Project 2 Part 3 - Huy Nguyen and Christopher Nguyen

CSE 3330 Section 003

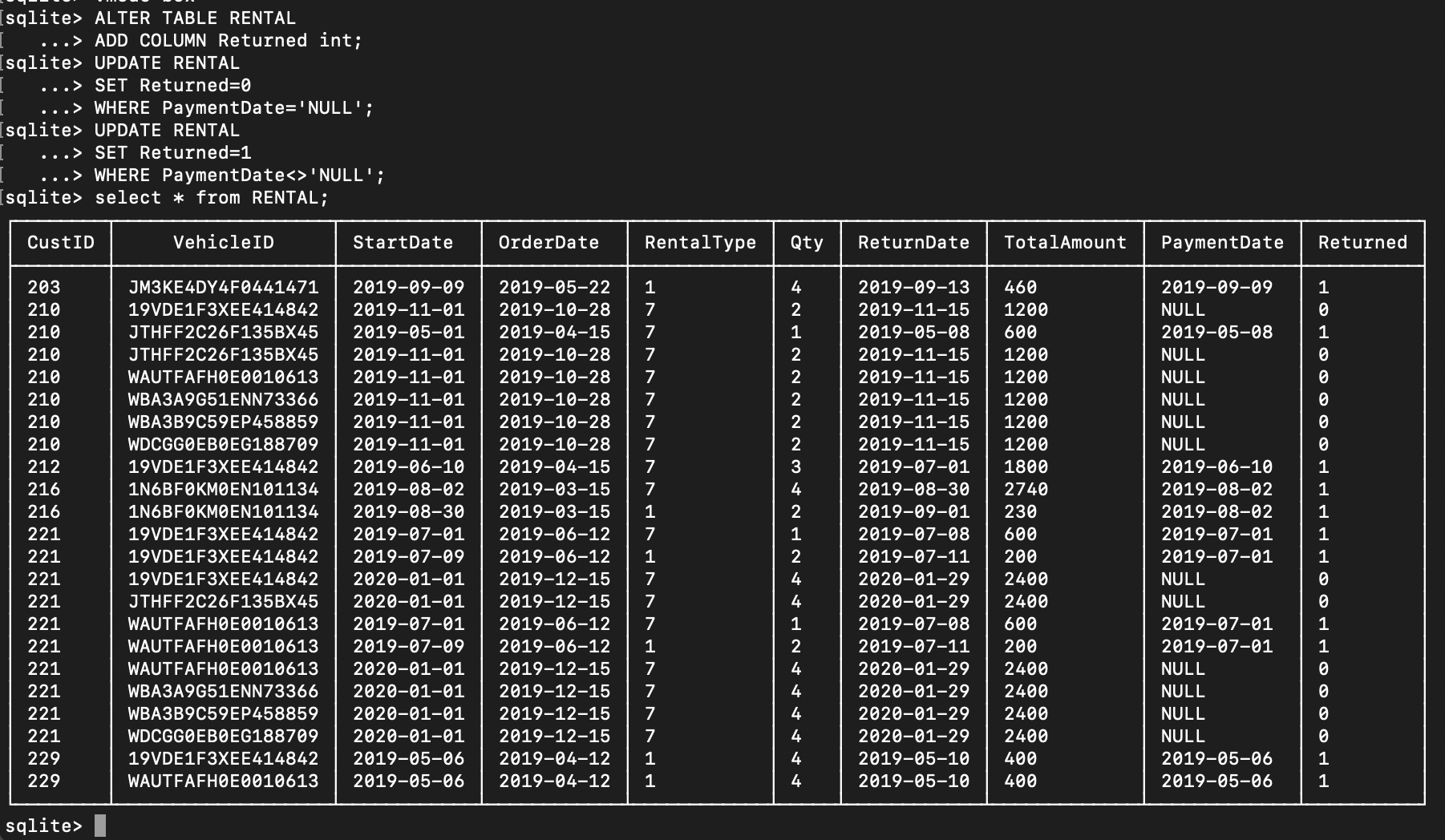
In the front-page, type a title for your project submission and specify your name. On the second page, include your honor code. Failing to do so will cost [10 points]

| **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. |

Task 1: Execute the following queries on the CarRental2019 database tables:

Query 1: [5 points]

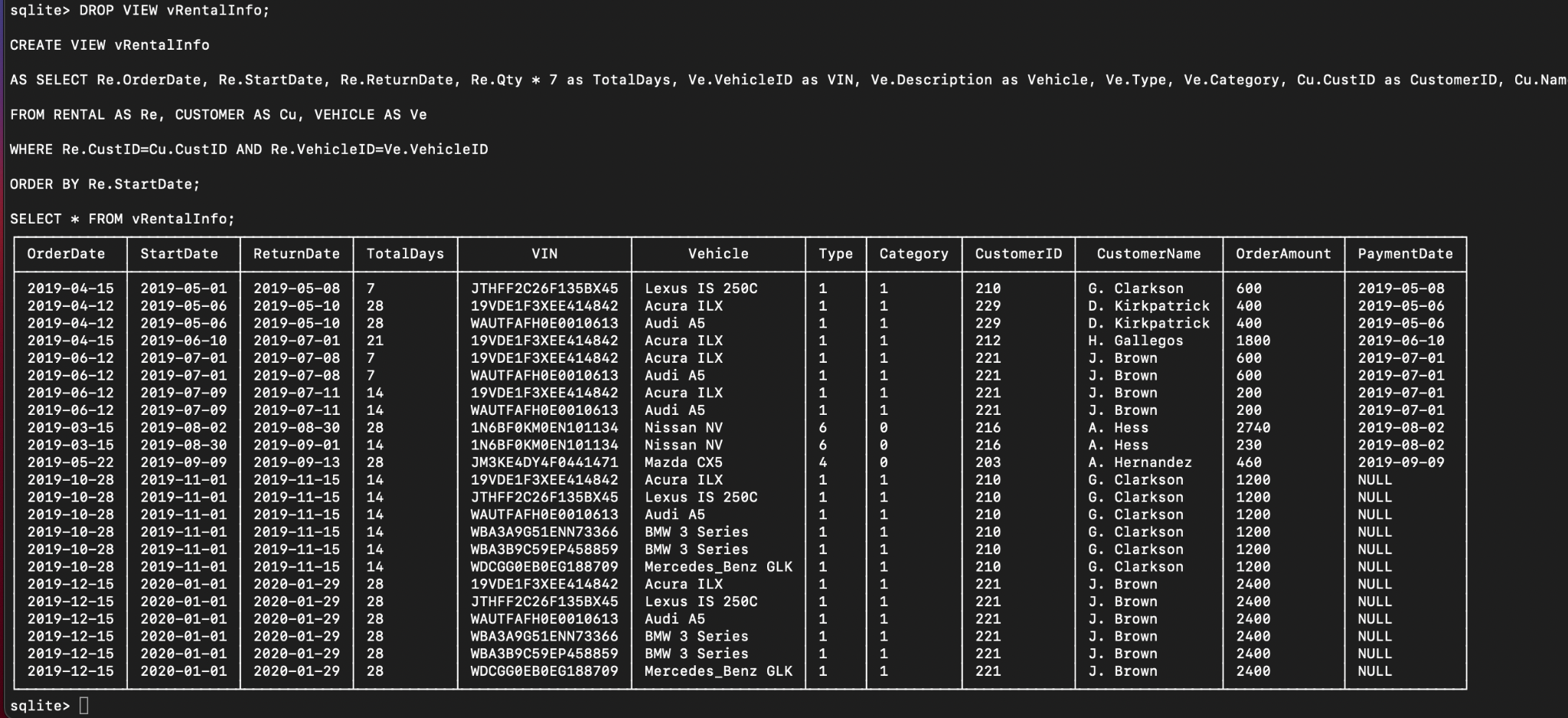
| ALTER TABLE RENTAL  ADD COLUMN Returned int;  UPDATE RENTAL  SET RENTAL.Returned=0  WHERE RENTAL.PaymentDate=’NULL’;  UPDATE RENTAL  SET RENTAL.Returned=0  WHERE RENTAL.PaymentDate<>’NULL’; |
| --- |



Query 2:

| DROP VIEW vRentalInfo;  CREATE VIEW vRentalInfo  AS SELECT Re.OrderDate, Re.StartDate, Re.ReturnDate, Re.Qty \* 7 as TotalDays, Ve.VehicleID as VIN, Ve.Description as Vehicle, Ve.Type, Ve.Category, Cu.CustID as CustomerID, Cu.Name as CustomerName, Re.TotalAmount as OrderAmount, Re.PaymentDate  FROM RENTAL AS Re, CUSTOMER AS Cu, VEHICLE AS Ve  WHERE Re.CustID=Cu.CustID AND Re.VehicleID=Ve.VehicleID  ORDER BY Re.StartDate;  SELECT \* FROM vRentalInfo; |
| --- |

NOTE: DROP VIEW is assumed to be executed if and only if there is an existing view



Task 2: Create a GUI for the CarRental2019 database:

1.The first requirement is to add information about a new customer. Do not provide the customer ID in your query. Submit your editable SQL query that your code executes.

def insert\_customer():

inC = sqlite3.connect('carrental.db')

inC\_cur = inC.cursor()

inC\_cur.execute("INSERT INTO CUSTOMER(CustID, Name, Phone) VALUES(?,?,?)",customer\_id.get(),customer\_name.get(),customer\_phone.get())

inC.commit()

inC.close()

2. The second requirement is to add all the information about a new vehicle. Submit your editable SQL query that your code executes.

def insert\_vehicle():

inV = sqlite3.connect('carrental.db')

inV\_cur = inV.cursor()

inV\_cur.execute("INSERT INTO VEHICLE VALUES(:VehicleID, :Description, :Year, :Type, :Category)",

{

'VehicleID': vehicle\_vehicleID.get(),

'Description': vehicle\_description.get(),

'Year': vehicle\_year.get(),

'Type': vehicle\_type.get(),

'Category': vehicle\_category.get()

})

inV.commit()

inV.close()

3. The third requirement is to add information of a new rental reservation

def list\_view\_customer\_rental():

liV = sqlite3.connect('carrental.db')

liV\_cur = liV.cursor()

liV\_cur.execute("CREATE VIEW vRentalInfo AS SELECT Re.OrderDate, Re.StartDate, Re.ReturnDate, Re.Qty \* 7 as TotalDays, Ve.VehicleID as VIN, Ve.Description as Vehicle, Ve.Type, Ve.Category, Cu.CustID as CustomerID, Cu.Name as CustomerName, Re.TotalAmount as OrderAmount, Re.PaymentDate FROM RENTAL AS Re, CUSTOMER AS Cu, VEHICLE AS Ve WHERE Re.CustID=Cu.CustID AND Re.VehicleID=Ve.VehicleID ORDER BY Re.StartDate")

liV\_cur.execute("SELECT SUM(OrderAmount) FROM vRentalInfo WHERE (CustomerID=? OR CustomerName=?) AND PaymentDate='NULL'", (customer\_id.get(), customer\_name.get(),))

output\_records = liV\_cur.fetchall()

print\_record = ''

for output\_record in output\_records:

print\_record += str(str(output\_record[0])+"\n")

liV\_label = Label(root, text = print\_record)

liV\_label.grid(row=21, column=5, columnspan=2)

#PLEASE NOTE THAT PYTHON AND SQL DOES NOTHING WITH dropping the views on exit, so we have to manually do this ourselves

liV\_cur.execute("DROP VIEW vRentalInfo")

liV.commit()

liV.close()

4. The fourth requirement is to handle the return of a new car

def list\_view\_vehicle():

liVV = sqlite3.connect('carrental.db')

liVV\_cur = liVV.cursor()

liVV\_cur.execute("CREATE VIEW vRentalInfo AS SELECT Re.OrderDate, Re.StartDate, Re.ReturnDate, Re.Qty \* 7 as TotalDays, Ve.VehicleID as VIN, Ve.Description as Vehicle, Ve.Type, Ve.Category, Cu.CustID as CustomerID, Cu.Name as CustomerName, Re.TotalAmount as OrderAmount, Re.PaymentDate FROM RENTAL AS Re, CUSTOMER AS Cu, VEHICLE AS Ve WHERE Re.CustID=Cu.CustID AND Re.VehicleID=Ve.VehicleID ORDER BY Re.StartDate")

liVV\_cur.execute("SELECT VIN, Vehicle FROM vRentalInfo WHERE (Vehicle=? OR VIN=?)",(vehicle\_description.get(),vehicle\_vehicleID.get(),))

output\_records = liVV\_cur.fetchall()

print\_record = ''

for output\_record in output\_records:

print\_record += str(str(output\_record[0])+" "+str(output\_record[1])+"\n")

liVV\_label = Label(root, text = print\_record)

liVV\_label.grid(row=21, column=0, columnspan=2)

#PLEASE NOTE THAT PYTHON AND SQL DOES NOTHING WITH dropping the views on exit, so we have to manually do this ourselves

liVV\_cur.execute("DROP VIEW vRentalInfo")

liVV.commit()

liVV.close()

5. The fifth requirement is to return the view results

def list\_view\_customer\_rental():

liV = sqlite3.connect('carrental.db')

liV\_cur = liV.cursor()

#PLEASE NOTE THAT PYTHON AND SQL DOES NOTHING WITH dropping the views on exit, so we have to manually do this ourselves

liV\_cur.execute("DROP VIEW vRentalInfo")

liV\_cur.execute("CREATE VIEW vRentalInfo AS SELECT Re.OrderDate, Re.StartDate, Re.ReturnDate, Re.Qty \* 7 as TotalDays, Ve.VehicleID as VIN, Ve.Description as Vehicle, Ve.Type, Ve.Category, Cu.CustID as CustomerID, Cu.Name as CustomerName, Re.TotalAmount as OrderAmount, Re.PaymentDate FROM RENTAL AS Re, CUSTOMER AS Cu, VEHICLE AS Ve, RATE AS Ra WHERE Re.CustID=Cu.CustID AND Re.VehicleID=Ve.VehicleID ORDER BY Re.StartDate")

#liV\_cur.execute("SELECT \* FROM vRentalInfo")

liV\_cur.execute("SELECT SUM(OrderAmount) FROM vRentalInfo WHERE (CustomerID=? OR CustomerName=?) AND PaymentDate='NULL'", (customer\_id.get(), customer\_name.get(),))

output\_records = liV\_cur.fetchall()

print\_record = ''

for output\_record in output\_records:

print\_record += str(str(output\_record[0])+"\n")

liV\_label = Label(root, text = print\_record)

liV\_label.grid(row=12, column=0, columnspan=2)

liV.commit()

liV.close()

def list\_view\_vehicle():

liVV = sqlite3.connect('carrental.db')

liVV\_cur = liVV.cursor()

#PLEASE NOTE THAT PYTHON AND SQL DOES NOTHING WITH dropping the views on exit, so we have to manually do this ourselves

liVV\_cur.execute("DROP VIEW vRentalInfo")

liVV\_cur.execute("CREATE VIEW vRentalInfo AS SELECT Re.OrderDate, Re.StartDate, Re.ReturnDate, Re.Qty \* 7 as TotalDays, Ve.VehicleID as VIN, Ve.Description as Vehicle, Ve.Type, Ve.Category, Cu.CustID as CustomerID, Cu.Name as CustomerName, Re.TotalAmount as OrderAmount, Re.PaymentDate FROM RENTAL AS Re, CUSTOMER AS Cu, VEHICLE AS Ve WHERE Re.CustID=Cu.CustID AND Re.VehicleID=Ve.VehicleID ORDER BY Re.StartDate")

liVV\_cur.execute("SELECT VIN, Vehicle FROM vRentalInfo WHERE (Vehicle=? OR VIN=?)",(vehicle\_description.get(),vehicle\_vehicleID.get(),))

output\_records = liVV\_cur.fetchall()

print\_record = ''

for output\_record in output\_records:

print\_record += str(str(output\_record[0])+" "+str(output\_record[1])+"\n")

liVV\_label = Label(root, text = print\_record)

liVV\_label.grid(row=13, column=0, columnspan=2)

liVV.commit()

liVV.close()

Contributions:

Christopher Nguyen - Task 1, Task 2, Task 3

Huy Nguyen - Task 1, Task 2, Task 3