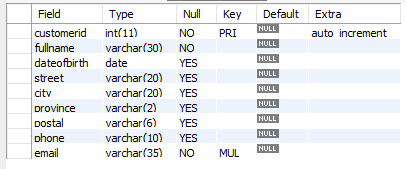
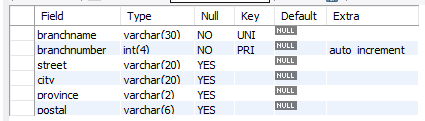
Assignment 3 SE3309 2017 clientBanking

Exercise 2:

customer

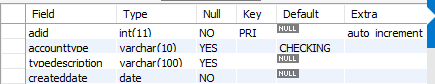


branch

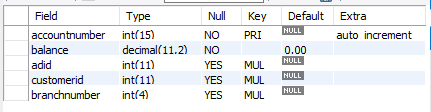


Accountdescription

Accounttype => CHECKING, SAVINGS, CREDIT, STUDENT, YOUTH



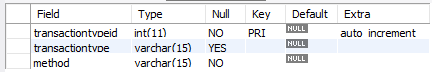
customeraccount



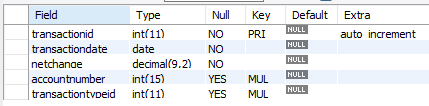
Transactiontype

Transactiontype => GOODS/MATERIALS, SERVICES, SALES, WAGES, TAX

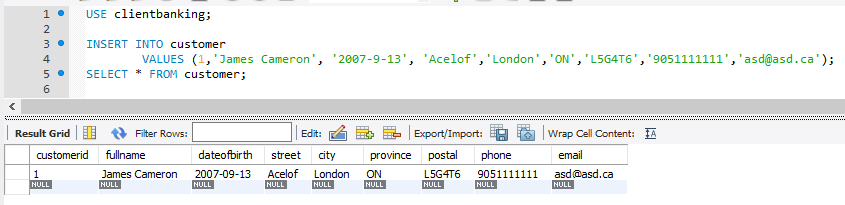
Method => CHECK, ATM, ONLINE, TRANSFER, POS

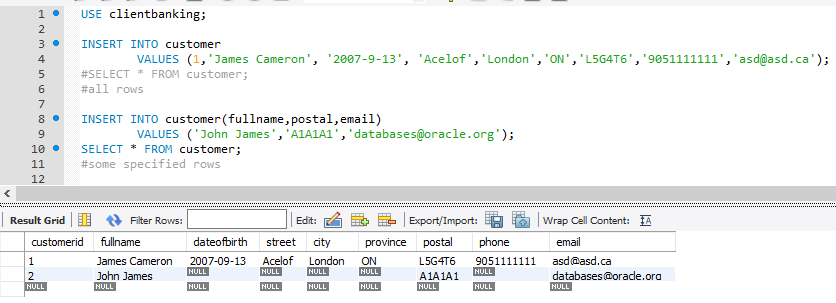


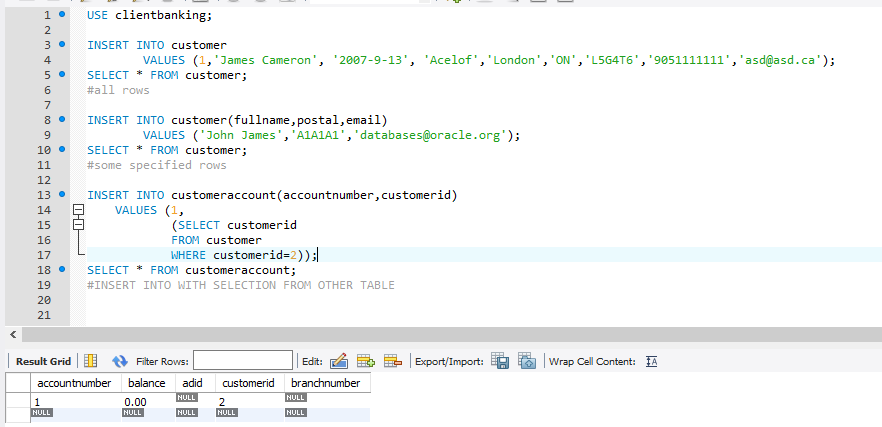
transactionhistory



Exercise 3:







Exercise 4

The code below was written in c++ and randomizes strings for each tuple, which are exported to individual .csv files for each relation. The C++ code is located in assignment3/src/FabData/FabData/FabricatedData.cpp. All of the csv files were saved in assignment3/src/FabData/FabData/ .

The foreign keys were linked based on the information below.

This code is also screenshotted in the next few pages.

The .csv files were appended with a starting row which has random information (although it should be the column headers)

Customer 2000 touples

no fk

Branch 5 touples

no fk

customeraccount 500 touples

no fk

Account Description 500 touples

first 100 customers have a random number of accounts

account has random branch from 1 to 5

customeraccountid = adid

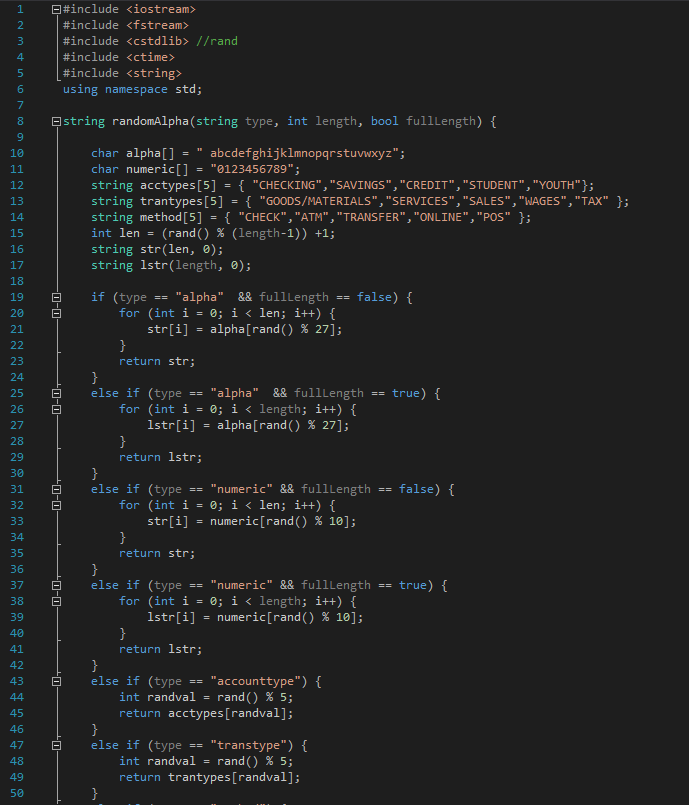
Transaction type 2000

No fk

Transaction History 2000

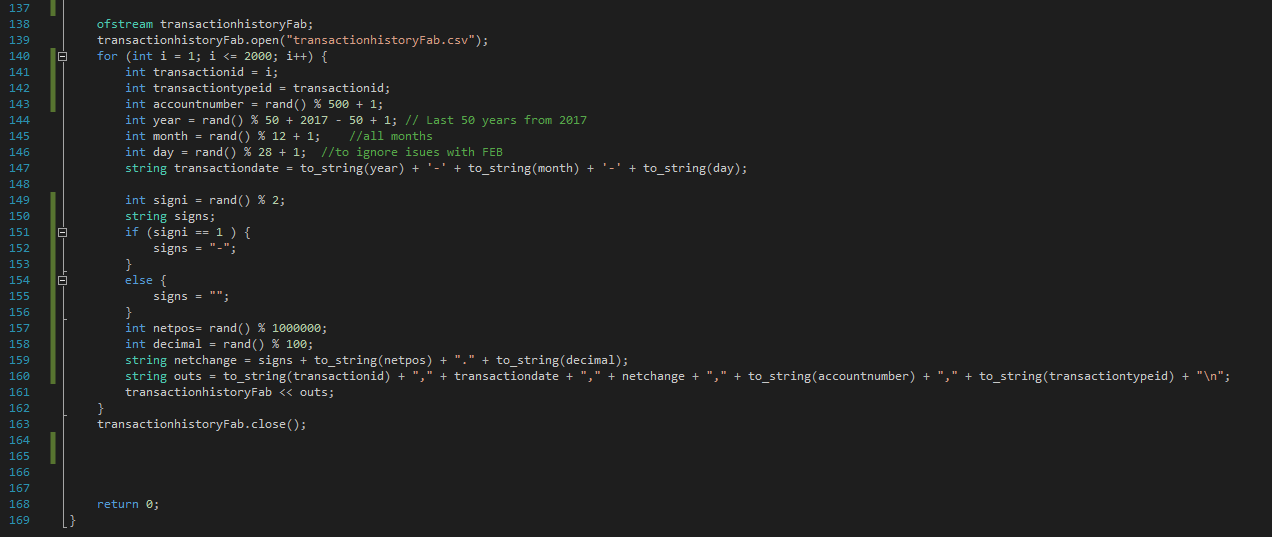
Transactionid = transactiontypeid

Customer accounts have random number of transactions



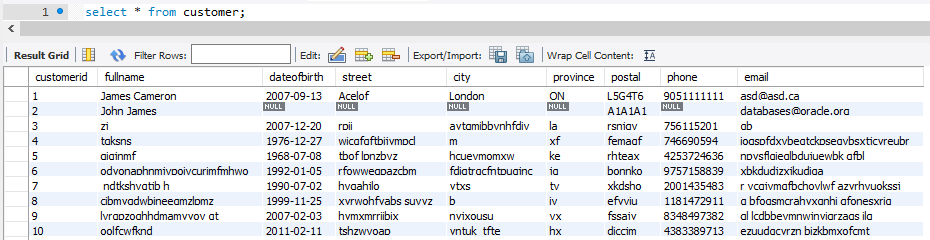




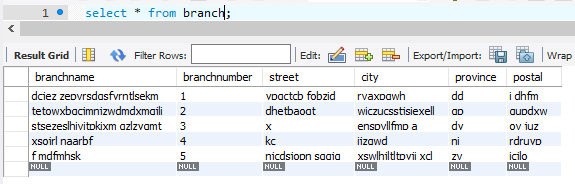


Here are a few small samples from each relation

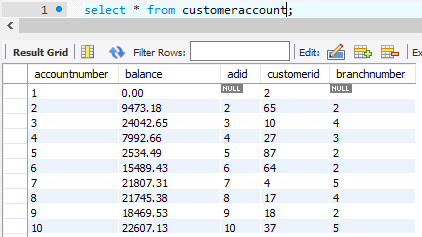
customer



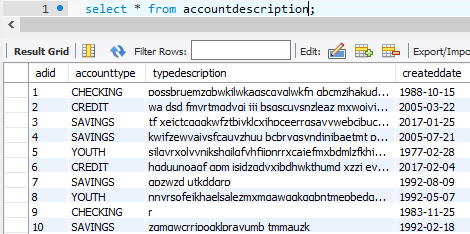
branch



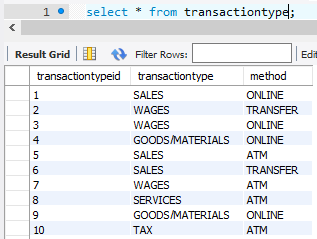
Customeraccount



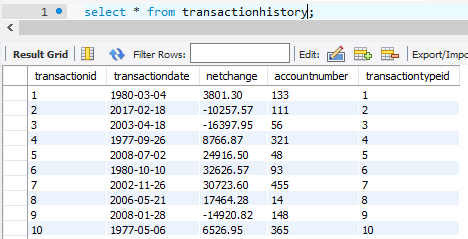
Accountdescription



transactiontype

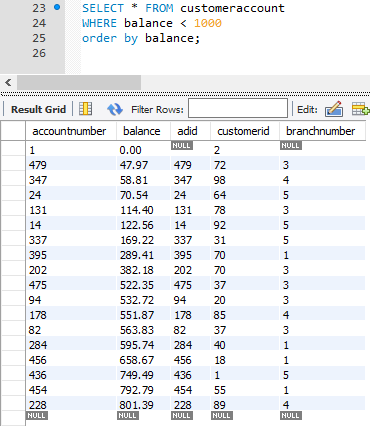


Transactionhistory

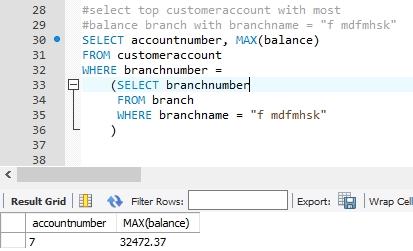


Exercise 5

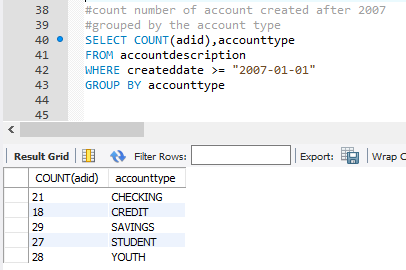
Using an aggregate less than and an order by we can find all accounts with a balance under $1000. Close to being in the red, in order



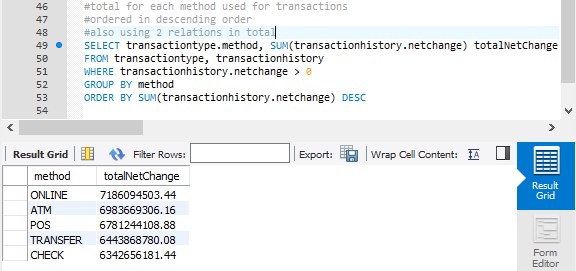
Using a subquery, we can see below,



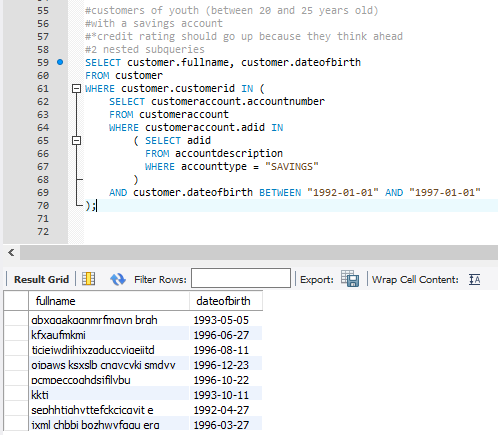
Using a grouped by and a COUNT, we can see below,



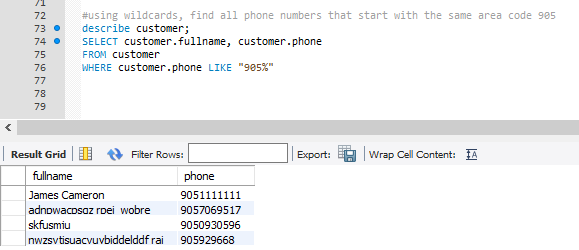
Below uses Group by, Order by, and more than 1 relation



Below has 2 nested subqueries



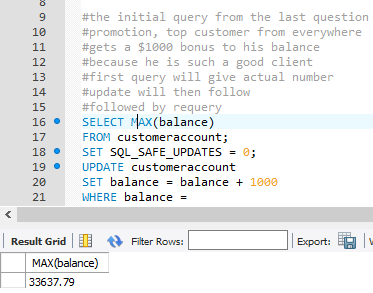
The wildcard and the LIKE operations



Ex 6

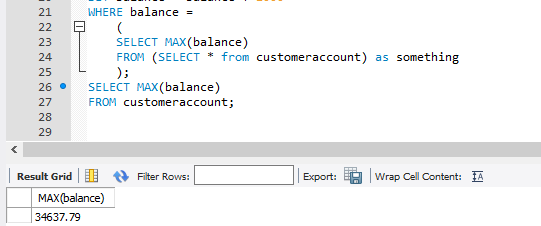
Due to various reasons, it’s kind of hard to show the response of the system when you forget to screenshot the results at the end and can’t redo anything without quite a bit of effort. I’ll screenshot the code, and whichever ones will work after applying them the second time

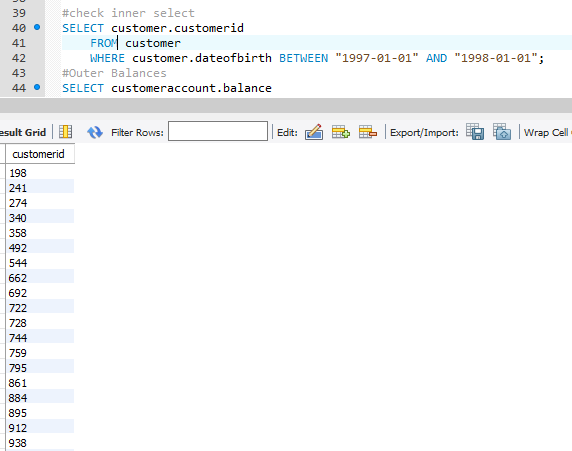
P1

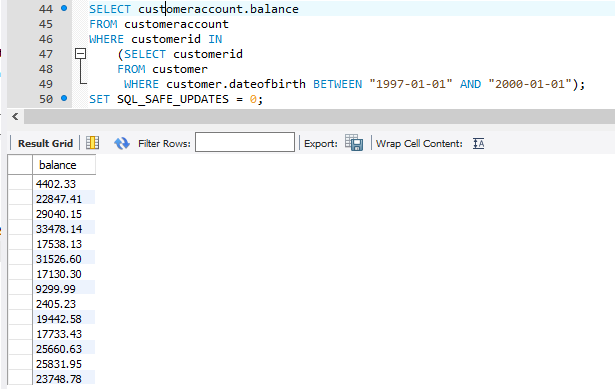


P2

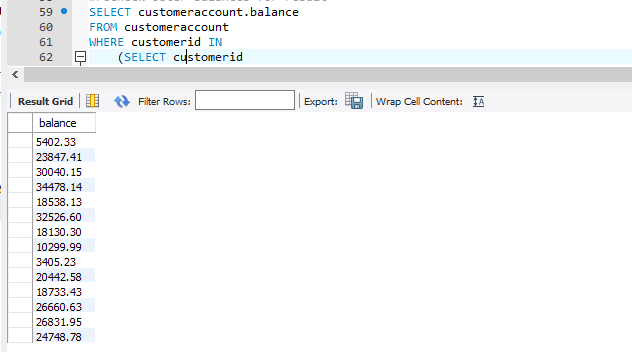


P3

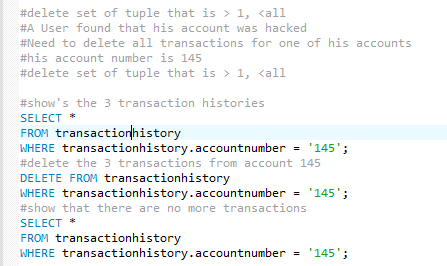
P1 returns quite a bit

P2

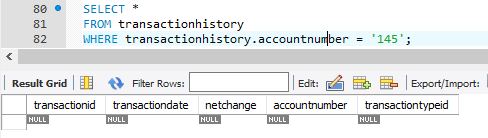
P3 

P4 

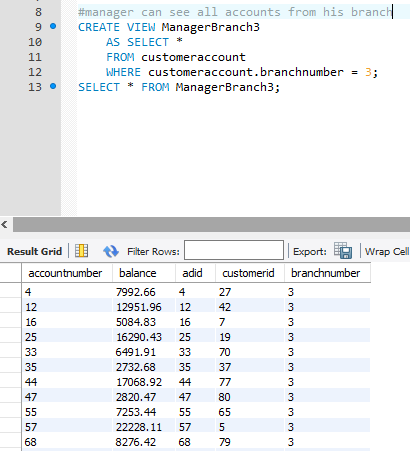
P1



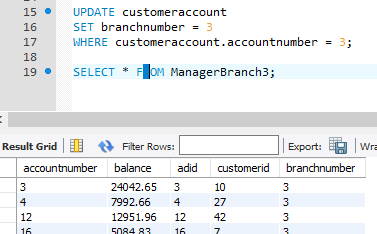
Can’t show the deletion as it already happened. The 3 transactions really do get deleted, test with my code to see



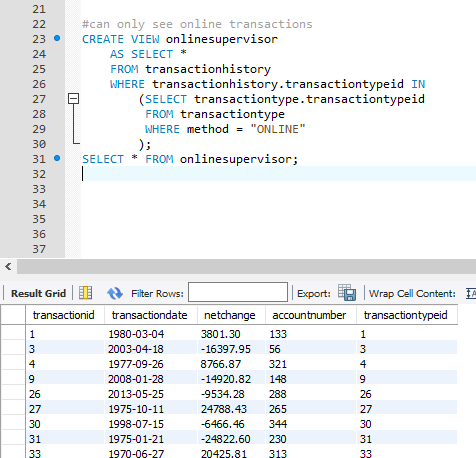
Q7



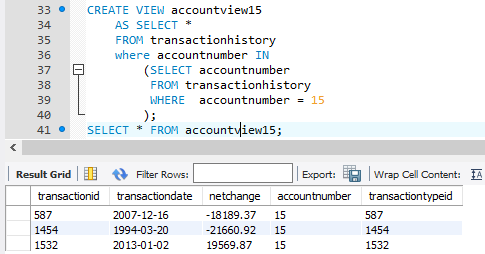
The update works, as shown below. The updated accountnumber 3 changed to branch 3, which caused for the view to update. Also because it only affect the base table



P2 as commented



A view for the person with account number 14. Can see the 3 transaction’s he/she has made for the account



Ex 8

The domain does not work in mysql

After researching, this seems to work in most other sql implementations. This isn’t to important of a difference however, as it just saves some “time” in a well organized product. Domain is effectively used to create user-defined types/groups of user defined types, which does not exist. Sql Server seems to include this functionality