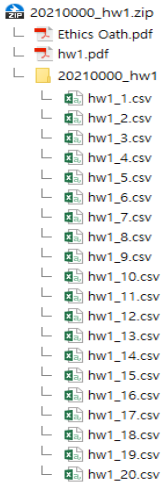
EE477 Database and Big Data Systems, Spring 2021
HW1*

Due date: 04/02/2021 (11:59pm)

Submission instructions: Use KAIST KLMS to submit your homeworks. Your submission should be one gzipped tar file whose name is `YourStudentID_hw1.tar.gz`. For example, if your student ID is 20210000, and it is for homework #1, please name the file as `20210000_hw1.tar.gz`. You can also use these extensions: tar, gz, zip, tar.zip. Do not use other options not mentioned here.

Your zip file should contain three things;



- one PDF file for writeup queries of Section 3 (`hw1.pdf`),
- one folder named `YourStudentID_hw1`(put all output files {hw1_1.csv, hw1_2.csv, ..., hw1_20.csv})
- Ethics Oath pdf file.

Do not include Korean letters in any file name or directory name. **If you violate any format (file name or extension), we will deduct 1 point per mistake.**

Submitting writeup: Prepare answers to the homework questions into a single PDF file. You can use the following template. Please write as succinctly as possible.

Ethics Oath: For every homework submission, please fill out and submit the **PDF** version of this document that pledges your honor that you did not violate any ethics rules required by this course and KAIST. You can either scan a printed version into a PDF file or make the Word document into a PDF file after filling it out. Please sign on the document and submit it along with your other files.

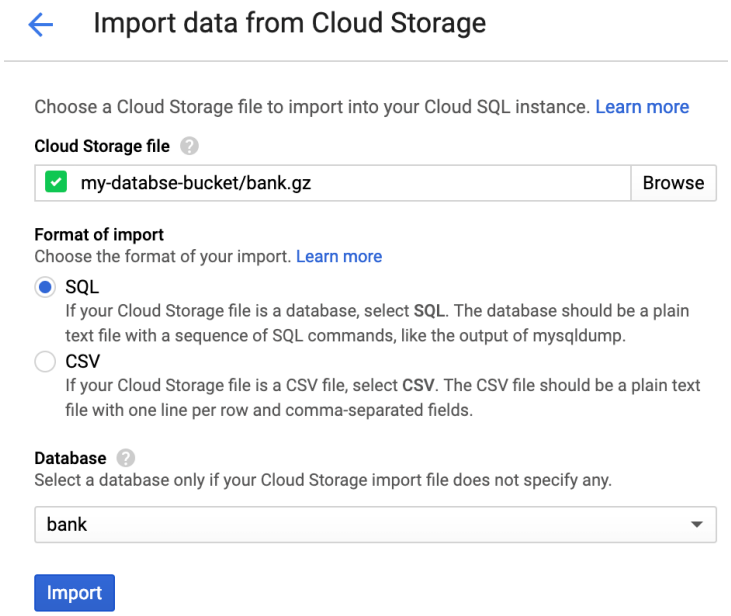
Discussions with other people are permitted and encouraged. However, when the time comes to write your solution, such discussions (except with course staff members) are no longer appropriate: you must write down your own solutions independently. If you received any help, you must specify on the top of your written homework any individuals from whom you received help, and the nature of the help that you received. *Do not, under any circumstances, copy another person's solution.* We check all submissions for plagiarism and take any violations seriously.

*Material adapted from Google Cloud Manual and Simon Fraser University CMPT354.

1 Download Database on your Google Cloud Platform

- Download *bank.gz* from KLMS.
- Upload *bank.gz* on your bucket
- From your bucket, import *bank.gz* to your MySQL instance. The database *bank* you are importing into **must already exist.**

```
CREATE DATABASE bank;
```



Details are described in the link: https://cloud.google.com/sql/docs/mysql/import-export/importing

2 Database Description

The database has five tables. Primary key attributes are underlined and foreign keys are noted in superscript.

- Customer = {customerID, firstName, lastName, income, birthData}
- Account = {accNumber, type, balance, branchNumber$^{\text{FK-Branch}}$}
- Owns = {customerID$^{\text{FK-Customer}}$, accNumber$^{\text{FK-Account}}$}
- Transactions = {transNumber, accNumber$^{\text{FK-Account}}$, amount}
- Employee = {sin[1], firstName, lastName, salary, branchNumber$^{\text{FK-Branch}}$}

[1] Social Insurance Number

- Branch = {<u>branchNumber</u>, branchName, managerSIN$^{\text{FK-Employee}}$, budget}

Notes

- The customerID attribute (Customer) is a unique number that represents a customer, it is not a customer's SIN
- The accNumber attribute (Account) represents the account number
- The balance (Account) attribute represents the total amount in an account
- The type (Account) attribute represents the type an account: chequing, saving, or business
- The Owns relation represents a many-to-many relationship (between Customer and Account)
- The transNumber attribute (Transactions) represents a transaction number, combined with account number it uniquely identify a transaction
- The branchNumber attribute (Customer) uniquely identifies a branch
- The managerSIN attribute (Customer) represents the SIN of the branch manager

3 SQL Advanced Questions (20 points)

As we mentioned before, we strongly recommend you to **STOP** instance when not in use due to a limited number of GCP credit coupon. In this section, you will be learned about more complex SQL queries including:

- Use *Order By* to sort data
- Use *Set Operators* to union/intersect multiple tables
- Use *Aggregate functions, Group By, Having* to aggregate data
- How to write *subqueries*

Query Requirement & Noted Items

- The answer to each question should be a single SQL query
- You must order each query as described in the question, order is always ascending unless specified otherwise
- You must print the attributes as the same order in the problem statement. If the problem asks the "first name, last name, income of customers", then you should print out the result with (first name, last name, income).
- Every column in the result should be named, so if the query asks you to return something like income minus salary make sure that you include an AS statement to name the column (depend on your choice)

- You should not apply the relation between branchNumber and branchName. For example, if the branchNumber of "London" is "1", when the problem requires a salary of employees working at the London branch, you have to use "branchName="London"" information, not "branchNumber="1"".
- The Age follows international age system with Korea Standard Time (integer).
- The Managers are also considered as employees.
- You should print up to **10 LINES** for fast evaluation. So, your queries must end with **LIMIT 10** even if total number of output lines is less than 10

  ```
  SELECT * FROM Customer ORDER BY income LIMIT 10;
  ```

  You should see;

```
+------------+-----------+----------+--------+-------------+
| customerID | firstName | lastName | income | birthData   |
+------------+-----------+----------+--------+-------------+
|      66744 | Rachel    | Edwards  |    383 | 1997-11-02  |
|      87013 | Patrick   | Peterson |    933 | 1978-02-09  |
|      87822 | Dennis    | Long     |   2333 | 1966-10-10  |
|      47953 | Frank     | Martinez |   3254 | 1966-09-06  |
|      13230 | Marie     | Brooks   |   3300 | 1963-06-03  |
|      46058 | Adam      | Rivera   |   3622 | 1972-08-15  |
|      28505 | Joe       | Cook     |   4293 | 1976-11-09  |
|      80321 | Kimberly  | Kelly    |   8431 | 1965-10-06  |
|      41545 | Terry     | Bailey   |   9534 | 1977-03-08  |
|      34069 | Earl      | Lee      |  10630 | 1973-11-09  |
+------------+-----------+----------+--------+-------------+
```

Your queries will not be evaluated on efficiency. So, providing one of the correct answers is good enough. Please refer to several documents including following sites to understand how to use MySQL(5.7 version in GCP).

- http://www.mysqltutorial.org/basic-mysql-tutorial.aspx
- https://dev.mysql.com/doc/

[Task] Using SQL to query a database

(1) First name, last name, income of customers whose income is within [30,000, 80,000], order by income (desc), lastName, firstName.

(2) SIN, branch name, salary and manager's salary - salary (that is, the salary of the employee's manager minus salary of the employee) of all employees in London or New York, order by ascending (manager' salary - salary).

(3) First name, last name, and income of customers whose income is at least twice the income of every customer whose lastName is Butler, order by last name then first name.

(4) Customer ID, income, account numbers and branch numbers of customers with income greater than 70,000 who own an account at both London and Latveria branches, order by customer ID then account number. The result should contain all the account numbers of customers who meet the criteria, even if the account itself is not held at London or Latveria.

(5) Customer ID, types, account numbers and balances of business (type BUS) and savings (type SAV) accounts owned by customers who own at least one business account or at least one savings account, order by customer ID, then type, then account number.

(6) Branch name, account number and balance of accounts with balances greater than $80,000 held at the branch managed by Phillip Edwards, order by account number.

(7) Customer ID of customers who have an account at the New York branch, who do not own an account at the London branch and who do not co-own an account with another customer who owns an account at the London branch, order by customer ID. The result should not contain duplicate customer IDs (write the query satisfying all three conditions).

(8) SIN, first name, last name, and salary of employees who earn more than $60,000, if they are managers show the branch name of their branch in a fifth column (which should be NULL/NONE for most employees), order by branch name (desc) and firstName (asc). You must use an outer join in your solution (which is the easiest way to do it).

(9) Exactly as question (8), except that your query cannot include any join operation.

(10) Customer ID, first name, last name and income of customers who have income greater than 6000 and own accounts in ALL of the branches that Helen Morgan owns accounts in, order by income in decreasing order (The output includes Helen Morgan).

(11) SIN, first name, last name and salary of the lowest paid employee (or employees) of the Berlin branch, order by sin.

(12) Branch name, and the difference of maximum and minimum (salary gap) and average salary of the employees at each branch, order by salary gap.

(13) Count of the number of employees working at the New York branch and Count of the number of different last names of employees working at the New York branch (two numbers in a single row).

(14) Sum of the employee salaries (a single number) at the Berlin branch.

(15) Customer ID, first name and last name of customers who own accounts from three different branches (only three different types of branches), order by Last Name and first Name.

(16) Average income of customers older than 58 and average income of customers younger than 28, the result must have two named columns, with one row, in one result set (hint: look up MySQL time and date functions http://www.mysqltutorial.org/mysql-timestampdiff/).

(17) Customer ID, first name, income, and average account balance of customers who have at least three accounts, and whose last names begin with S and contain an e (e.g. **S**t**e**ve), order by customer ID.

(18) Account number, balance, sum of transaction amounts, and balance - transaction sum (sum of transaction amount) for accounts in the London branch that have at least 14 transactions, order by transaction sum.

(19) Branch name, account type, and average transaction amount of each account type for each branch for branches that have at least 40 accounts of any type, order by branch name, then account type.

(20) Account type, account number, transaction number and amount of transactions of accounts where the average transaction amount is greater than two-and-half times the (overall) average transaction amount of accounts of that type. For example, if the average transaction amount of all business accounts is $2,000 then return transactions from business accounts where the average transaction amount for that account is greater than $5,500. Order by account type, then account number and finally transaction number. Note that all transactions of qualifying accounts should be returned even if they are less than the average amount of the account type.