**Assignment 1**

Please work in **groups** to complete this lab. This assignment is worth **15% of the total course grade** and will be evaluated through your written submission. Each day being late will result in 10% mark penalty.

Please submit the following files through Blackboard:

* Assign1.docx
* All stored procedures and trigger names should end with your group number ‘grpX’.

1. Add this declaration on the top of your Assign1.docx file.

We, Mansoor Ahmad Zafar declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. No part of this assignment has been copied manually or electronically from any other source (including web sites) **or distributed to other students.**

1. Specify what each member has done towards the completion of this work:

|  |  |  |
| --- | --- | --- |
|  | Name | Task(s) |
| 1 | Mansoor Ahmad Zafar | Everything |
| 2 |  |  |
| 3 |  |  |

# Before you start

You are to create a new database named “Aviaco” and run the sql script you are given to create the tables in the database.

Add INV\_TOTAL column to the invoice table by running the following command:

update table customer

set cus\_balance =0;

# Part 1: Stored Procedures (30 pts)

**Q1. (10 pts)**

Create a stored procedure named updateAllCustomerBalances\_grpX that updates the customer balance in the customer table for all the customers. The customer balance is calculated as the sum of all charter charges booked by a customer, as CHAR\_DISTANCE \* MOD\_CHG\_MILE.

create procedure updateAllCustomerBalance\_grp11

as

update customer

set cus\_balance = (select sum(charter.char\_distance\*model.MOD\_CHG\_MILE)

from charter

join aircraft on aircraft.AC\_NUMBER = charter.AC\_NUMBER

join model on model.MOD\_CODE = aircraft.MOD\_CODE

where charter.CUS\_CODE = customer.CUS\_CODE)

go

**Testing and result set:**

A screenshot of a computer

Description automatically generated

**Q2. (20 pts)**

Linear regression is a predictive analytics technique that aims at modeling a linear relationship between variables used for prediction. Given the distance flown and fuel consumption data, we can derive a linear equation that can be used to predict the fuel consumption for any given distance flown.

The linear regression model is represented with a simple equation y= b0 +b1\*x.

For the distance/fuel consumption data, the linear regression model is represented as

The model is defined when b0 and b1 are set. Given distance/fuel consumption data retrieved from the charter table, the goal is to calculate b0 and b1.

,

Create a stored procedure named predictfuel\_grpX with no input parameters that gets the distance-fuel data from the charter table, calculates b0 and b1 and print the equation in the form

Below.

**Pseudocode:**

* declare a cursor for looping on distance-fuel result set.
* declare the variables @b0, @b1
* declare @numerator, @denom that will help calculating @b1
* loop through the cursor to accumulate @numerator and @denom
* once the loop is over, calculate @b1 as @numerator/@denom
* calculate b0
* print the equation in the form fuel= b0 + b1\*distance

create procedure predictfuel\_grp11

as

declare ptr cursor

for select char\_distance, CHAR\_FUEL\_GALLONS from charter

declare @distance real

declare @distance\_fuel real

declare @b0 real

declare @b1 real

declare @numerator real

set @numerator = 0

declare @denom real

set @denom = 0

declare @avgDistance real

set @avgDistance = (select avg(char\_distance) from charter)

declare @avgFuel real

set @avgFuel = (select avg(CHAR\_FUEL\_GALLONS) from charter)

open ptr

fetch next from ptr into @distance, @distance\_fuel

while @@fetch\_status=0

begin

set @numerator = @numerator + ((@distance - @avgDistance) \* (@distance\_fuel - @avgFuel))

set @denom = @denom + ((@distance-@avgDistance) \* (@distance-@avgDistance))

fetch next from ptr into @distance, @distance\_fuel

end

close ptr

deallocate ptr

set @b1 = @numerator/@denom

set @b0 = @avgFuel - (@b1\*@avgDistance)

print concat('fuel = ', @b0, ' + ', @b1, ' \* distance')

go

**Testing and result set:**

A screenshot of a computer

Description automatically generated

# Part 2: Triggers (40 pts)

**Q3. (10 pts)**

Create a trigger named checkCrew that fires with a 'instead of' statement when crew records are inserted in the crew table.

When there is already a crew with the same crew job on the same charter for one of the inserted rows, print a message that says: ‘There is already the same job on one of the trips’. Otherwise, insert the rows from inserted in the crew table. See the testing commands below.

create Trigger checkCrew\_grp11

on crew

instead of insert

as

begin

declare @job varchar(20)

declare @trip int

declare @empnum int

declare ptr cursor

for select i.CREW\_JOB, i.EMP\_NUM, i.CHAR\_TRIP from inserted i

open ptr

fetch next from ptr into @job, @empnum, @trip

while @@fetch\_status=0

begin

if @job in (select c.crew\_job from crew c where c.CHAR\_TRIP = @trip)

begin

print 'There is already the same job on one of the trips.'

end

else

begin

insert into crew values(@trip, @empnum, @job);

end

fetch next from ptr into @job, @empnum, @trip

end

close ptr

deallocate ptr

--Made to work with mutli insert

end

**Testing and output:**

Show the top 5 rows from the crew table:

select \* from crew order by CHAR\_TRIP;

A screenshot of a computer

Description automatically generated

**Testing 1:**

insert into crew values (10002, 104, 'Copilot');

output: Show the top 5 rows from the crew table.

The insert should go through and add the highlighted row as shown below.

**A screenshot of a data

Description automatically generated**

**Testing 2:**

insert into crew values (10003, 109, 'Pilot');

A screenshot of a computer

Description automatically generated

**Q4. (10 pts)**

Create a trigger named addRating on EARNEDRATING table that adds the inserted rating code to PIL\_RATINGS in the PILOT table.

create Trigger addRating\_grp11

on earnedrating

after insert

as

begin

declare @empN int

declare @rtg varchar(5)

declare @date datetime

declare ptr cursor for select emp\_num,rtg\_code, earnrtg\_date from inserted

open ptr

fetch next from ptr into @empN, @rtg, @date

while @@fetch\_status=0

begin

update pilot

set PIL\_RATINGS = concat(PIL\_RATINGS, '/', @rtg)

where pilot.EMP\_NUM = @empN

fetch next from ptr into @empN, @rtg, @date

end

--Insert into the pilot table edit

--rollback; -- so I dont keep having to redo this..............

end

**Before trigger:**

select \* from EARNEDRATING;

output: top 8 rows

A screenshot of a computer

Description automatically generated

select \* from PILOT;

output:

A screenshot of a computer

Description automatically generated

**Fire the trigger:**

insert into EARNEDRATING values (101, 'SES', '2021-01-01')

**After trigger:**

select \* from EARNEDRATING;

output: top 8 rows

insert went through, highlighted row is added

A screenshot of a computer

Description automatically generated

select \* from PILOT;

output:

SES is appended to PIL\_RATINGS

A screenshot of a computer

Description automatically generated

**Q5. (20 pts)**

Create a trigger named update\_cusbalance that fires after an insert event in the charter table. The customer balance should be incremented by the charter charges calculated as CHAR\_DISTANCE (given in inserted) \* MOD\_CHG\_MILE (from the model table given the AC\_number in inserted).

create Trigger update\_cusbalance\_grp11

on charter

after insert

as

begin

declare ptr cursor for

select char\_trip, ac\_number, char\_distance, cus\_code from inserted order by cus\_code

declare @cTrip int -- chartrip

declare @acnum varchar(5) -- ac\_number

declare @distance real -- char\_distance

declare @cCode int -- cus\_code

open ptr

fetch next from ptr into @cTrip, @acnum, @distance, @cCode

while @@fetch\_status=0

begin

update customer

set cus\_balance = cus\_balance + (select (@distance \* m.MOD\_CHG\_MILE)

from model m

join aircraft a on a.MOD\_CODE = m.MOD\_CODE

where a.AC\_NUMBER = @acnum)

where cus\_code = @cCode

fetch next from ptr into @cTrip, @acnum, @distance, @cCode

end

close ptr

deallocate ptr

end

**Testing scenario 1: one-row insert**

**Before trigger:**

select \* from customer where cus\_code= 10010;

A screenshot of a computer

Description automatically generated

**Fire the trigger:**

insert into CHARTER values(10019, '2012-02-09', '2778V', 'MQY', 312, 1.5, 0, 67.2, 0, 10010);

**After the trigger:**

select \* from charter where char\_trip= 10019;

output:



select \* from customer where cus\_code= 10010;

output: top 5 rows. cus\_balance updated.

A white background with black text

Description automatically generated

**Testing scenario 2: multi-row insert, with diff cus\_code**

**Before trigger:**

select \* from customer where cus\_code in (10011, 10012);

output:

**A close up of a screen

Description automatically generated**

**Fire the trigger:**

insert into CHARTER values

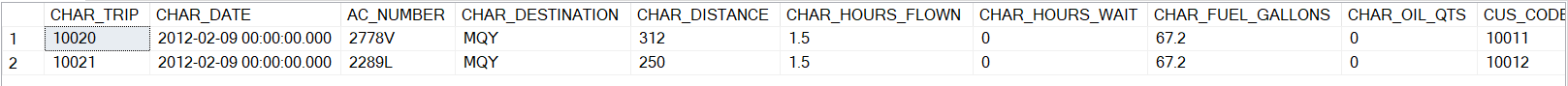
(10020, '2012-02-09', '2778V', 'MQY', 312, 1.5, 0, 67.2, 0, 10011),

(10021, '2012-02-09', '2289L', 'MQY', 250, 1.5, 0, 67.2, 0, 10012);

**After trigger:**

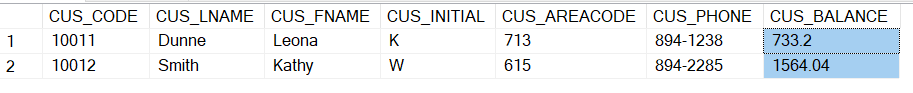
select \* from charter where char\_trip in (10020, 10021);

output:

****

select \* from customer where cus\_code in (10011, 10012);

output:



**Testing scenario 3: multi-row insert, with same cus\_code**

**Before trigger:**

select \* from customer where cus\_code=10016;

output:

A screenshot of a computer

Description automatically generated

**Fire the trigger:**

insert into CHARTER values

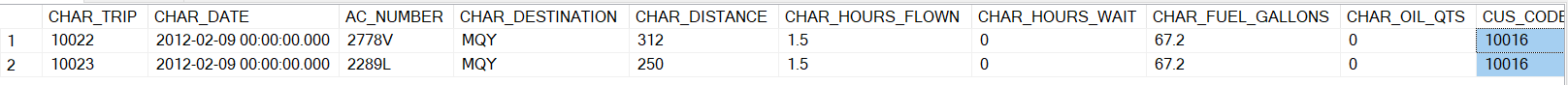
(10022, '2012-02-09', '2778V', 'MQY', 312, 1.5, 0, 67.2, 0, 10016),

(10023, '2012-02-09', '2289L', 'MQY', 250, 1.5, 0, 67.2, 0, 10016);

**After the trigger:**

select \* from charter where char\_trip in (10022, 10023);

output:

****

select \* from customer where cus\_code=10016;

**A close up of a screen

Description automatically generated**

Manual calculation:

-- MOD\_CHG\_MILE= 2.35

-- CHAR\_DISTANCE= 312

-- cus\_balance increment by 733.2

-- MOD\_CHG\_MILE= 2.67

-- char\_distance= 250

-- cus\_balance increment by 667.5

-- total increment for cus 10016 is : 733.2 + 667.5