

Query Analyzer

Project Summary

Query Analyzer is an interactive web application to practice SQL. The goal of this application is to aid students to learn SQL by understanding the step by step execution of a SQL query. This tool visualizes the results of the intermediate steps during query execution. The application also provides a dashboard for the Instructors to see the various aspects of the students' performance on a set of predefined SQL practise questions.

Project Description

For SQL beginners, it is not easy to imagine and derive the procedure mentally when facing a query goal. Visualizing each step can help the learners understand the abstract problems more clearly.

Practice reinforces concepts. This is true especially while learning to write SQL queries. Instead of just knowing the function of clauses, students need to combine them flexibly to solve a variety of problems. Seeing the intermediate output enhances the understanding of logical flow of data during the execution of SQL queries.

Unlike Prairielearn, where instructors only see a general statistical result, our application provides more significant metrics to evaluate students' performance such as average number of clauses used for a successful submission, total number of failed attempts before successful submission etc.

Usefulness

The application allows students to understand the sequence of steps involved in the execution of SQL queries and the outputs of each step. As a result students get a better understanding of SQL internals and query performance. The tool provides more insights than merely the correctness of the solution.

In CS 411 we have a query testing platform on Prairielearn which executes the given SQL query on multiple instances of the database (populated dynamically with some constraints) and computes the difference between the expected query output and the output from the student's query, thereby providing instant difference based feedback. It also specifies any errors mentioned by the SQL compiler. Questions are scored on the number of successful test cases. There is a dashboard to see the percentage of successful test cases per question and also percentage of questions completed.

Tools like [QueryVis](#), minimize the amount of time taken for a person to understand existing database queries and write new ones. The tool interprets the intent behind existing queries and

groups similar ones. This provides users the ability to browse query design patterns across schemas before writing a new one. Many DB Visualizers such as [DBVis](#) provide features like autocomplete, organization of queries, query generation based on drag and drop of tables, formatting, creating reusable templates of queries where column names can alone be changed. This tool provides a query performance analyzer which displays the query plan graph and the CPU, I/O costs per node in the plan. It has a feature for tuning granularity of detail(can be set to join level or indices used for join level) as shown in Figure 1 [\[DB Visualizer\]](#)

This is a novel feature for advanced users, however it could be overwhelming for beginners to follow such a graph. It is easier to understand if step numbers are shown clearly, with the inputs and outputs determined at that step along with the corresponding clause highlighted and the relative execution time of the step. Such features make our application unique and more suitable for learners than database experts.

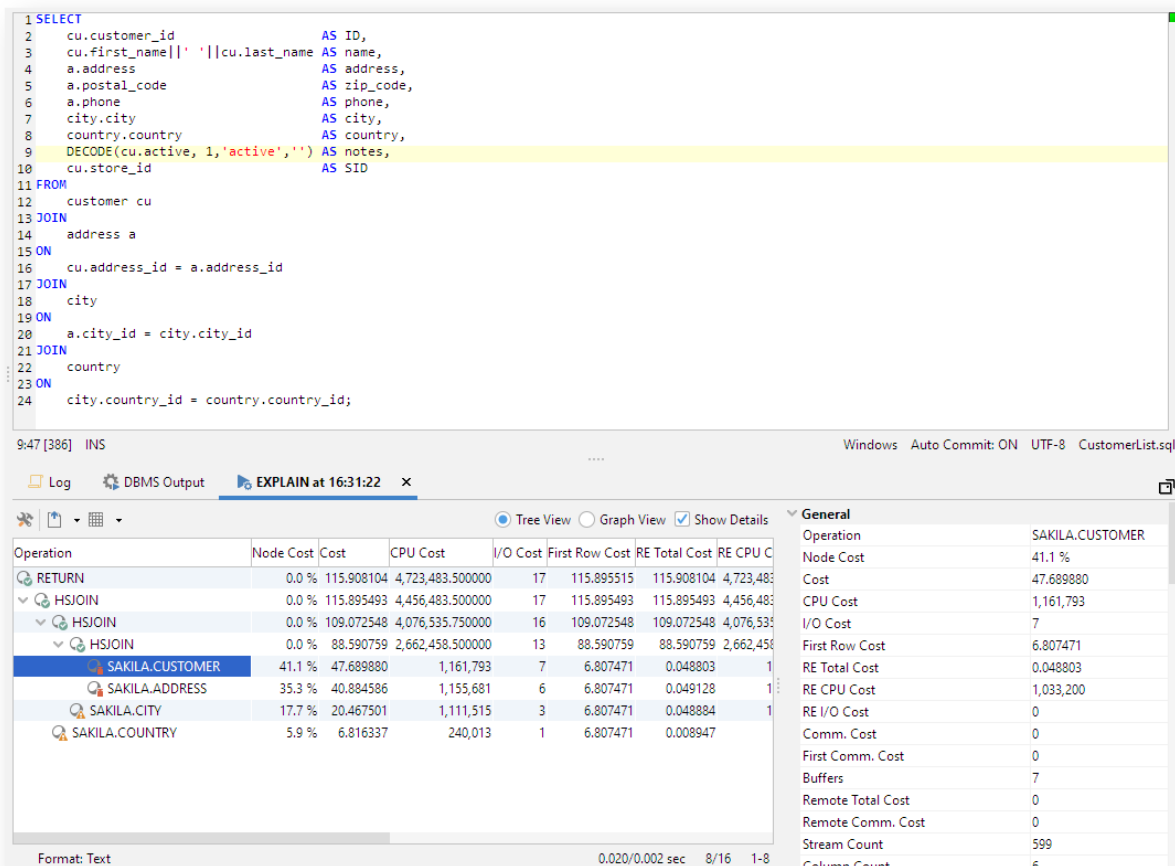


Figure 1: DB Visualizer Tool

Realness

Query Analyzer will deal with a variety of data. We list them as follows:

1. As the primary objective of this application is to help the user visualize intermediate results of a complex SQL query, we receive this data as an input from the user. The

query input is then broken down into a series of subqueries which are then executed on a well-known and widely used database. For the scope of this project, we will make use of the “Students Enroll in Courses” database.

2. Secondly, we allow the user to study and gain competence in SQL by solving practice questions. As the developers of the application, we will decide the practice questions for the students and they will be stored along with corresponding solutions in a dedicated table. Initially, we will make use of the SQL practice and assignment questions on Prairielearn so that students can better understand their own solutions and perhaps optimize their queries. We reserve the functionality of instructors to upload their own questions for the future.
3. Furthermore, the application will keep track of user metadata such as query performance, query history, number of submissions, number of intermediate steps per submission etc for every user. This data will be collected dynamically when the users engage with the application. This metadata will be used to provide insights to the instructor by uncovering interesting observations such as question difficulty, success rate, time-to-solve, etc.

Functionality

The major focus of the application is to analyze the queries. However, we also plan to build minimal functionality around it so that the application makes sense as a standalone real time application.

Main functionality

Query analyzer will support

- 6 clauses FROM, WHERE, HAVING, GROUP BY, SELECT, ORDER BY
- uncorrelated nested queries in FROM, WHERE and HAVING clauses.

Role specific functionality

We are providing two roles in the application:

Role 1: Instructor

View a table of questions and statistics such as % of successful submissions, % of students who answered right, total # of attempts, average number of clauses executed across successful submissions. (The questions, database data and correct queries are stored statically in the DB.) Refer [Figure 5](#).

- Sorting of the table based on columns.
- Search for a question using search bar

Role 2: Student

- View a list of questions along with the status as completed / not completed. Refer [Figure 2](#)

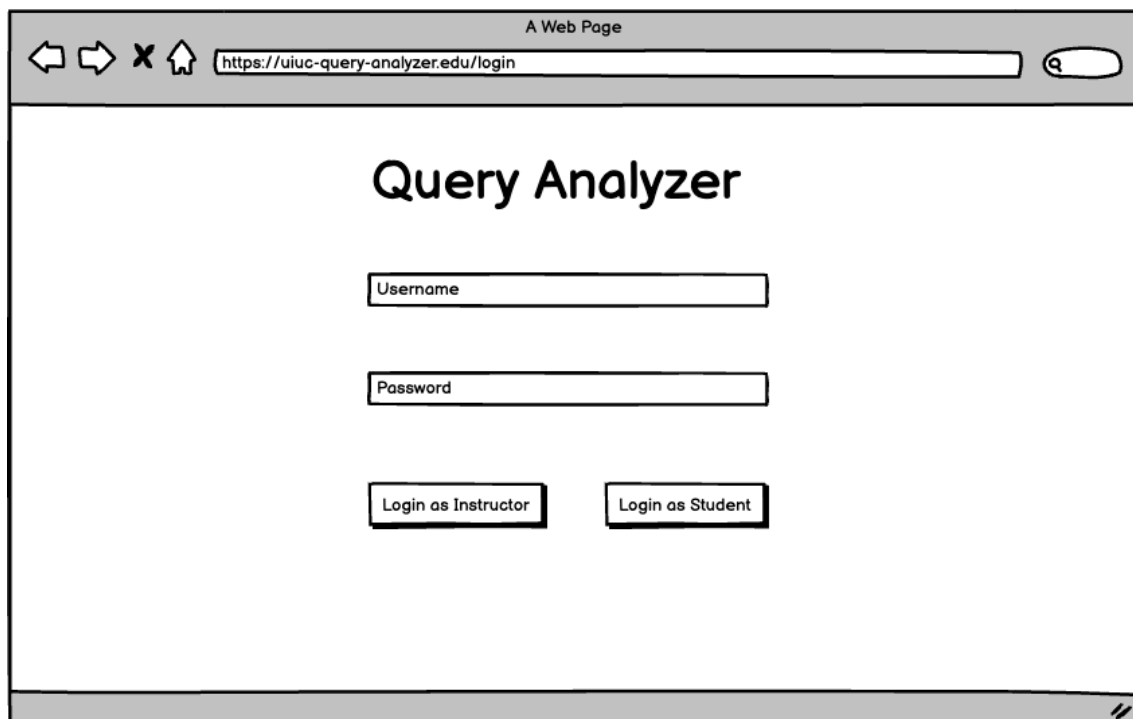
Common functionalities

- **Login Page:** Authentication using username and password. Encrypted passwords will be stored in the database along with the username
- **Query Analysis Page:**
 - Query analysis results which specifies the order of steps. As the student goes from one step to the next the corresponding clause is highlighted in the pretty printed query. We also plan to indicate a fraction of total execution time for each clause to help students understand the bottlenecks.
 - Query analyzer specifies the input and output tables per intermediate step. Pagination per table with page sizes will be added to improve user experience.
 - Compiler output result is shown that specifies errors produced by the SQL compiler.
 - Grading output: Executes the student query on the database tables and compares the result against that produced by the reference query, which will be stored in the database and mention correct/incorrect. In case it's incorrect, display the expected result.

UI Mockup

We envision our web application to have the following pages.

- **Login Page**



The image shows a web browser window titled "A Web Page" with the address bar displaying "https://uiuc-query-analyzer.edu/login". The main content area features the heading "Query Analyzer" in a large, bold font. Below the heading are two input fields: "Username" and "Password". At the bottom of the form are two buttons: "Login as Instructor" and "Login as Student". The browser window includes standard navigation icons (back, forward, stop, home) and a search icon in the address bar.

Figure 2: Login Page

- Page with a list of practise questions

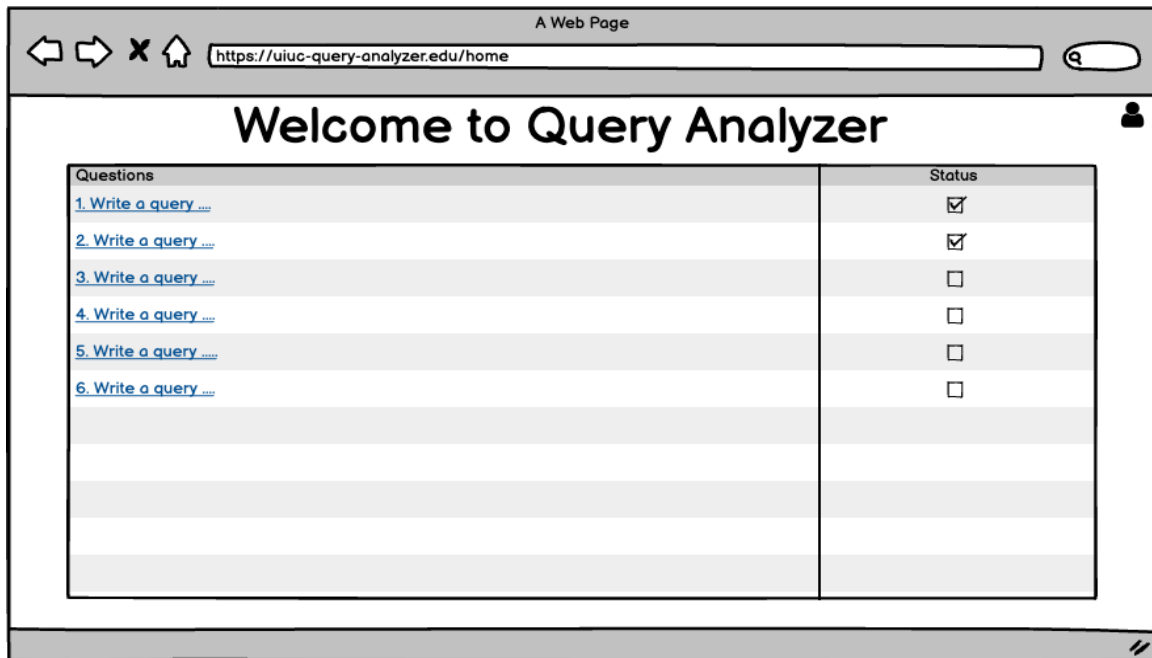


Figure 3: List of questions with status

- Page to write query, execute and view the query analysis results

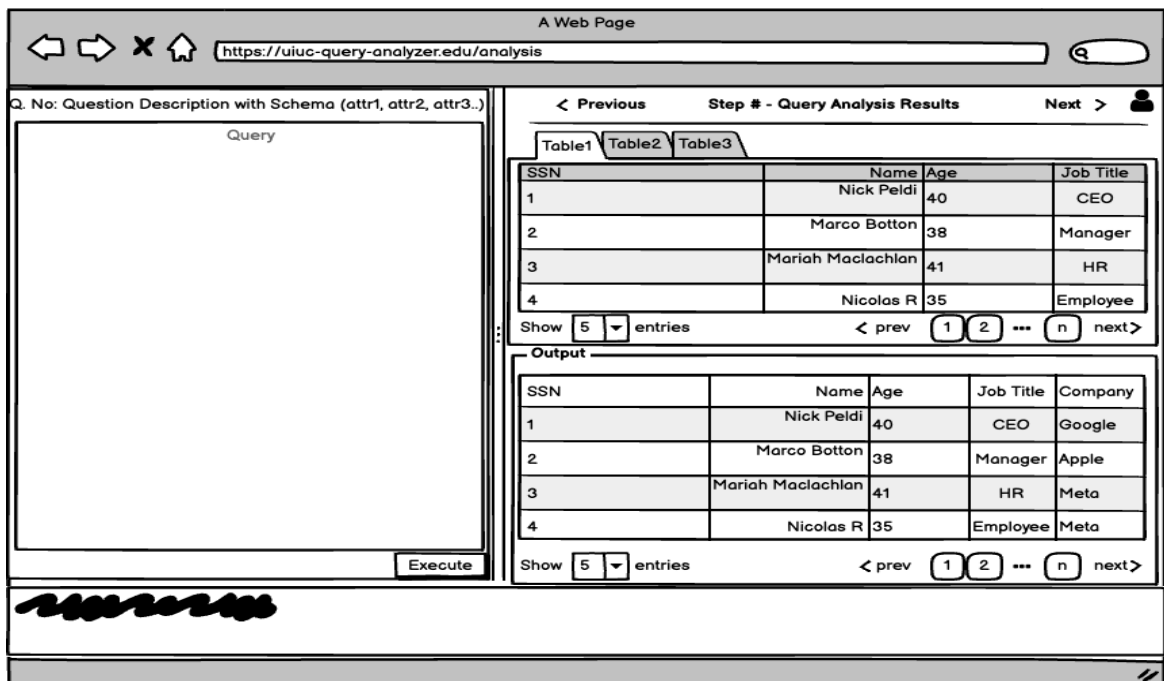


Figure 4: Query Analyzer View

- Dashboard with performance metrics

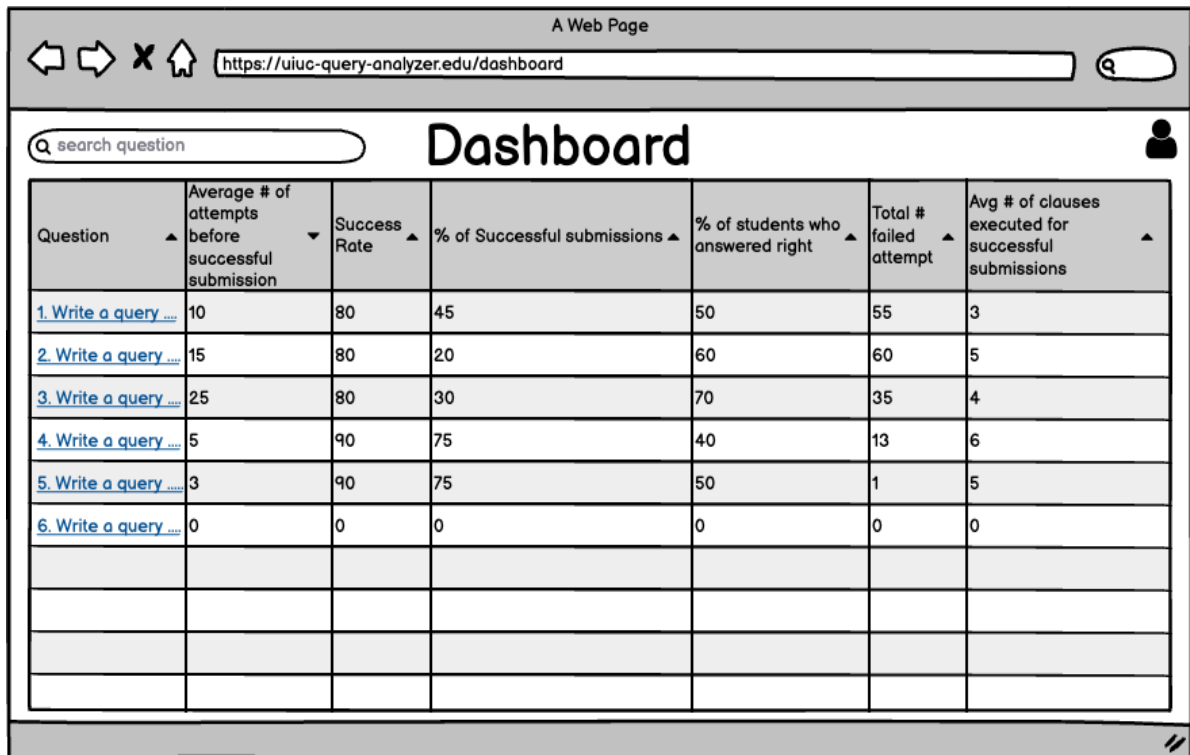


Figure 5: Dashboard with metrics about Students' Performance

Project Work Distribution

Everyone in the team will distribute the work across all the tasks. However the team member mentioned in the below table is responsible for the completion of the Task.

Task	Responsibility
Initial Environment setup	Yeshwanth
Establish GCP to server connection	Rajiv
Establish client server connection	Vidya
Schema design (meta data)	Siqi
Query parsing to identify the different clauses	Rajiv
Intermediate query generation	Yeshwanth
Chaining the Intermediate results	Siqi
Business logic (server side)	Vidya

UI for Login screen	Yeshwanth
UI for Query Visualiser	Siqi
UI for Students Question screen	Vidya
UI for Instructor view (Dashboard)	Rajiv