

CSE 3330: Project 2 Phase 3

May 1, 2021

Team 15

Jorge Estrada

Josh Burkey

Bevan Philip

Honor Code

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.

Members:

Jorge Estrada

Josh Burkey

Bevan Philip

Task 1

Query 1:

```
ALTER TABLE RENTAL
ADD Returned INT(1);
SELECT *,
CASE PaymentDate
WHEN 'NULL' THEN '0'
ELSE '1'
END AS Returned
FROM RENTAL;
```

```
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
```

23 rows were returned

Query 2:

```
CREATE VIEW vRentalInfo
AS
SELECT RENTAL.OrderDate,
RENTAL.StartDate,
RENTAL.ReturnDate,
(RENTAL.RentalType*RENTAL.Qty) AS "TotalDays",
RENTAL.VehicleID AS "VIN",
VEHICLE.Description AS "Vehicle",
CASE VEHICLE.Type
WHEN '1' THEN 'Compact'
WHEN '2' THEN 'Medium'
WHEN '3' THEN 'Large'
WHEN '4' THEN 'SUV'
WHEN '5' THEN 'Truck'
ELSE 'VAN'
END AS Type,
CASE VEHICLE.Category
WHEN '0' THEN 'Basic'
ELSE 'Luxury'
END AS Category,
RENTAL.CustID AS "CustomerID",
CUSTOMER.Name AS "CustomerName",
RENTAL.TotalAmount AS "OrderAmount",
CASE PaymentDate
WHEN 'NULL' THEN '0'
ELSE RENTAL.TotalAmount
END AS RentalBalance
FROM RENTAL, CUSTOMER, VEHICLE
WHERE RENTAL.CustID = CUSTOMER.CustID
AND VEHICLE.VehicleID = RENTAL.VehicleID
ORDER BY RENTAL.StartDate ASC;
```

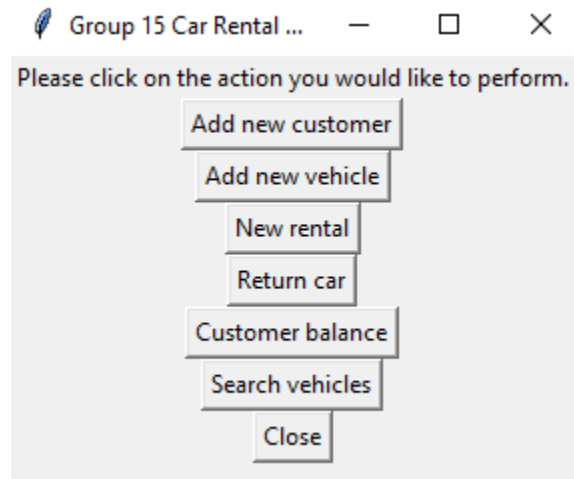
```

sqlite> select * FROM vRentalInfo;
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. Kirkpatrick|400|400
2019-04-12|2019-05-06|2019-05-10|4|WAUTFAFH0E0010613|"Audi A5"|Compact|Luxury|229|D. Kirkpatrick|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 GLK"|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 GLK"|Compact|Luxury|221|J. Brown|2400|0

```

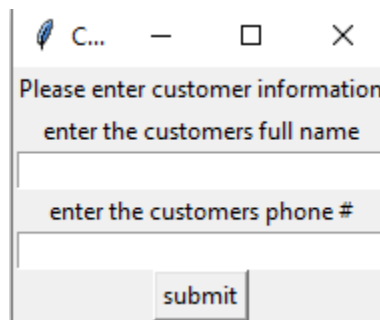
23 rows were returned

Task 2



The above screenshot is of the main page of the GUI for this project. It features the buttons which take the user to do whatever task they would like to complete. It also features a close button that will just close the gui without interfering with the database.

1)



This screenshot contains the GUI for part 1 of the GUI queries. This GUI contains two files where the user just enters their full name and their phone number and they will be inserted as a new customer into the database.

```
sqliteInsert= """"INSERT INTO CUSTOMER (CustID, Name, Phone) VALUES (?, ?, ?);"""
```

2)

Vehicle entry screen

Please enter vehicle information

enter the VIN #

enter the vehicles make

enter the year

enter the type 1-6

1:Compact, 2:Medium, 3:Large, 4:SUV, 5:Truck, 6:VAN

enter the category #

0 = Basic, 1 = Luxury

submit

This part of our GUI features the required fields the user must enter in order to add a vehicle into the database. The user must enter the VIN, Vehicle make and the year. Along with that information the user must also enter the type and category of the vehicle, which needs to be inputted by the given number for the certain type and category. For example to enter a compact vehicle the user must type '1' and if the car is a luxury car then the user must enter '1' for the category.

```
sqliteInsert= """"INSERT INTO VEHICLE (VehicleID, Description, Year, Type, Category)
VALUES (?, ?, ?, ?, ?);"""
```

3)

rental entry screen

Enter the type 1-6
1:Compact, 2:Medium, 3:Large, 4:SUV, 5:Truck, 6:VAN

1

Enter the category #
0 = Basic, 1 = Luxury

1

Enter the rental period start and end
yyyy-mm-dd(space)yyyy-mm-dd
2019-06-01 2019-06-20

find available vehicles

enter car VIN:

enter customer id:

weekly or daily? 1 or 7:

how many weeks or days?:

Pay now? y/n:

rent vehicle

available vehicles:

This screenshot shows the rental entry screen. First the user searches for an available car to find the type and category of car they are looking for, then they enter a pair of dates for the rental period they are interested in hitting “find available vehicles” will show available vehicles for the period between those dates and display them under “available vehicles”. The user then enters the VIN number of the vehicle they want, their customer ID, the rate they want either weekly or daily and how many days they want and choose whether to pay now or at the return day. Rent vehicle then updates the database with the new rental entry.

This query searches for available vehicles in the rental period

```
SELECT v.VehicleID AS VIN, v.Description, v.year
FROM VEHICLE v, RENTAL r
WHERE v.VehicleID = r.VehicleID AND (v.Type = ? AND v.Category = ?) AND ((r.ReturnDate
>= ?) OR (r.StartDate <= ?))
GROUP BY r.VehicleID;
```

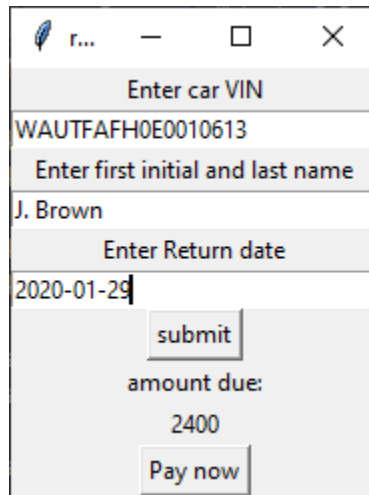

This query gets the rental rate

```
SELECT *  
FROM RATE  
WHERE Type = ? AND Category = ?;
```

Which is then used by this query to update the database with the new rental

```
INSERT INTO RENTAL  
    (CustID, VehicleID, StartDate, OrderDate, RentalType, Qty, ReturnDate,  
    TotalAmount, PaymentDate)  
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);
```

4)



Enter car VIN
WAUTFAFH0E0010613
Enter first initial and last name
J. Brown
Enter Return date
2020-01-29
submit
amount due:
2400
Pay now

This screenshot shows the rental return screen. When a user wants to return a vehicle they put in the VIN number of the car, their first initial and last name and the return date for the car. Hitting submit returns the amount due for that specific rental. Pay now updates the payment date in the rental database.

This query gets the amount the user owes for that specific rental:

```
UPDATE RENTAL
```

```
SET PaymentDate = ?
```

```
WHERE VehicleID = ? AND ReturnDate = ?;
```

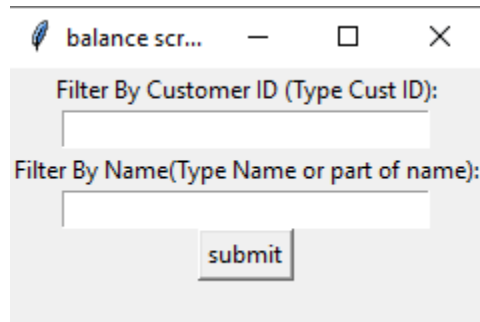
This query updates the database payment date

```
SELECT TotalAmount
```

```
FROM RENTAL r, CUSTOMER c
```

```
WHERE r.CustID = c.CustID AND r.VehicleID = ? AND c.Name = ? AND ReturnDate = ?;
```

5a)

A screenshot of a Windows-style application window titled "balance scr...". The window has a standard title bar with minimize, maximize, and close buttons. Inside the window, there are two text input fields. The first field is labeled "Filter By Customer ID (Type Cust ID):" and the second is labeled "Filter By Name (Type Name or part of name):". Below these fields is a button labeled "submit".

This screenshot handles the query that shows customers and their respective balances that they have paid or still owe to the rental company. The way this part of the GUI works is that the user can type in either a customer ID or name of the user and retrieve the records from the view that was created with the filter provided by the user. The user can also leave the fields empty and it will return all records without the filter. Another part of the GUI is that the user can just input part of a name in the filter by name part and it will return any information on a user with that part of the name.

If the user provides an empty field for CustomerID and Name:

```
cursor.execute("SELECT CustomerID, CustomerName, '$'||RentalBalance||'.00' RentalBalance  
FROM vRentalInfo ORDER BY RentalBalance ASC;")
```

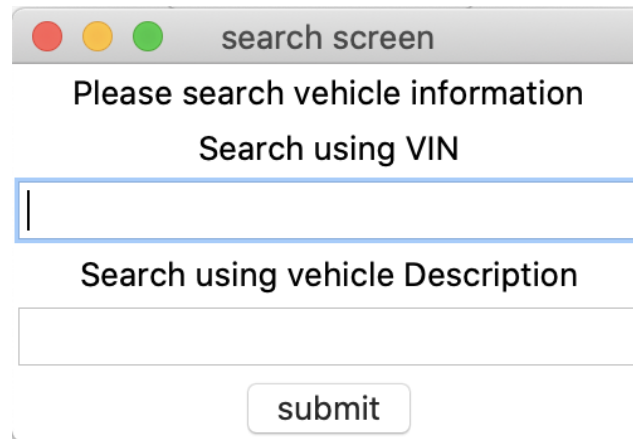
If the user only fills in Name:

```
cursor.execute("SELECT CustomerID, CustomerName, '$'||RentalBalance||'.00' RentalBalance  
FROM vRentalInfo WHERE CustomerName LIKE ? ORDER BY RentalBalance  
ASC;",(Name,))
```

If the user only fills in CustomerID:

```
cursor.execute("SELECT CustomerID, CustomerName, '$'||RentalBalance||'.00' RentalBalance  
FROM vRentalInfo WHERE CustomerID= ? ORDER BY RentalBalance ASC;",(CustID,))
```

5b)



search screen

Please search vehicle information

Search using VIN

Search using vehicle Description

submit

This screenshot shows the search screen which allows you to search information about a particular car using its VIN number, description or both included in the vRentalInfo view. The description may be provided as a whole or partially. Searching with no input lists information about every car contained in the view. The search provides the VIN number, description and daily rate in dollars of each rental based on input.

If user inputs just the description or partial description:

```
cursor.execute("SELECT vRentalInfo.VIN, vRentalInfo.Vehicle, '$'||RATE.Daily||'.00' FROM  
RATE, vRentalInfo, VEHICLE WHERE vRentalInfo.Vehicle LIKE ? AND RATE.Type =  
VEHICLE.Type AND RATE.Category = VEHICLE.Category AND VEHICLE.VehicleID =  
vRentalInfo.VIN;",(Description,))
```

If user inputs just the VIN number:

```
cursor.execute("SELECT vRentalInfo.VIN, vRentalInfo.Vehicle, '$'||RATE.Daily||'.00' FROM  
RATE, vRentalInfo, VEHICLE WHERE vRentalInfo.VIN LIKE ? AND RATE.Type =  
VEHICLE.Type AND RATE.Category = VEHICLE.Category AND VEHICLE.VehicleID =  
vRentalInfo.VIN;",(Vin,))
```

If user inputs both:

```
cursor.execute("SELECT vRentalInfo.VIN, vRentalInfo.Vehicle, '$'||RATE.Daily||'.00' FROM  
RATE, vRentalInfo, VEHICLE WHERE (vRentalInfo.VIN = ? AND vRentalInfo.Vehicle = ?)  
AND RATE.Type = VEHICLE.Type AND RATE.Category = VEHICLE.Category AND  
VEHICLE.VehicleID = vRentalInfo.VIN;",(Vin, Description,))
```

If there is no user input:

```
cursor.execute("SELECT vRentalInfo.VIN, vRentalInfo.Vehicle, '$'||RATE.Daily||'.00' FROM  
RATE, vRentalInfo, VEHICLE WHERE RATE.Type = VEHICLE.Type AND RATE.Category =  
VEHICLE.Category AND VEHICLE.VehicleID = vRentalInfo.VIN ORDER BY RATE.Daily;")
```

If user inputs an incorrect value:

else:

print("Not applicable")