Homework Assignment 1

ISCI 410 Introduction to Databases
Fall 2016
Due: 10/14/2016 at 11:59PM*

*Note: For assignments of this scale, expect a week to do them. I am giving more time for this assignment because we will not have class on 10/3/2016. However, you should expect an additional assignment to be given on 10/10/2016 that will be due on 10/17/16.

This assignment will use the learning_sql database. The following table shows the primary and foreign keys.

Learning SQL Database Relationships		
Table	**Primary Key**	**Foreign Key**
account	**account_id**	**open_branch_id** *references* **branch.id** **cust_id** *references* **customer.cust_id** **open_emp_id** *references* **employee.emp_id** **product_cd** *references* **product.product_cd**
branch	**branch_id**	
business	**cust_id**	**cust_id** *references* **customer.cust_id**
customer	**cust_id**	
department	**dept_id**	
employee	**emp_id**	**dept_id** *references* **department.dept_id** **assigned_branch_id** *references* **branch.branch_id** **superior_emp_id** *references* **employee.emp_id**
individual		**cust_id** *references* **customer.cust_id**
officer	**officer_id**	**cust_id** *references* **business.cust_id**
product	**product_cd**	**product_type_cd** *references* **product_type.product_type_cd**
transaction	**txn_id**	**execution_branch_id** *references* **branch.branch_id** **account_id** *references* **account.account_id** **teller_emp_id** *references* **employee.emp_id**

Rules:

No collaboration is allowed.

Each of the following requested queries is worth 10 points. Your queries must be logically sound and remain correct even as the database transitions from valid state to another. In other words, if the data in the database were to change, your queries must still provide valid responses. Do not hardcode anything assuming that the data will never change. You can assume that the constraints in the above table will always be satisfied.

For this assignment, the full testing suite will be available on sql-tester.tomchik.net. This means that if you upload your queries and all tests pass you should get a grade of 100%. Note that the *sql-tester* only checks for column names and correct output. I will need to inspect the queries manually to ensure that a subquery was used for query 10. I will also need to make sure that your queries do not contain hardcoded values that would become invalid if the database were to change.

Submission instructions: each query must be in a file named *N*.sql, where *N* is the query number. For example, your answer for query 1 should be in a file named 1.sql. Your final submission should include files 1.sql, 2.sql, …, 10.sql. All .sql files should be submitted in a zip archive with the following naming format: *lastname_firstname_hw1.zip*. For example, my submission would be *tomchik_paul_hw1.zip.*

I recommend that you create a directory on your computer named *lastname_firstname_hw1*. Save all of your queries *(1.sql, 2.sql, …, 10.sql)* in that directory. Then, you can simply create a zip archive of the directory to upload to the sql-tester app and for submission to Blackboard.

Let me add that if you use this naming format for your *sql-tester* uploads, I will be able to find your uploaded scripts on the server. If you have any questions, I will be able to look at your scripts to help you debug them.

For those of you who want a challenge, I encourage you to explore the JOIN syntax

- http://dev.mysql.com/doc/refman/5.7/en/join.html
- https://www.postgresql.org/docs/9.5/static/queries-table-expressions.html

This was not covered in the class lectures, but the concepts were covered in the Relational Algebra lectures and self-teaching is a very valuable skill in programming. The column naming in this database is conducive to that syntax.

Please do not hesitate to send me questions, however please *do not give up on yourself too easily*.

Regretfully, not all of these queries can be performed on the Relax app. I would recommend that you use that app to explore the data, but not much else.

Queries*:

1. Display all the data in the **account** table.
2. Display the **city** and **state** columns from the **branch** table.
3. Display the **cust_id** for all **customer**s whose address is in Waltham, MA.
4. Display all the **postal_code**s in the **customer** table, *removing duplicates*.
5. Display the **account_id**, **cust_id**, and **avail_balance** columns from **account**, in *descending order by avail_balance, then account_id.*
6. Display **cust_id**, **lname**, and **fname** from **individual**, *in ascending order by lname, then fname, then cust_id*.
7. Join the **business** and **officer** tables and display all columns. *Each column name should appear only once in the result.*
8. Join the **account** and **customer** tables and display the **city** and **state** columns in *descending order by avail_balance, then ascending order by state, then city*. In other words, accounts should appear in descending order by avail_balance. When two accounts have the same avail_balance, then they should appear in ascending order by state, then city.
9. Do a three way join on **account**, **product**, and **product_type** and display **account_id**, **cust_id**, and **product_cd** for all accounts where the product_type name is 'Individual and Business Loans'. Each column name should only appear once in the result.
10. Display the **name**s of all branches where at least one business account (cust_type_cd = 'B') was opened. The necessary data is in the **account**, **branch**, **customer** tables. You must *use a subquery*. (The sql-tester will not check this. I will need to check for your using subqueries manually. The tester will only check if the result has the correct column names and data.)

*Note: The required column names are in bold black text. The required table names are in bold blue text.