Lab 4 (2%)

Stored Procedures

topics

* Stored Procedures: <https://docs.microsoft.com/en-us/sql/relational-databases/stored-procedures/create-a-stored-procedure?view=sql-server-ver15>
* variables: <https://docs.microsoft.com/en-us/sql/t-sql/language-elements/variables-transact-sql?view=sql-server-ver15>
* sql case when statement: <https://docs.microsoft.com/en-us/sql/t-sql/language-elements/case-transact-sql?view=sql-server-ver15>
* if else: <https://docs.microsoft.com/en-us/sql/t-sql/language-elements/if-else-transact-sql?view=sql-server-ver15>
* Control of flow: <https://docs.microsoft.com/en-us/sql/t-sql/language-elements/control-of-flow?view=sql-server-ver15>
* Stored procedures, return : <https://docs.microsoft.com/en-us/sql/relational-databases/stored-procedures/return-data-from-a-stored-procedure?view=sql-server-ver15>

Group work acknowledgment

We members of Group 2 **[Kabir Narula] [Maksym Volkovynskyi**] 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.**

**Specify below what each member has done towards the completion of this work:**

|  |  |  |
| --- | --- | --- |
|  | **Name** | **Task(s)** |
| **1-** | **Kabir Narula** | **Everything** |
| **2-** | **Maksym Volkovynskyi** | **Everything** |

**We did each question individually so that we can practice better and then later compared and matched our answers…**

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.

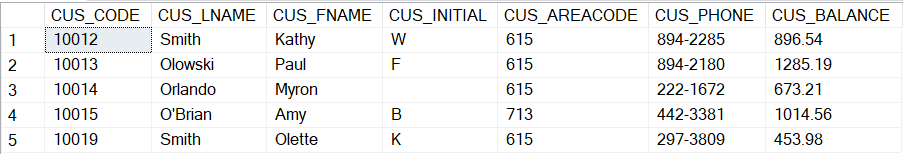
Instructions

Answer each of the following questions and show the result set underneath each part.

1. Write a stored procedure named displayCustomers\_grpX that shows customers with outstanding balances (balance > 0).

Testing and result set:

exec displayCustomers;



**create** **procedure** displayCustomers\_grp1

**as**

**select** \*

**from** CUSTOMER

**where** CUS\_BALANCE > 0;

**exec** displayCustomers\_grp1

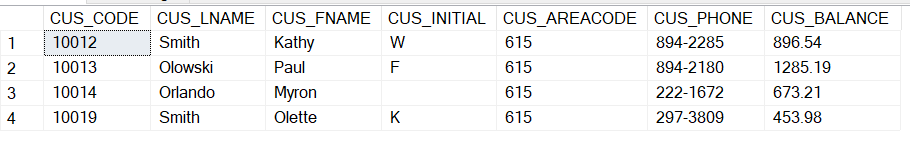
A screenshot of a computer

Description automatically generated

1. Write a stored procedure named displayCustomers\_areacode\_grpX that takes the area code as parameter and shows the customers in the given area code that have outstanding balance.

Testing and result set:

exec displayCustomers\_areacode '615';



**create** **procedure** displayCustomers\_areacode\_grp1

**@areaCode** **varchar**(3)

**as**

**select** \*

**from** CUSTOMER

**where** CUS\_BALANCE > 0 **and** CUS\_AREACODE = **@areaCode**;

**exec** displayCustomers\_areacode\_grp1 '615';

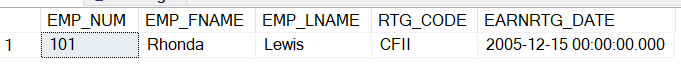
A screenshot of a computer

Description automatically generated

1. Write a stored procedure named displayPilots\_grpX that takes a rating code and year as input and shows the pilot emp num, Lname, fname, rtg\_code, earnrtg\_date who earned the rating given in input at the specified year given in input.

Testing and result set:

exec displayPilots 'CFII', 2005;



**create** **procedure** displayPilots\_grp1

**@ratingCode** **varchar**(5),

**@ratingYear** **decimal**(4)

**as**

**select** e.EMP\_NUM, e.EMP\_FNAME, e.EMP\_LNAME, r.RTG\_CODE, r.EARNRTG\_DATE

**from** EARNEDRATING r

**join** EMPLOYEE e **on** e.EMP\_NUM = r.EMP\_NUM

**where** r.RTG\_CODE = **@ratingCode** **and** **YEAR**(r.EARNRTG\_DATE) = **@ratingYear**;

**exec** displayPilots\_grp1 'CFII', 2005;

A screenshot of a computer program

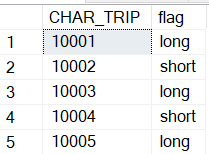
Description automatically generated

1. Write a stored procedure named tagCharters\_grpX that takes the number of hours flown @nbhours as input and label each of the charters with 'short' if the number of hours flown is less than @nbhours and 'long' otherwise.

Write the statement to execute the stored procedure by passing 5 as argument to @nbhours.

Testing and result set: top 5 of 18 rows

EXEC tagCharters @nbhours=5;



**create** **procedure** tagCharters\_grp1

**@nbhours** **decimal**(10,4)

**as**

**select** CHAR\_TRIP, **case**

**when** CHAR\_HOURS\_FLOWN < **@nbhours** **then** 'short'

**else** 'long'

**end**

**as** flag

**from** CHARTER;

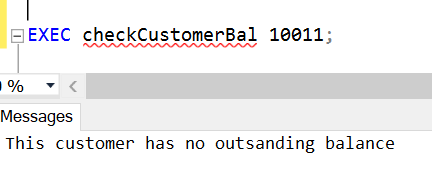
**exec** tagCharters\_grp1 **@nbhours**=5;

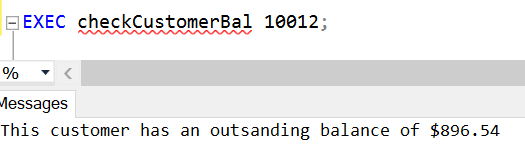
A screenshot of a computer

Description automatically generated

1. Write a stored procedure named checkCustomerBal\_grpX that takes a customer code @cus\_code as input and prints a message (not a table) of whether the customer has an outstanding balance along with the amount or no outstanding balance.  
   Write the statement to execute the stored procedure by testing with 10011 and 10012 as argument.

Result set:





**create** **procedure** checkCustomerBal\_grp1

**@cus\_code** **decimal**(10,4)

**as**

**declare** **@cus\_balance** **decimal**(24,4) = (**select** CUS\_BALANCE **from** CUSTOMER **where** CUS\_CODE = **@cus\_code**)

**if** **@cus\_balance** > 0

**print** 'The customer has an outstanding balance of $' + **CAST**(**@cus\_balance** **as** **varchar**(10))

**else**

**print** 'The customer has no outstanding balance';

**exec** checkCustomerBal\_grp1 10011;

**exec** checkCustomerBal\_grp1 10012;

A screenshot of a computer

Description automatically generated

1. Write a stored procedure named getnbChartersCus\_grpX that takes a customer code as input and returns the number of charters the customer has booked.

Use the return statement so the stored procedure returns a value.

If the customer code is not existing in the customer table, print a message 'invalid customer code' and return -1.

For testing, run the code below. @count gets assigned the return value out of the stored procedure.

DECLARE @count int;

exec @count= getnbChartersCus 10011;

print @count;

Result set:



**alter** **procedure** getnbChartersCus\_grp1

**@cusCode** **decimal**(10,4)

**as**

**declare** **@cnt** **int**

**set** **@cnt** = (

**select** **count**(\*)

**from** CUSTOMER cu

**join** CHARTER ch **on** cu.CUS\_CODE = ch.CUS\_CODE

**where** cu.CUS\_CODE = **@cusCode**

**group** **by** cu.CUS\_CODE

)

**return** **@cnt**

go

**declare** **@count** **int**

**exec** **@count** = getnbChartersCus\_grp1 10011

**print** **@count**

A screen shot of a computer

Description automatically generated

1. Write a stored procedure named checkConsumptionv1\_grpX that takes no input and show the charter trip, destination, fuel consumption (CHAR\_FUEL\_GALLONS) ,

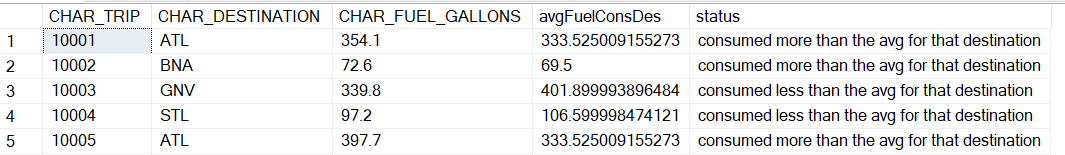
the average fuel consumption of charters of the same destination, and an additional column named 'status' with the values:

'consumed more than the average fuel consumption for that destination'

or 'consumed less than the average fuel consumption for that destination'.

Testing and result set: top 5 out of 18 rows

exec checkConsumptionv1;



**create** **procedure** checkConsumptionv1\_grp1

**as**

**select** CHAR\_TRIP, CHAR\_DESTINATION, CHAR\_FUEL\_GALLONS, (

**select** **avg**(CHAR\_FUEL\_GALLONS)

**from** CHARTER

**where** CHAR\_DESTINATION = ch.CHAR\_DESTINATION

) **as** avgFuelConsDes,

**case**

**when** CHAR\_FUEL\_GALLONS > (

**select** **avg**(CHAR\_FUEL\_GALLONS)

**from** CHARTER

**where** CHAR\_DESTINATION = ch.CHAR\_DESTINATION

) **then** 'consumed more than the avg for that destination'

**else** 'consumed less than the avg for that destination'

**end** **as** status

**from** CHARTER ch;

**exec** checkConsumptionv1\_grp1

A screenshot of a computer

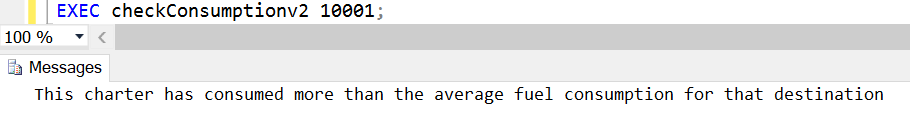
Description automatically generated

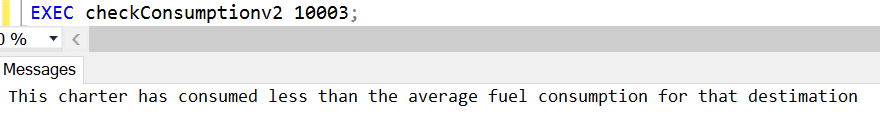
1. Write a stored procedure named checkConsumptionv2\_grpX that takes a charter trip code as input, and output a message saying :

"This charter has consumed more than the average fuel consumption compared to other charters flying to the same destination" or less otherwise.

Result set:

Test the procedure as per the following commands:





**create** **procedure** checkConsumptionv2\_grp1

**@charTripCode** **decimal**(10,4)

**as**

**declare** **@fuelCons** **decimal**(10,4)

**set** **@fuelCons** = (

**select** c.CHAR\_FUEL\_GALLONS

**from** CHARTER c

**where** c.CHAR\_TRIP = **@charTripCode**

)

**declare** **@avgFuelCons** **decimal**(10,4)

**set** **@avgFuelCons** = (

**select** **avg**(sub.fuel)

**from** (

**select** c.CHAR\_FUEL\_GALLONS **as** fuel

**from** CHARTER c

**where** c.CHAR\_DESTINATION **in** (

**select** c2.CHAR\_DESTINATION

**from** CHARTER c2

**where** c2.CHAR\_TRIP = **@charTripCode**

)

) sub

)

**if** **@fuelCons** > **@avgFuelCons**

**print** 'This charter has consumed more than the average fuel consumption for that destination'

**else**

**print** 'This charter has consumed less than the average fuel consumption for that destination'

go;

**exec** checkConsumptionv2\_grp1 10001;

**exec** checkConsumptionv2\_grp1 10003;

A screen shot of a computer

Description automatically generated

SUBMISSION

The following files should be submitted on BB:

* Lab4\_GroupX.doc file. Replace X with your group number.
* Lab4\_GroupX.sql: this file should contain
  + the question number and the question text enclosed with /\*..\*/ taken from the lab file.
  + your answer sql code underneath each question,
  + the sql statement(s) for executing the procedure with the supplied arguments as per the question specifications.

Note: If a student does NOT contribute to the work, do not list his/her name(s) under the group section in the lab file and will get 0.

Grading rubrics

Each question is worth 5pts. Total is 40 pts.

If the output is shown without the code, the answer is worth 0.