# DTSC660: Data and Database Management with SQL
# Module 6
# Assignment 4

## Purpose

For this assignment, you have been asked to create a banking database for Eastern Banking Holdings. For this assignment you will rely on the skills obtained from Modules 5 and 6 to successfully create the banking database and query it to retrieve the appropriate responses. Note that many, but not necessarily all, of the tools you learned in Modules 5 and 6 will be applied in this assignment. To complete this assignment, download and import the dataset and then create queries that respond to each prompt. Please make sure that you only use PostgreSQL language conventions.

**Each question is all or nothing. Graders will not attempt to correct or interpret malformed SQL queries.**

You will be responsible for testing your code on the provided data set before submission. Each question will be graded based on whether or not it generates the correct output and addresses all requirements specified in the question. Extraneous columns will not count against you as long as correct results are obtained.

## Submission

You will submit a total of **1** sql file to CodeGrade. Each file must use the postgres standards taught in the course. Use of other flavors of SQL such as T-SQL will result in an automatic 0 for the assignment. Do not submit files as archives (ZIP) files.

- ***File 1***: You must submit a SQL document called <LastName>_Assignment4. This document must include ALL queries requested in the instructions below.

- You will submit the file to the Assignment 4 folder.

- **You will have one attempt to complete this assignment. Additional attempts will not be accepted.** Test carefully. It is expected that if you have questions or difficulties with any portion of this assignment that you utilize the assignment discussion board or email the GAs to gain clarity (dtsc_ga_660@eastern.edu).

---

**PLEASE NOTE:** None of these queries should return more than 15 results (records). Beware of cartesian products.

**There are six questions total in this assignment. Please see the rubric on the last page for grading rules.**

## Question 1:

Consider the bank database schema given below, where the primary keys are underlined. Write the SQL DDL corresponding to this schema (i.e. the CREATE TABLE statements). When you are writing the DDL, make sure you do the following:

a. Create the tables in the order specified in the schema.
b. Use the exact naming convention for table names and attributes.
c. Do not add or remove tables or attributes.
d. For each table, I have provided some assumptions that can be made about the various attributes and relationships between the tables. Your DDL should ensure that these assumptions are true by how you define the table, data types used, and enforcement of triggers and keys, including both primary *and* foreign keys.
e. For this example, there is only one bank, and the individual branches listed in the data are all owned by the one bank.
f. Any attributes identified as the varchar data type should have a length of 40. (varchar(40)).
g. The 'text' field type should **not** be used for primary keys.
h. By the time you have completed you will have at least one of each of the following. Failure to define these will result in lost points:
    i. CHECK constraint
    ii. NOT NULL constraint
    iii. ON DELETE CASCADE clause
    iv. ON UPDATE CASCADE clause
    v. DEFAULT value statement.

Database Schema:

*branch ( branch_name, branch_city, assets )*
*customer ( cust_ID, customer_name, customer_street, customer_city )*
*loan ( loan_number, branch_name, amount )*
*borrower ( cust_ID, loan_number )*
*account ( account_number, branch_name, balance )*
*depositor ( cust_ID, account_number )*

Important notes for each table:

- **Branch**: All branches must have assets that are monetary (they have a dollar value). There are four and only four cities with branches: Brooklyn, Bronx, Manhattan, and Yonkers.

- **Customer**: Customers must have a name. Note from our dataset that cust_IDs are NOT sequential, and you can reasonably expect that new customers will also not be issued sequential IDs.
- **Loan**: Loan numbers can contain both letters and numbers. All loans must have a monetary amount. The default amount for loans is zero dollars and zero cents. If a branch closes or changes its name, these activities should be reflected in the loan table.
- **Borrower**: A borrower is a type of customer, so if the cust_ID is deleted or changed, the borrower table should reflect these actions. The same is true of the loan_number.
- **Account**: Accounts should always have a monetary balance. If a branch closes or changes its name, these activities should be reflected in the account table. Note from our dataset that account numbers are NOT sequential, and you can reasonably expect that new customers will also not be issued sequential IDs.
- **Depositor**: A depositor is a type of customer, so if the cust_ID is deleted or changed, the depositor table should reflect these actions. The same is true of the account_number.

**For the next section, you will need to use the data file attached to this assignment to complete your queries. If your queries do not run, go back and fix your Banking DDL. Make sure you use the correct table names and attributes. If your code does not execute with the provided data, you will not receive credit for the remaining parts of the assignment.**

### Question 2:

Write a query to find all customers who are depositors and return their Customer ID, Branch Name, Account Number, and Balance.

### Question 3:

Write a query that returns all customers who are both depositors *and* borrowers. Include in your query the customer ID, account number, and loan_number.
*Tip: In order for a customer to show in this query, they should have at least one deposit account and one loan. Some customers may have more than one of either, which means you may end up with multiple rows per customer.*

### Question 4:

Write a query that finds the account number of all customers who have a deposit account in the same city in which they live. Include the customer's city, branch city, branch name, and account number.

*Tip: The "home city" of a deposit account is defined as the city of the branch the account is assigned to. In order to find the correct customers for this query, you will need to compare the customer's city to the branch city for the customer's account(s).*

### Question 5:

Write a query to generate a unique list of customer IDs that are both depositors and borrowers WITHOUT USING JOINS.
*Tip: In order for a customer to show in this query, they should have at least one deposit account and one loan. As this is a unique list of customer IDs, each customer should appear only once.*

---

**THIS FINAL QUESTION USES THE UNIVERSITY SCHEMA, NOT THE BANKING DDL.**

### Question 6:

The following question references the university schema instead of the banking schema referenced in previous questions. **You must use the schema as specified in module 5.1.** Please write an appropriate query for the question below:

Write a SQL query using the university schema to find the ID of each student who has never taken a course at the university. Do this using no subqueries and no set operations (use an outer join).
*Tip: Do **not** simply use the tot_cred field on the student table. A student may have a tot_cred above 0 if they have transferred to the school, but have not taken any courses yet. On the other hand, they may have taken and failed a course, or may be actively taking a course that hasn't yet been finished, and they theoretically would have 0 tot_cred but would still have taken courses.*
*Depending on whether you use the "large" or "small" university dataset, you may need to add records to your database to test your query.*

**********GRADING RUBRIC ON NEXT PAGE**********

This assignment will be graded on the following rubric. Remember that questions are ALL OR NOTHING. Incorrect syntax, extraneous results, or incorrectly addressing all question requirements will result in loss of points for that question. Graders will NOT attempt to correct malformed sql code. :

| **Question Number** | **Points** |
|---|---|
| 1 - 6 Tables correctly named with correct attribute names | 12 (2 points Per Table) |
| 1 - Primary Keys - Correctly added constraints with 8 Primary Key attributes | 8 (1 Point Per Primary Key attribute) |
| 1 - Other Constraints - Correctly added 5 different constraint types outlined above | 10 (2 Points Per Constraint Type)*** |
| 2 | 15 |
| 3 | 15 |
| 4 | 15 |
| 5 | 15 |
| 6 | 10 |
| **Total** | 100 |

***Please see part h of Question 1 and make sure you have at least one of each constraint type