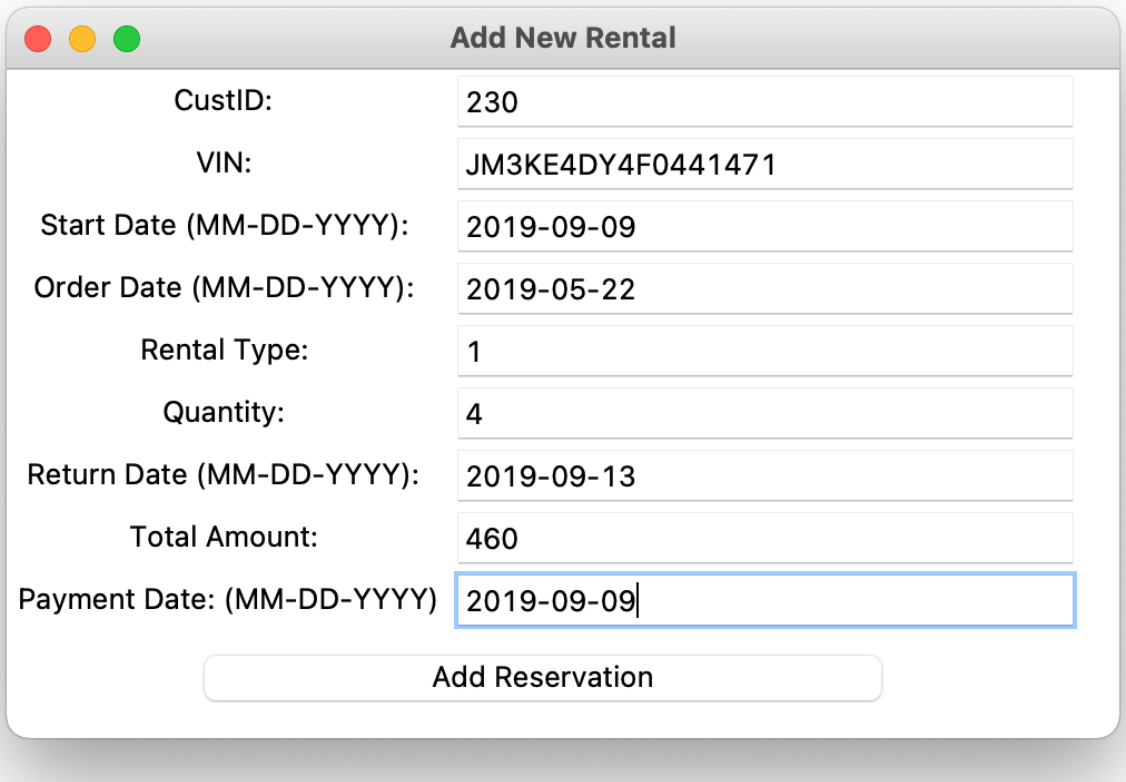
Output:





Before Adding Rental Reservation (Rental Table):

```
221|WBA3B9C59EP458859|2020-01-01|2019-12-15|7|4|2020-01-29|2400|NULL|0
221|WDCGG0EB0EG188709|2020-01-01|2019-12-15|7|4|2020-01-29|2400|NULL|0
229|19VDE1F3XEE414842|2019-05-06|2019-04-12|1|4|2019-05-10|400|2019-05-06|1
229|WAUTFAFH0E0010613|2019-05-06|2019-04-12|1|4|2019-05-10|400|2019-05-06|1
```



After Adding Rental Reservation (Rental Table):

```
221|WBA3B9C59EP458859|2020-01-01|2019-12-15|7|4|2020-01-29|2400|NULL|0
221|WDCGG0EB0EG188709|2020-01-01|2019-12-15|7|4|2020-01-29|2400|NULL|0
229|19VDE1F3XEE414842|2019-05-06|2019-04-12|1|4|2019-05-10|400|2019-05-06|1
229|WAUTFAFH0E0010613|2019-05-06|2019-04-12|1|4|2019-05-10|400|2019-05-06|1
230|JM3KE4DY4F0441471|2019-09-09|2019-05-22|1|4|2019-09-13|460|2019-09-09|0
sqlite>
```

The user must first check the vehicles currently available by clicking the ‘View Available Vehicles’ button. Afterwards, the user can click on the ‘Add New Reservation’ button to create a new rental reservation.

Requirement 4: Handle the return of a rented car. Transaction prints the total customer payment due for that rental, enters it in the database, and updates the Returned attribute accordingly. Retrieves a rental by the return date, customer name, and vehicle information.

To retrieve the CustID and TotalAmount for the Rental:

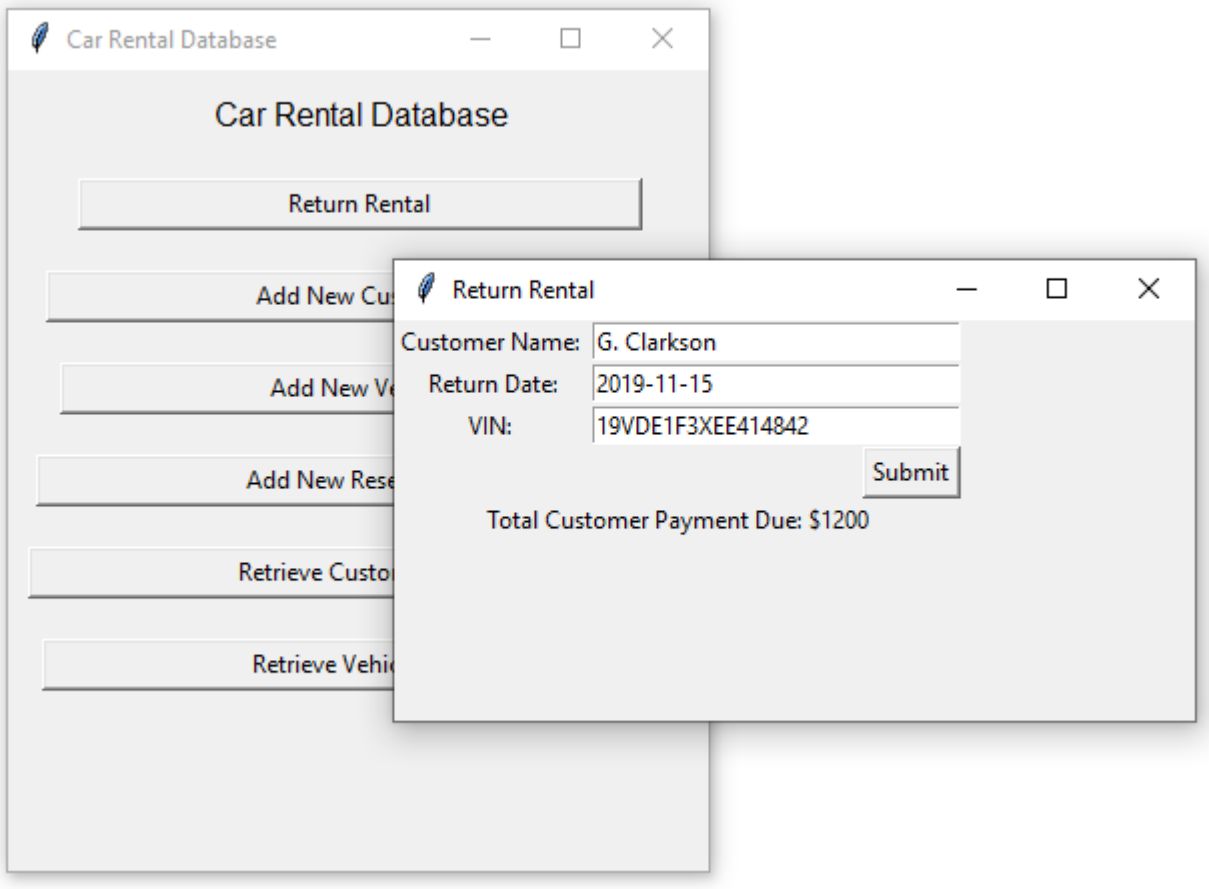
```
query = "SELECT R.CustID, R.TotalAmount FROM RENTAL AS R JOIN CUSTOMER AS C ON R.CustID =
C.CustID WHERE R.ReturnDate = '" + return_date + "' AND C.CustName = '" + cust_name + "' AND
R.VehicleID = '" + vehicle_info + """

print(query)
db_cur.execute(query)
```

To update rental Returned and PaymentDate attributes:

db_cur.execute("UPDATE RENTAL SET Returned = 1, PaymentDate = CASE WHEN PaymentDate = 'NULL' THEN '" + return_date + "' END WHERE CustID = " + str(result[0][0]) + " AND ReturnDate = '" + return_date + "' AND VehicleID = '" + vehicle_info + "'")

Output:



To return a rental, the user clicks the “Return Rental” button on the main “Car Rental Database” window and types the query’s input parameters for the rental information. The user confirms by clicking the “Submit” button. Feedback to the user about the payment made is displayed below the button.

Before rental return update:

CustID	VehicleID	StartDate	OrderDate	RentalType	Qty	ReturnDate	TotalAmount	PaymentDate	Returned
203	JM3KE4DY4F0441471	2019-09-09	2019-05-22	1	4	2019-09-13	460	2019-09-09	1
210	19VDE1F3XEE414842	2019-11-01	2019-10-28	7	2	2019-11-15	1200	NULL	0

After rental return update:

CustID	VehicleID	StartDate	OrderDate	RentalType	Qty	ReturnDate	TotalAmount	PaymentDate	Returned
203	JM3KE4DY4F0441471	2019-09-09	2019-05-22	1	4	2019-09-13	460	2019-09-09	1
210	19VDE1F3XEE414842	2019-11-01	2019-10-28	7	2	2019-11-15	1200	2019-11-15	1

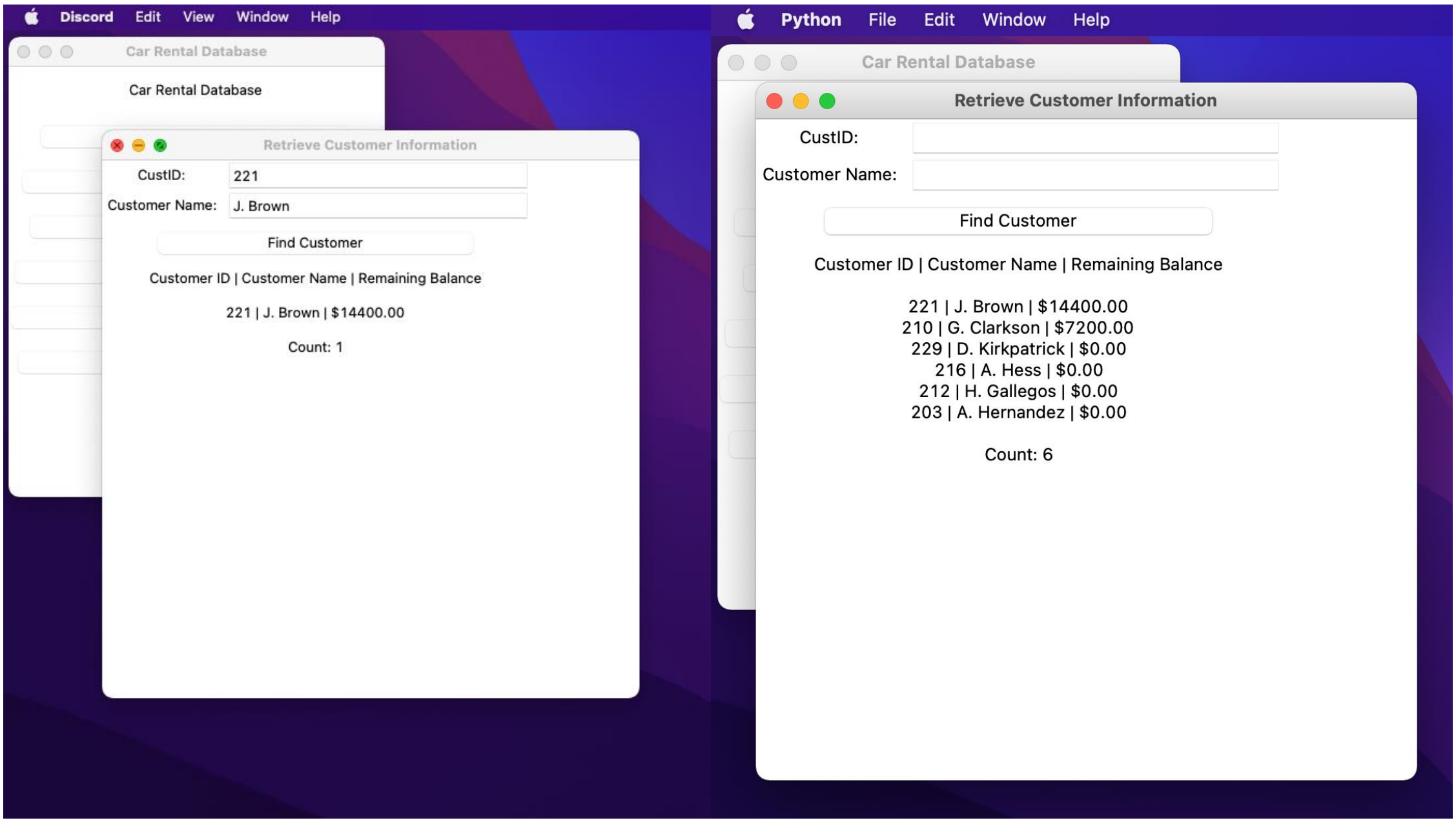
Requirement 5a: List for every customer the ID, name, and if there is any remaining balance. The user has the right to search either by a customer’s ID, name, part of the name, or to run the query with no filters/criteria. The amount is in US dollars. For customers with zero (0) or NULL balance, returns zero dollars ($0.00). In the case that the user decides not to provide any filters, order the results based on the balance amount. Returns meaningful attribute names and all records.

Query:

```
 if CustID != '':
   retrieve_cust_cursor.execute(
       "SELECT CustomerID, CustomerName, SUM(RentalBalance) FROM vRentalInfo WHERE CustomerID LIKE ? AND CustomerName LIKE ? GROUP BY CustomerID ORDER BY COUNT(RentalBalance) DESC", (('%'+CustID+'%'), ('%'+CustName+'%'),))
 elif CustName != '':
```

```
    retrieve_cust_cursor.execute("SELECT CustomerID, CustomerName, SUM(RentalBalance) FROM
vRentalInfo WHERE CustomerName LIKE ? GROUP BY CustomerName ORDER BY
COUNT(RentalBalance) DESC", ('%'+CustName+'%',))
  else:
    retrieve_cust_cursor.execute("SELECT CustomerID, CustomerName, SUM(RentalBalance) FROM
vRentalInfo GROUP BY CustomerName ORDER BY COUNT(RentalBalance) DESC")
```

Output:

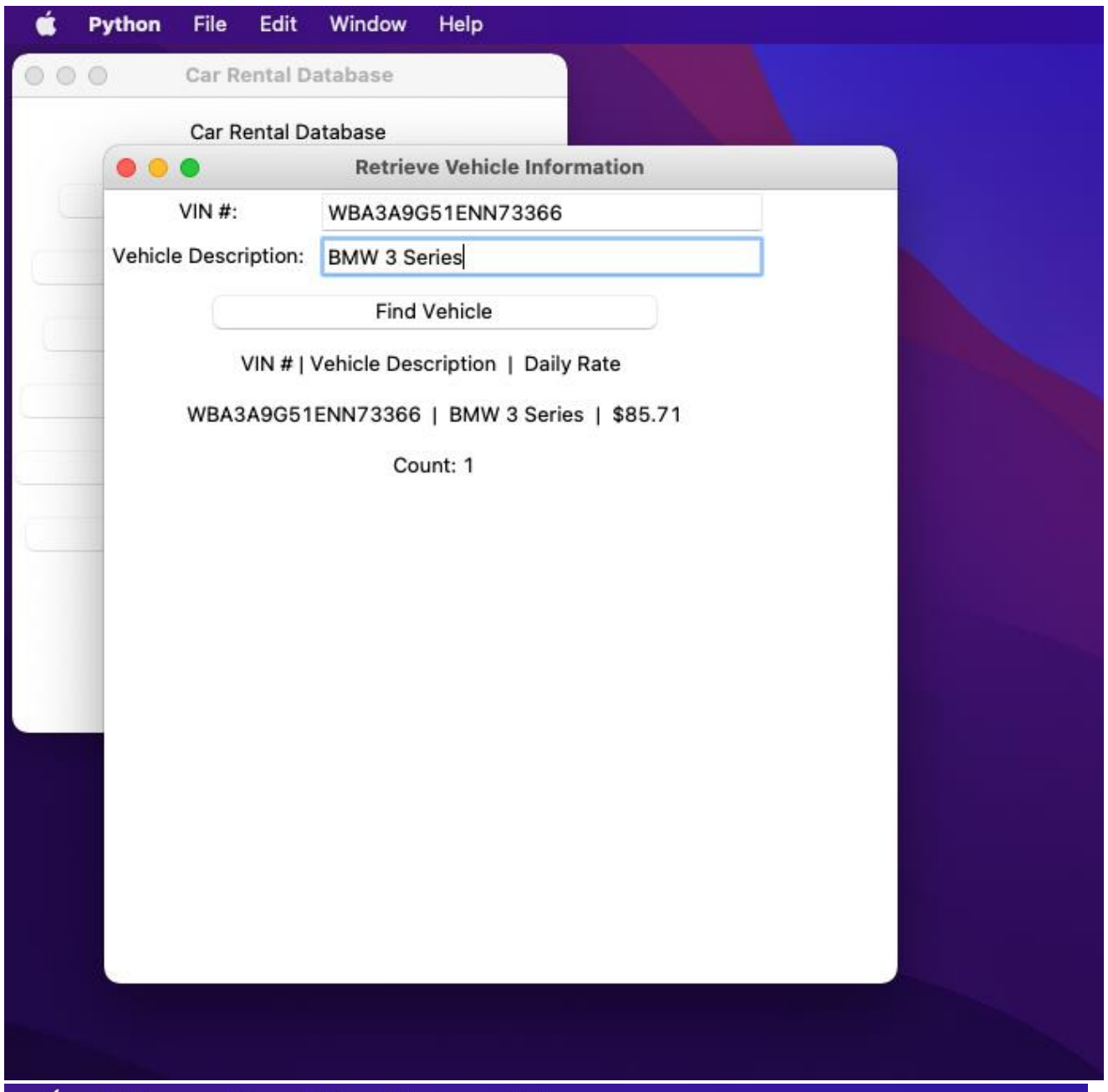

Discord Edit View Window Help
Car Rental Database
Car Rental Database
Retrieve Customer Information
CustID: 221
Customer Name: J. Brown
Find Customer
Customer ID | Customer Name | Remaining Balance
221 | J. Brown | $14400.00
Count: 1
Python File Edit Window Help
Car Rental Database
Retrieve Customer Information
CustID:
Customer Name:
Find Customer
Customer ID | Customer Name | Remaining Balance
221 | J. Brown | $14400.00
210 | G. Clarkson | $7200.00
229 | D. Kirkpatrick | $0.00
216 | A. Hess | $0.00
212 | H. Gallegos | $0.00
203 | A. Hernandez | $0.00
Count: 6

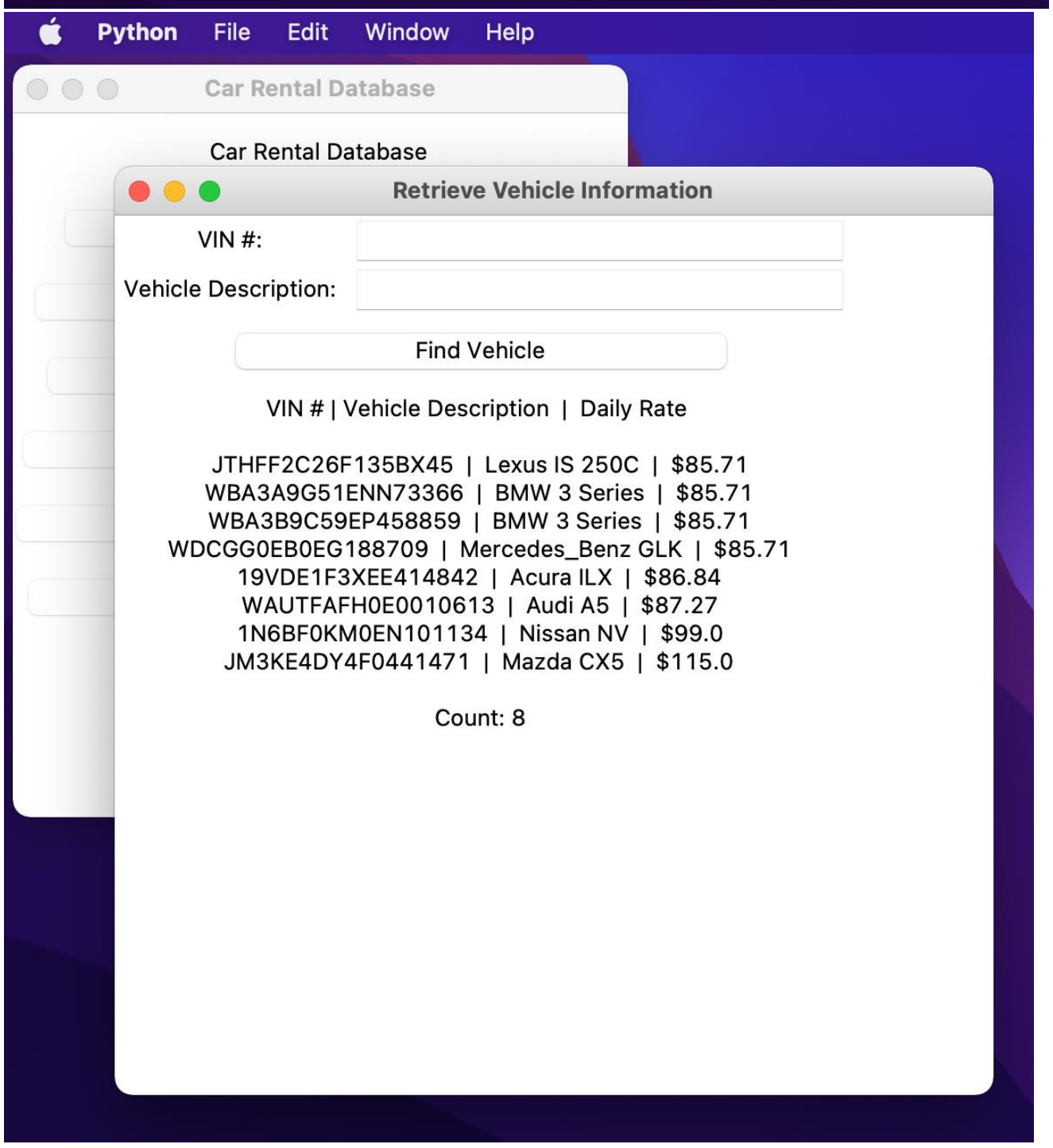
Requirement 5b: List for every vehicle the VIN, the description, and the average DAILY price. The user has the right either to search by the VIN, vehicle's description, part of the description, or to run the query with no filters/criteria. An example criterion would be all 'BMW' vehicles. The amount needs to be in US dollars. The average DAILY price derives from the rental table, and the amount needs to have two decimals as well as the dollar '$' sign. For vehicles that they do not have any rentals, substitute the NULL value with a 'Non-Applicable' text. Returns meaningful attribute names. In the case that the user decides not to provide any filters, order the results based on the average daily price.

Query:

```
 if VehicleID != '':
   retrieve_vehicle_cursor.execute(
       "SELECT VIN, Vehicle, (ROUND(CAST(SUM(OrderAmount) AS float)/SUM(TotalDays), 2)) FROM
vRentalInfo WHERE VIN LIKE ? AND Vehicle LIKE ? GROUP BY VIN ORDER BY
(SUM(OrderAmount)/SUM(TotalDays)) ASC", (('%'+VehicleID+'%'),('%'+CarDescription+'%'),))
 elif CarDescription != '':
   retrieve_vehicle_cursor.execute(
       "SELECT VIN, Vehicle, (ROUND(CAST(SUM(OrderAmount) AS float)/SUM(TotalDays), 2)) FROM
vRentalInfo WHERE Vehicle LIKE ? GROUP BY VIN ORDER BY (SUM(OrderAmount)/SUM(TotalDays))
ASC", (('%'+CarDescription+'%'),))
 else:
   retrieve_vehicle_cursor.execute(
       "SELECT VIN, Vehicle, (ROUND(CAST(SUM(OrderAmount) AS float)/SUM(TotalDays), 2)) FROM
vRentalInfo GROUP BY VIN ORDER BY (SUM(OrderAmount)/SUM(TotalDays)) ASC")
```

Output:


Python File Edit Window Help
Car Rental Database
Car Rental Database
Retrieve Vehicle Information
VIN #:
WBA3A9G51ENN73366
Vehicle Description:
BMW 3 Series
Find Vehicle
VIN # | Vehicle Description | Daily Rate
WBA3A9G51ENN73366 | BMW 3 Series | $85.71
Count: 1


Python File Edit Window Help
Car Rental Database
Car Rental Database
Retrieve Vehicle Information
VIN #:
Vehicle Description:
Find Vehicle
VIN # | Vehicle Description | Daily Rate
JTHFF2C26F135BX45 | Lexus IS 250C | $85.71
WBA3A9G51ENN73366 | BMW 3 Series | $85.71
WBA3B9C59EP458859 | BMW 3 Series | $85.71
WDCGG0EB0EG188709 | Mercedes_Benz GLK | $85.71
19VDE1F3XEE414842 | Acura ILX | $86.84
WAUTFAFH0E0010613 | Audi A5 | $87.27
1N6BF0KM0EN101134 | Nissan NV | $99.0
JM3KE4DY4F0441471 | Mazda CX5 | $115.0
Count: 8

TEAM CONTRIBUTIONS

Mohammed Ahmed, Hoang Ho, and Shubhayu Shrestha worked on Task 1 as a team. The following is the individual contributions to the GUI requirements.

Mohammed Ahmed: Requirements 3 and 5b
Hoang Ho: Requirements 1 and 4
Shubhayu Shrestha: Requirements 2 and 5a