

**CS 306, Database Systems**

**Project Phase 2**

**Application of Relational Algebra, SQL, and Constraints in Database Management**

Can Çiftçioğlu, 26532

**Submission Date:** 19.11.2023

**Contents**

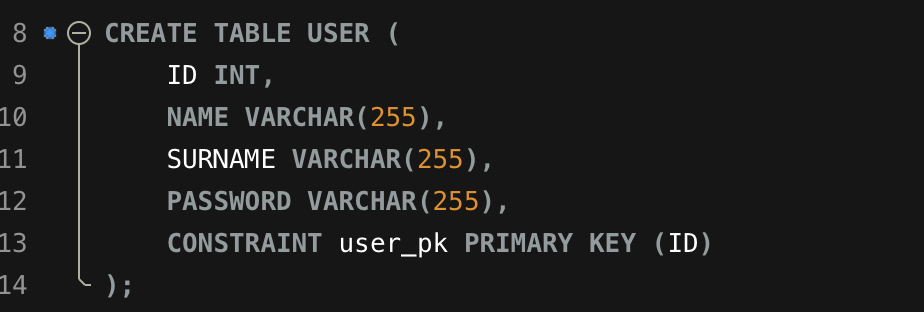
* Introduction
* Table Selection and Relationship
* Data Insertion
* Data Display
* Relational Algebra Query and Its SQL Equivalent
* SQL Query Execution
* Group By Query With Aggregate Function
* Constraint Addition and Testing

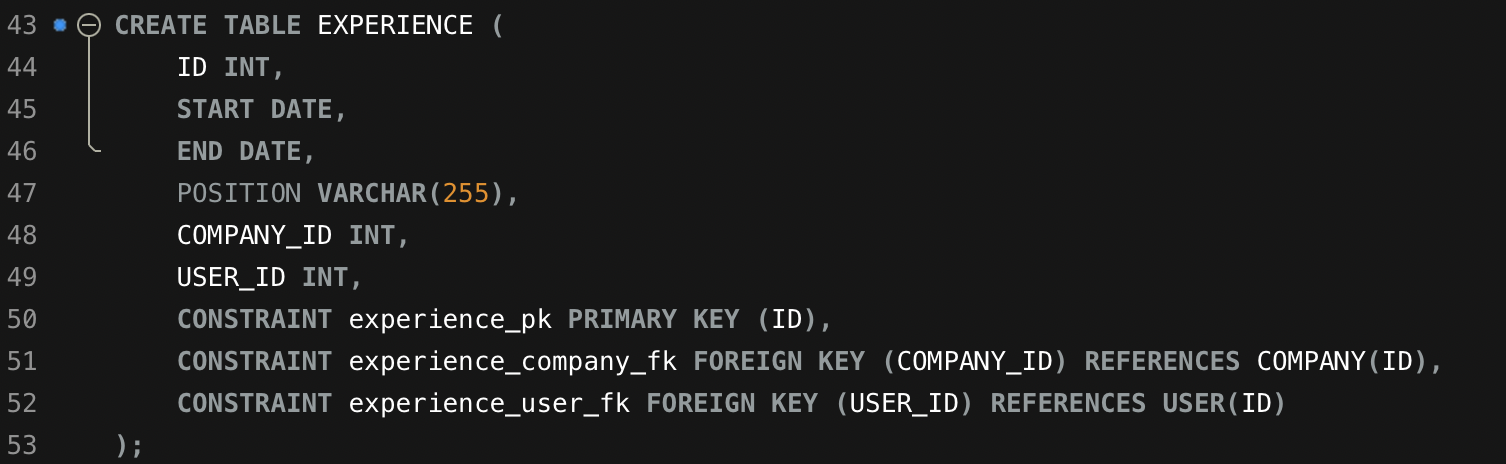
**Introduction**

This phase of the project expands on Phase I's objective, which was to develop a professional network management system similar to LinkedIn. Designed to manage user interactions, work experiences, and professional relationships within the business community, the system is a database-driven tool. Using SQL, we will add relevant user data to our database, including identifiers, posts, comments, employment histories, educational backgrounds, and more. The functionality of our application includes using this data to make recommendations based on user profiles and their data-driven interactions, in addition to efficiently storing it.

**Table Selection and Relationship**

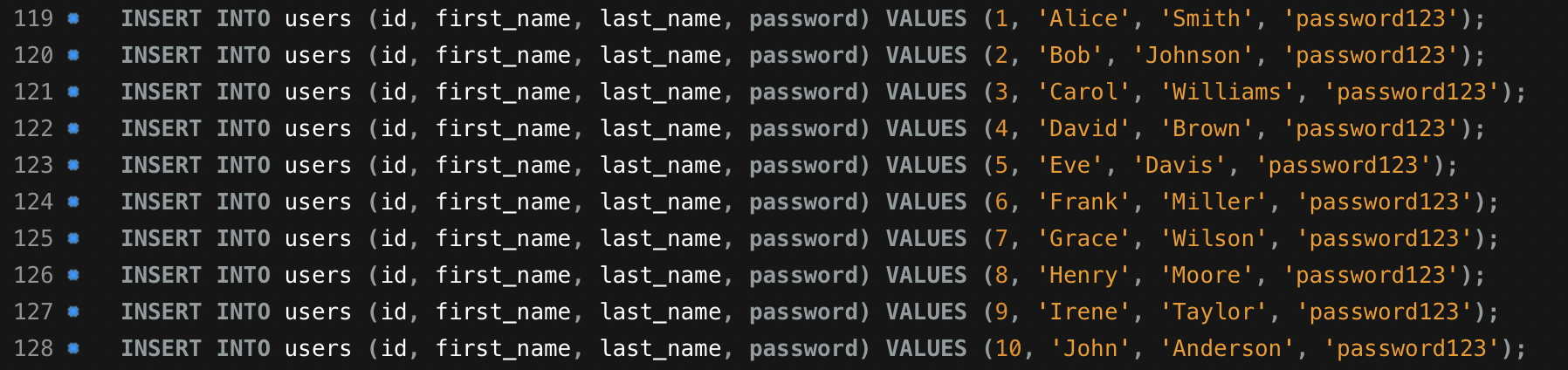
For this phase, I have chosen the Users and Experience tables from my database project. These tables illustrate a fundamental relationship where a user's profile is linked to their professional experiences, with the Users table's primary key, id, serving as a foreign key in the Experience table.

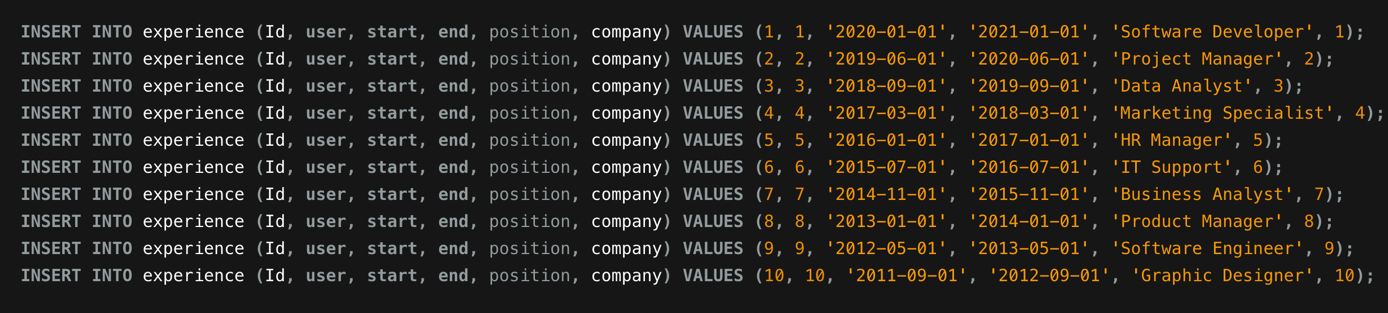




**Data Insertion**

Ten rows were inserted into each of the Users and Experience tables. The insertion statements were crafted to adhere to the data integrity rules defined in the schema, ensuring that each experience is associated with a valid user in the system.





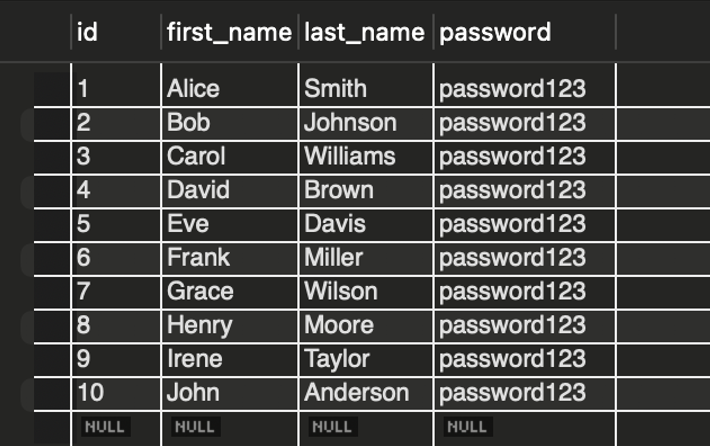
**Data Display**

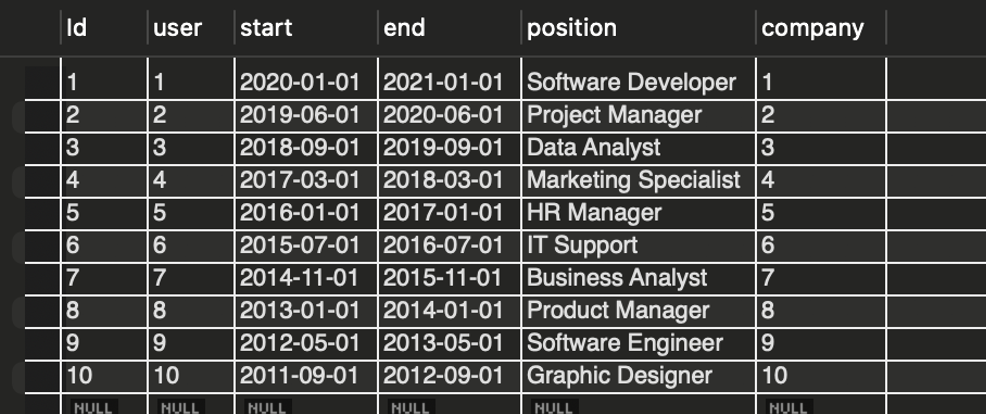
I executed





statements to display the data from both tables, ensuring successful data insertion and schema compliance.



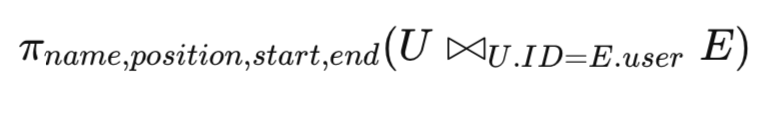


**Relational Algebra Query and Its SQL Equivalent**

"Retrieve the names of users along with their position and the start and end dates of their experience."

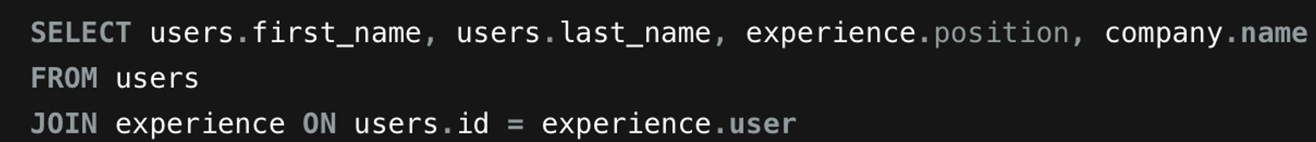
This query requires joining the user and experience tables to match each user with their respective experience details. To represent this query in relational algebra, we'll use the **JOIN** operation. Assume U is the user table, and E is the experience table. The attributes we're interested in are name from U and position, start, and end from E.

The relational algebra expression would be:



**SQL Query Execution**

The relational algebra was translated into the SQL query:



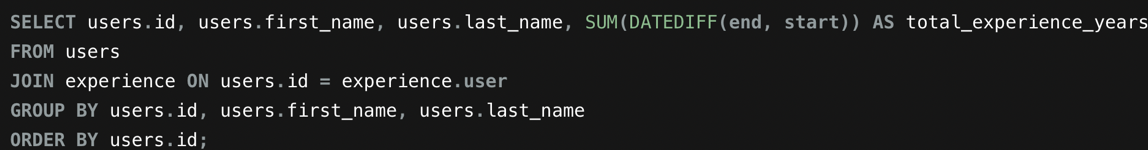
Execution of this query in the MySQL environment fetched the required joint data set, demonstrating the association between users and their experiments.



**Group By Query With Aggregate Function**

"Determine the average duration of experience for each user, calculated as the average number of days between the start and end dates of their experiences."

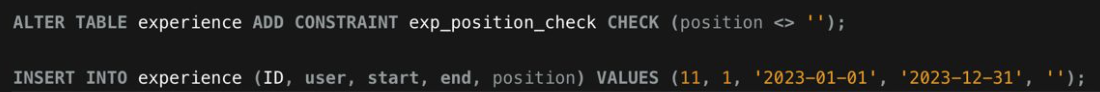
This query requires joining the user and experience tables and then using the **GROUP BY** operation along with a statistical operator (**AVG** in this case) to calculate the average duration of experience for each user.

****

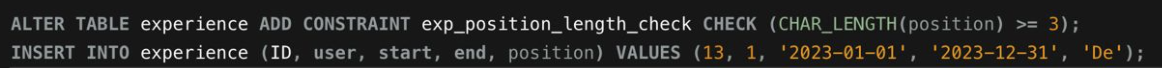
****

**Constraint Addition and Testing**

An **ALTER TABLE** command was used to add a **CHECK** constraint to the Experience table, ensuring that position is not allowed to be empty and position must be at least 3 characters long. Attempts to insert non-compliant data resulted in errors, which will be documented in the report to demonstrate constraint enforcement.









Also, a new **CHECK** constraint was added to the Users table to ensure that user-entered first and last name input cannot include empty characters.

