CSE 3330: Project 2 Phase 2

April 15, 2021

Team 15

Task 1

```
def createTables():
    try:
        sqliteConnection = sqlite3.connect('Project2.db')
        cursor = sqliteConnection.cursor()
        cursor.execute('''PRAGMA foreign_keys = ON;''')
        sqlite_create_customer = '''CREATE TABLE CUSTOMER
                                 (CustID INT(3) NOT NULL PRIMARY KEY,
                                   Name TEXT NOT NULL,
                                   Phone TEXT NOT NULL);'''
        cursor.execute(sqlite_create_customer)

        sqlite_create_rental = '''CREATE TABLE RENTAL
                                   (CustID INT(3) NOT NULL,
                                    VehicleID TEXT NOT NULL,
                                    StartDate TEXT NOT NULL,
                                    OrderDate TEXT NOT NULL,
                                    RentalType INT(1) NOT NULL,
                                    Qty INT(1) NOT NULL,
                                    ReturnDate TEXT NOT NULL,
                                    TotalAmount INT NOT NULL,
                                    PaymentDate TEXT,
                                    FOREIGN KEY(CustID) REFERENCES CUSTOMER(CustID),
                                    FOREIGN KEY(VehicleID) REFERENCES VEHICLE(VehicleID));'''
        cursor.execute(sqlite_create_rental)

        sqlite_create_vehicle = '''CREATE TABLE VEHICLE
                                 (VehicleID TEXT NOT NULL PRIMARY KEY,
                                   Description TEXT NOT NULL,
                                   Year INT(4) NOT NULL,
                                   Type INT(1) NOT NULL,
                                   Category INT(1) NOT NULL);'''
        cursor.execute(sqlite_create_vehicle)

        sqlite_create_rate = '''CREATE TABLE RATE
                                   (Type INT(1) NOT NULL,
                                    Category INT(1) NOT NULL,
                                    Weekly INT NOT NULL,
                                    Daily INT NOT NULL,
                                    FOREIGN KEY(Type) REFERENCES VEHICLE(Type),
                                    FOREIGN KEY(Category) REFERENCES VEHICLE(Category));'''
        cursor.execute(sqlite_create_rate)

        cursor.close()
    except sqlite3.Error as error:
        print("Failed to insert variable into sqlite table", error)
    finally:
        if(sqliteConnection):
            sqliteConnection.close()
            print("Sqlite Connection is closed")

createTables()
```

Customer Table Explanation:
For the customer table we decided that CustID was going to be the primary key, because CustID was going to define every customer since it is a unique value for different customers. We also decided to make CustID as NOT NULL, because since it is the primary key we needed to have a value there to go back to. For Name and Phone we decided to make both of the attributes Text types since they have a mixture of characters for the fields, and also making the attributes NOT NULL, because we also needed to have values there since it is the information for the customer.

Rental Table Explanation:
For the rental table we decided to make CustID and VehicleID as foreign keys, because it would help the rental table connect to both the customer and vehicle table. We also made the foreign keys as NOT NULL just so we have the value to go back on whenever we need information from the customer or vehicle table. For RentalType, Qty, and TotalAmount we decided to go with INT types since they were numerical values that we would eventually have to do arithmetic on in our queries. Aside from the foreign keys all other attributes, except PaymentDate had to be NOT NULL since the information is needed for the table. On the other hand PaymentDate could be NULL because there can be customers that have not paid their rentals, so we did not add the NOT NULL constraint to that attribute.

Vehicle Table Explanation:
For the vehicle table we decided to have VehicleID as the primary key for the same reason we had CustID as the primary key for customer. We also made VehicleID as NOT NULL for the fact that we need to have a value there to reference in our queries and for the foreign keys to reference. For all of the other attributes we also decided to have them as NOT NULL, because that was also information that needed to be there to properly identify and describe a car in our queries. Description needed to be a text type since we use characters in the description of cars, and for the rest of the attributes we made them as int types just for the fact that they were numerical values.

Rate Table Explanation:
For the rate table we decided not to go with a primary key, because we didn't have any attributes that were unique or that would define the rate table, so we decided to go with two foreign keys, which are Type and Category and they reference the vehicle table. We also decided to make those foreign keys as NOT NULL since we need values there to help us reference the vehicle table. For the rest of the attributes in rate we had them as int types since they were all numerical values and they had the payment rates for certain vehicles. Not only that, we also decided to make them NOT NULL, since we need every field to have a value to help us define the rate for a given type and category.

Task 2

File loading:

For file loading we used a modified version of the system we used for project 1 (import, snip by comma, load in). The main difference aside from shortening some functions to account for different attributes was adding a “skip” to ignore the first line of each file because that contained the header names.

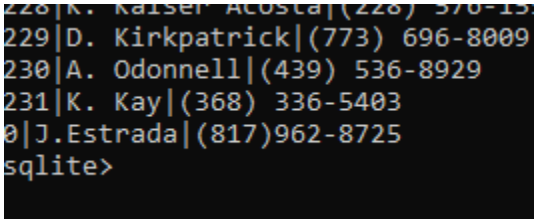
As an example this is the CUSTOMER loading loop in python:

```
146 def filereadCustomer(filename):
147     with open(filename,"r") as fileo:
148 
149         newline = fileo.readlines()
150         fline = 0
151         for line in newline:
152             count = 0
153             'for each line strip the new line characters for each os'
154             line = line.rstrip('\n')
155             line = line.rstrip('\r')
156 
157             print("reading customer file to database")
158 
159             'split the line into segments for processing'
160             segment = line.split(",")
161             'for each split part of the line strip off the comma and feed them in'
162             if fline == 0:
163                 'skip first line for read in to not copy header names'
164                 fline = 1
165             elif fline == 1:
166                 for part in segment:
167                     if count == 0:
168                         CustID = part
169                         count = count+1
170                     elif count == 1:
171                         Name = part
172                         count = count+1
173                     elif count == 2:
174                         Phone = part
175                         count = count+1
176 
177 
178                 'call the insert function at the end of each line'
179                 customerVariableTable(CustID, Name, Phone)
180     'close the file'
181     fileo.close()
182 filename = "CUSTOMER.CSV"
183 filereadCustomer(filename)
```

Task 3

Question 1: Insert yourself as a New Customer. Do not provide the CustomerID in your query.

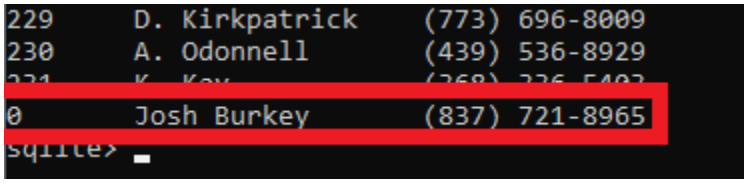
INSERT INTO CUSTOMER VALUES (0, 'J. Estrada', '(817-962-8725');



No records in this table were affected, however there were 32 records returned due to the addition of the insert. Question was a bit ambiguous so I just put in a 0 as the Customer ID for myself.

Question 2: Update your phone number to (837) 721-8965

UPDATE CUSTOMER
SET Phone = '(837) 721-8965'
WHERE Name LIKE 'Josh Burkey';



records affected: 1
Used the insert from Q1 for the test record

Question 3: Increase only daily rates for luxury vehicles by 5%

```
.header on
UPDATE RATE
SET Daily = Daily+(Daily*0.05)
WHERE Category = 1;
SELECT Daily FROM RATE;
```

Record returned: 10

There were updates to daily rates for luxury vehicles. 4 entries were increased by 5%.

Question 4-a: Insert a new luxury van with the following info: Honda Odyssey 2019, vehicle id: 5FNRL6H58KB133711

INSERT INTO VEHICLE VALUES ('5FNRL6H58KB133711', '"Honda Odyssey"', 2019, 6,1);

```
WBA3A9G51ENN73366|"BMW 3 Series"|2014|1|1
WBA3B9C59EP458859|"BMW 3 Series"|2014|1|1
WBAVL1C57EVR93286|"BMW X1"|2014|4|1
WDCGG0EB0EG188709|"Mercedes_Benz GLK"|2014|1
YV440MDD6F2617077|"Volvo XC60"|2015|4|1
YV4940NB5F1191453|"Volvo XC70"|2015|4|1
5FNRL6H58KB133711|"Honda Odyssey"|2019|6|1
sqlite>
```

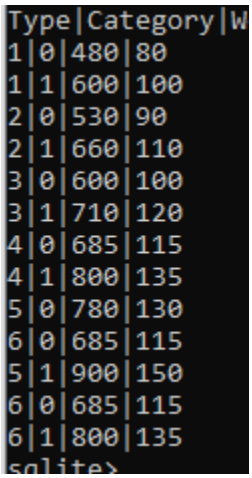
No records were affected but there were 61 records that were returned due to the addition in this insert statement

Question 4-b: You also need to insert the following rates:

5	1	900.00	150.00
6	1	800.00	135.00

INSERT INTO RATE VALUES(5,1,900,150);

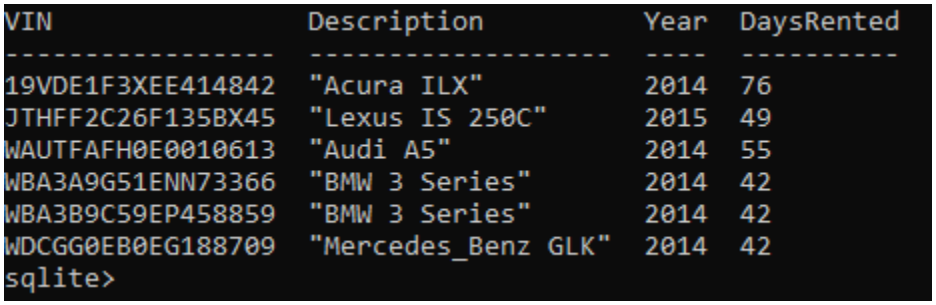
INSERT INTO RATE VALUES(6,1,800,135);



No records were affected but there were 12 records returned due to the addition of the two rates. Also, ignore the second to last entry, I accidentally inserted a rate with the information of the 10th rate.

Question 5: Return all Compact(1) & Luxury(1) vehicles that were available for rent from June 01, 2019 until June 20, 2019. List VechicleID as VIN, Description, year, and how many days have been rented so far. You need to change the weeks into days.

```
SELECT v.VehicleID AS VIN, v.Description, v.year, SUM(r.RentalType*r.Qty) as DaysRented
FROM VEHICLE v, RENTAL r
WHERE v.VehicleID = r.VehicleID AND (v.Type = 1 OR v.Category = 1) AND ((r.StartDate >= '2019-06-01') OR (r.StartDate <= '2019-06-20'))
GROUP BY r.VehicleID;
```



VIN	Description	Year	DaysRented
19VDE1F3XEE414842	"Acura ILX"	2014	76
JTHFF2C26F135BX45	"Lexus IS 250C"	2015	49
WAUTFAFH0E0010613	"Audi A5"	2014	55
WBA3A9G51ENN73366	"BMW 3 Series"	2014	42
WBA3B9C59EP458859	"BMW 3 Series"	2014	42
WDCGG0EB0EG188709	"Mercedes_Benz GLK"	2014	42

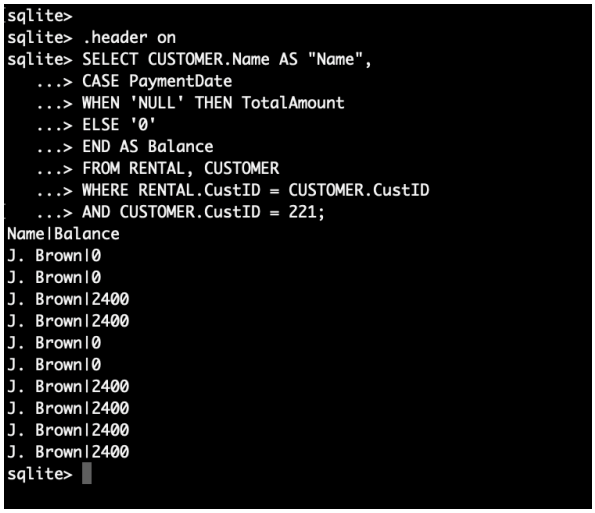
sqlite>

Records returned:6

This one was challenging because of some ambiguity, I was unsure if the question meant I should only return rental days since the period 6/01/2019-2/20/2019 or the total rental days for those vehicles or rental days after that period. I wound up returning the total rental days for the vehicles.

Question 6: Return a list with the remaining balance for the customer with the id ‘221’. List customer name, and the balance.

```
.header on
SELECT CUSTOMER.Name AS "Name",
CASE PaymentDate
WHEN 'NULL' THEN TotalAmount
ELSE '0'
END AS Balance
FROM RENTAL, CUSTOMER
WHERE RENTAL.CustID = CUSTOMER.CustID
AND CUSTOMER.CustID = 221;
```



Record returned: 10

The records for PaymentDate were affected since the heading was changed to balance using a case statement and the values were either 0 or the total amount depending on the NULL value for payment date. This was also the primary challenge for the query. We needed to check whether a rental was already paid for or not by checking the NULL value.

Question 7: Create a report that will return all vehicles. List the VehicleID as VIN, Description, Year,Type, Category, and Weekly and Daily rates. For the vehicle Type and Category, you need to use theSQL Case statement to substitute the numbers with text. Order your results based on Category (firstLuxury and then Basic) and Type based on the Type number, not the text.

```
SELECT V.VehicleID VIN, V.Description, V.Year, CASE  WHEN  V.Category = 0 THEN 'Basic'
WHEN V.Category = 1 THEN 'Luxury' END Category,

CASE WHEN V.Type = 1 THEN 'Compact' WHEN V.Type = 2 THEN 'Medium' WHEN V.Type
= 3 THEN 'Large' WHEN V.Type = 4 THEN 'SUV' WHEN V.Type = 5 THEN 'Truck' WHEN
V.Type = 6 THEN 'VAN' END Type, R.Weekly, R.Daily

FROM VEHICLE AS V, RATE AS R

WHERE V.Type = R.Type AND V.Category = R.Category

ORDER BY V.Category DESC, V.Type ASC;
```

```
VIN|Description|Year|Category|Type|Weekly|Daily
VehicleID|Description|Year|||Weekly|Daily
19VDE1F3XEE414842|"Acura ILX"|2014|Luxury|Compact|600|100
JTHFF2C26F135BX45|"Lexus IS 250C"|2015|Luxury|Compact|600|100
WAUTFAFH0E0010613|"Audi A5"|2014|Luxury|Compact|600|100
WBA3A9G51ENN73366|"BMW 3 Series"|2014|Luxury|Compact|600|100
WBA3B9C59EP458859|"BMW 3 Series"|2014|Luxury|Compact|600|100
WDCGG0EB0EG188709|"Mercedes_Benz GLK"|2014|Luxury|Compact|600|100
1VWCH7A3XEC037969|"Volkswagen Passat"|2014|Luxury|Medium|660|110
JTHBW1GG1F120DU53|"Lexus ES 300h"|2015|Luxury|Medium|660|110
JTHCE1BL3F151DE04|"Lexus GS 350"|2015|Luxury|Medium|660|110
JH4KC1F50EC800004|"Acura RLX"|2014|Luxury|Large|710|120
JH4KC1F56EC000095|"Acura RLX"|2014|Luxury|Large|710|120
```

No records were affected by the query, but all the records in the vehicle table were returned. A challenge faced on this query was figuring out the renaming and case statements, because when the query was made the headers would be blank. This was fixed by removing some ‘AS’ statements that I was using.

Question 8: What is the total of money that customers paid to us until today?

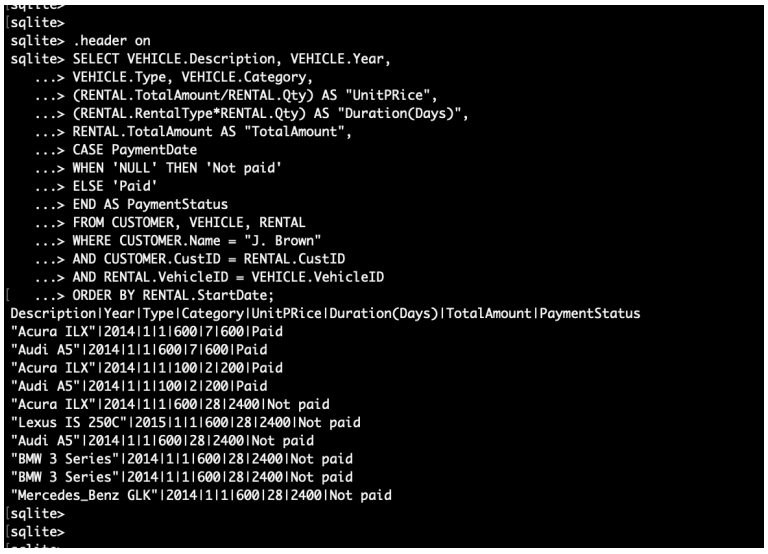
```
SELECT SUM(TotalAmount) as TotalPaid
FROM RENTAL;
```

Records returned: 1

Its unclear if TotalAmount is total paid or total owed.

Question 9-a:Create a report for the J. Brown customer with all vehicles he rented. List the description,year, type, and category. Also, calculate the unit price for every rental, the total duration mentioned if it is on weeks or days, the total amount, and if there is any payment. Similarly, as in Question 7, you need to change the numeric values to the corresponding text. Order the results by the StartDate.

```
.header on
SELECT VEHICLE.Description, VEHICLE.Year,
VEHICLE.Type, VEHICLE.Category,
(RENTAL.TotalAmount/RENTAL.Qty) AS "UnitPrice",
(RENTAL.RentalType*RENTAL.Qty) AS "Duration(Days)",
RENTAL.TotalAmount AS "TotalAmount",
CASE PaymentDate
WHEN 'NULL' THEN 'Not paid'
ELSE 'Paid'
END AS PaymentStatus
FROM CUSTOMER, VEHICLE, RENTAL
WHERE CUSTOMER.Name = "J. Brown"
AND CUSTOMER.CustID = RENTAL.CustID
AND RENTAL.VehicleID = VEHICLE.VehicleID
ORDER BY RENTAL.StartDate;
```



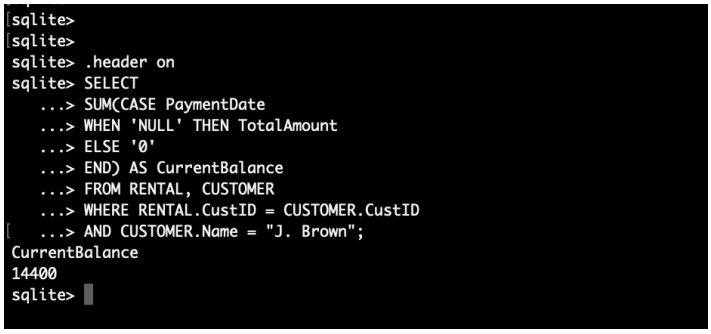
```
sqlite>
sqlite> .header on
sqlite> SELECT VEHICLE.Description, VEHICLE.Year,
   ...> VEHICLE.Type, VEHICLE.Category,
   ...> (RENTAL.TotalAmount/RENTAL.Qty) AS "UnitPRice",
   ...> (RENTAL.RentalType*RENTAL.Qty) AS "Duration(Days)",
   ...> RENTAL.TotalAmount AS "TotalAmount",
   ...> CASE PaymentDate
   ...> WHEN 'NULL' THEN 'Not paid'
   ...> ELSE 'Paid'
   ...> END AS PaymentStatus
   ...> FROM CUSTOMER, VEHICLE, RENTAL
   ...> WHERE CUSTOMER.Name = "J. Brown"
   ...> AND CUSTOMER.CustID = RENTAL.CustID
   ...> AND RENTAL.VehicleID = VEHICLE.VehicleID
   ...> ORDER BY RENTAL.StartDate;
Description|Year|Type|Category|UnitPRice|Duration(Days)|TotalAmount|PaymentStatus
"Acura ILX"|2014|1|1|600|7|600|Paid
"Audi A5"|2014|1|1|600|7|600|Paid
"Acura ILX"|2014|1|1|100|2|200|Paid
"Audi A5"|2014|1|1|100|2|200|Paid
"Acura ILX"|2014|1|1|600|28|2400|Not paid
"Lexus IS 250C"|2015|1|1|600|28|2400|Not paid
"Audi A5"|2014|1|1|600|28|2400|Not paid
"BMW 3 Series"|2014|1|1|600|28|2400|Not paid
"BMW 3 Series"|2014|1|1|600|28|2400|Not paid
"Mercedes_Benz GLK"|2014|1|1|600|28|2400|Not paid
sqlite>
sqlite>
```

Record returned: 10

The primary challenge for the query was outputting the unitprice, duration and payment status. The unit price was found using the total amount and quantity. Duration had to be done in days by multiplying the rental type and quantity. A case statement had to be used to check if there was a payment date and return the status of the payment

Question 9-b:For the same customer return the current balance.

```
.header on
SELECT
SUM(CASE PaymentDate
WHEN 'NULL' THEN TotalAmount
ELSE '0'
END) AS CurrentBalance
FROM RENTAL, CUSTOMER
WHERE RENTAL.CustID = CUSTOMER.CustID
AND CUSTOMER.Name = "J. Brown";
```



Record returned: 1

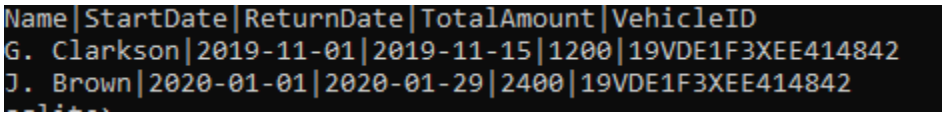
No records were affected. The main challenge with the query was determining whether the current balance should be returned as a list or as a sum. Once we figured out it was the current balance as a whole, we had to use an aggregate function which took a while to get it right.

Question 10: Retrieve all weekly rentals for the vehicleID ‘19VDE1F3XEE414842’ that are not paid yet. List the Customer Name, the start and return date, and the amount.

```
SELECT C.Name,R.StartDate, R.ReturnDate, R.TotalAmount, R.VehicleID

FROM CUSTOMER AS C, RENTAL AS R

WHERE RentalType=7 AND VehicleID = '19VDE1F3XEE414842' AND PaymentDate =
'NULL' AND C.CustID=R.CustID;
```



No records were affected as a result of the query, but all of the records that have not paid for the said vehicle were displayed. A challenge I faced on this query or more of a confusion I should say was that I thought the rental type was referring to the car type, but then I checked again, and rental type was if the car is weekly or daily, so I was trying to relate rental to vehicle in some way.

Question 11:Return all customers that they never rent a vehicle.

```
SELECT c.CustID, c.Name, c.Phone
FROM CUSTOMER c
WHERE c.CustID NOT IN(
                SELECT r.CustID
                FROM RENTAL r);
```

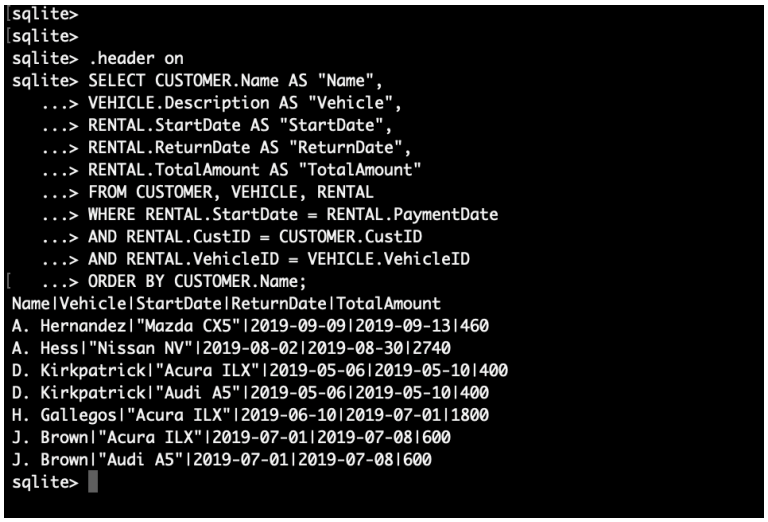
```
CustID  Name             Phone
------  ---------------  --------------
201     A. Parks         (214) 555-0127
202     S. Patel         (849) 811-6298
204     G. Carver        (753) 763-8656
205     Sh. Byers        (912) 925-5332
206     L. Lutz          (931) 966-1775
207     L. Bernal        (884) 727-0591
208     I. Whyte         (811) 979-7345
209     L. Lott          (954) 706-2219
211     Sh. Dunlap       (604) 581-6642
213     L. Perkins       (317) 996-3104
214     M. Beach         (481) 422-0282
215     C. Pearce        (599) 881-5189
217     M. Lee           (369) 898-6162
218     R. Booker        (730) 784-6303
219     A. Crowther      (325) 783-4081
220     H. Mahoney       (212) 262-8829
222     H. Stokes        (931) 969-7317
223     J. Reeves        (940) 981-5113
224     A. Mcghee        (838) 610-5802
225     L. Mullen        (798) 331-7777
226     R. Armstrong     (325) 783-4081
227     J. Greenaway     (212) 262-8829
228     K. Kaiser Acosta (228) 576-1557
230     A. Odonnell      (439) 536-8929
231     K. Kay           (368) 336-5403
0       Josh Burkey      (837) 721-8965
sqlite>
```

Records returned: 26

Count is plus 1 due to the insert and updates from Q1 and Q2

Question 12: Return all rentals that the customer paid on the StartDate. List Customer Name, VehicleDescription, StartDate, ReturnDate, and TotalAmount. Order by Customer Name.

```
.header on
SELECT CUSTOMER.Name AS "Name",
VEHICLE.Description AS "Vehicle",
RENTAL.StartDate AS "StartDate",
RENTAL.ReturnDate AS "ReturnDate",
RENTAL.TotalAmount AS "TotalAmount"
FROM CUSTOMER, VEHICLE, RENTAL
WHERE RENTAL.StartDate = RENTAL.PaymentDate
AND RENTAL.CustID = CUSTOMER.CustID
AND RENTAL.VehicleID = VEHICLE.VehicleID
ORDER BY CUSTOMER.Name;
```



Record returned: 7
No records were affected.

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