ECE356 Lab 1: Query Optimization

In this lab, you will:

- Load your local copy of the database
- Create SQL queries given natural language descriptions and analyze their performance
- Optimize those queries to meet a performance specification

Background

In the Fall of 2024, Mike the lab instructor set up a submission server for ECE250. Students submitted their `tar.gz` files, which contained their C++ code, and the server extracted, compiled, tested, and analyzed their submission. For more information on this, see the database documentation on Learn. The initial database was designed quickly, with little thought to scalability and performance. Since Mike only ran a few queries per day and didn't mind waiting a few seconds for the results, he decided that the important thing was getting it working rather than optimizing it. (Note: this part is actually true. The database you're using really did come from ECE250 submissions and I really didn't bother to optimize it).

Mike is now thinking of releasing the database in read-only form to researchers who are interested in mining it for insights into how students approach programming projects. Much to his surprise, he now has a mountain of technical debt to contend with. Mike reached out to some of his colleagues and asked them to test the database against queries that they would likely be interested in running, then let him know how well his database performed. Mike's colleagues are not experts in database administration, so their queries were potentially inefficient.

Mike now has a problem. Some queries ran very quickly, but others take multiple seconds to complete. This just won't scale! Luckily, many of his colleagues indicated that they'd be happy to use a pre-made query to accomplish common tasks. Mike has asked that you take a look at what his colleagues have said, and help to achieve the goals. (Note: this part is sort of true. I've had a lot of people interested in mining the database, but I artificially created some inefficient queries so that we have an interesting assignment to do. Such is the trade off between teaching and research I suppose).

Understanding the Common Queries

Below are the queries that Mike's colleagues mentioned they'd like to be able to run efficiently, along with data on how long they took to run.

Query Description	**Maximum Runtime Recorded by Users**	**Notes**
Obtain all comments for a given project	1.6 seconds	projects p3 through p5 tend to have many more comments than others. Speedup may not be possible for all projects.
Obtain all comments for an early project (either p0 or p1) for a given term	1.34 seconds	The query should return all comments for a single project, either p0 or p1, for a given term
Average cognitive complexity per function per project, sorted by project name then function name	0.81 seconds	At least a 50% speedup is required. This is a *very* common query to run
Obtain a list of unique function names that appear in one project but not another	0.19 seconds	This query might be used, for instance, to see which functions were added to p5 compared to p4
Find all submissions that have comments extracted but no test scores recorded	11.5 seconds	At least a 10x speedup is required. This speedup must be achievable on the **unmodified** database for a user with read-only privileges

What You Need to Do

1. Start by examining each query and trying to replicate its output. Each query has an associated output file on Learn so you can compare to make sure you've got it right.
2. Try at least 2 different techniques per query to speed them up. For queries without a required speedup, you should explore whether speedup is even possible for the entire range of input parameters. For queries with a required speedup, you *must* achieve that speedup with at least one of your techniques. I promise it is possible. If a particular technique was tried but failed, be sure to record that in the report template on Learn - failures or unexpected results are important!
3. For each query, you likely either optimized the database in some way, optimized the query, or both. You need to submit three SQL files: `setup.sql` should include **all** database optimizations but **no** queries. `queries_readwrite.sql` should include your queries for the first four queries described above. `queries_readonly.sql` should include your query for the final query described above. Two sample files have been provided on Learn for you to understand what we are looking for. Some queries above depend on different inputs (for instance, they may depend on the project we are interested in getting data about). The following table describes what must be in your query files.

Query Description	**Notes**	**Query File**
Obtain all comments for a given project	One query for p0, one query for p5	`queries_readwrite.sql`
Obtain all comments for an early project (either p0 or p1) for a given term	One query for p0 f24, one query for p1 w25	`queries_readwrite.sql`
Average cognitive complexity per function per project, sorted by project name then function name	One query only that produces the required output	`queries_readwrite.sql`

Query Description	Notes	Query File
Obtain a list of unique function names that appear in one project but not another	One query for function names in p5 not in p3	`queries_readwrite.sql`
Find all submissions that have comments extracted but no test scores recorded	One query only that produces the required output	`queries_readonly.sql`

What You Need to Submit

Submit a flat zipfile (that is, no subfolders) named `<your_uw_username>_lab1.zip` to the dropbox on Learn. For instance, I would submit `mstachow_lab1.zip`. This should contain exactly four files, named exactly as follows:

1. `report.txt`: a filled-in template document
2. `setup.sql`: described above
3. `queries_readwrite.sql`: described above
4. `queries_readonly.sql` : described above

WARNING: many students with macs are surprised to find that mac puts everything into subfolders whether you want it to or not. It is imperative that you do not provide a zipfile with anything other than the above four files, since they will be tested via an automated script that will not look in subfolders.

How This Will Be Tested

First, Your report will be reviewed to ensure that you have indeed tested at least 2 techniques and reported any speedup results. Then the queries will be tested in two phases. The machine we will use is the same one used to obtain the original runtime results. No other users will be logged in. No unnecessary processes will be running.

1. A fresh copy of the database will be loaded by a user with DB admin privileges. We will run `setup.sql` first, then `queries_readwrite.sql`. The output of your queries, as well as the time they take, will be compared to our solution. The outputs must match (except possibly for sorting) and the time must meet any given specifications.
2. A fresh copy of the database will then be loaded by a user without DB admin privileges. This user has read-only access. `queries_readonly.sql`

will be run. The output of your queries, as well as the time they take, will be compared to our solution. The outputs must match (except possibly for sorting) and the time must meet the 10x speedup requirement (that is, your query's runtime must be less than or equal to 1.15 seconds). Since the database must be the unmodified version, we will not be running your `setup.sql` script.