EC5070: Database Systems Lab 04

BANDARA H.G.T.D.

2022/E/048

06.06.2025

**Attach the below screenshots for following scenarios.**

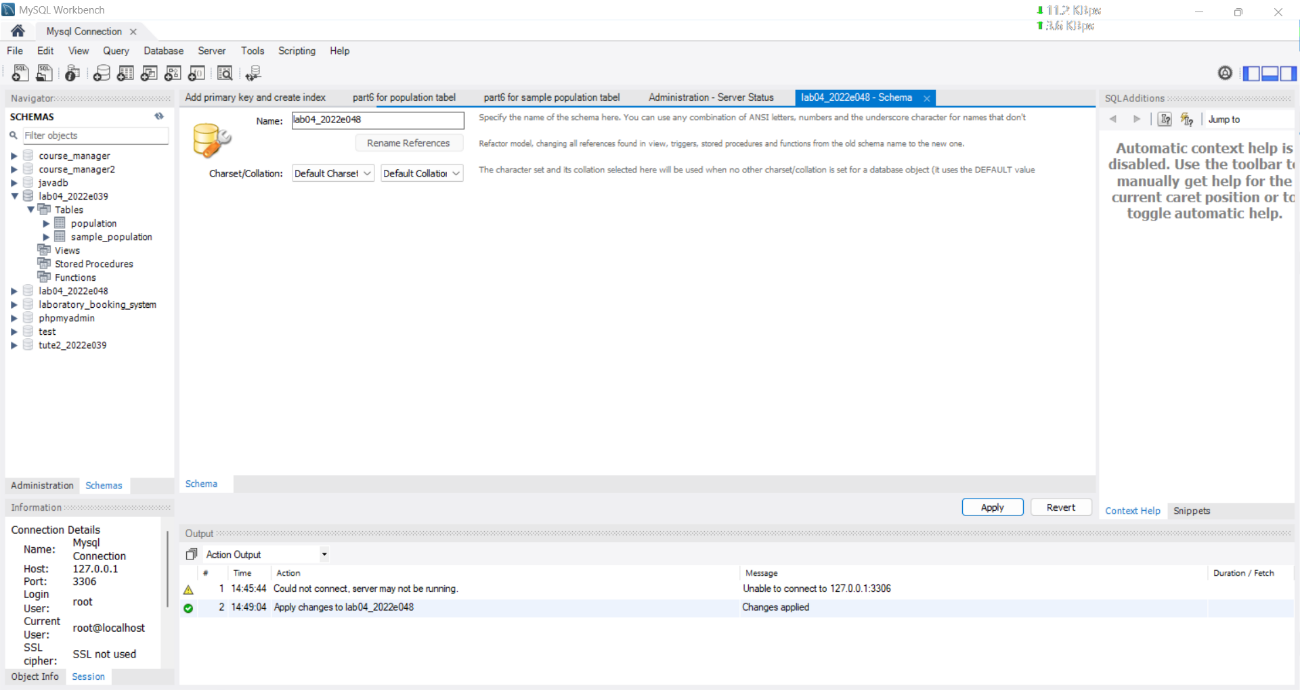
Q1. Database with your registration number and lab number.

FIGURE 01:CREATE DATABASE

Q2. Tables with the name sample population and population.

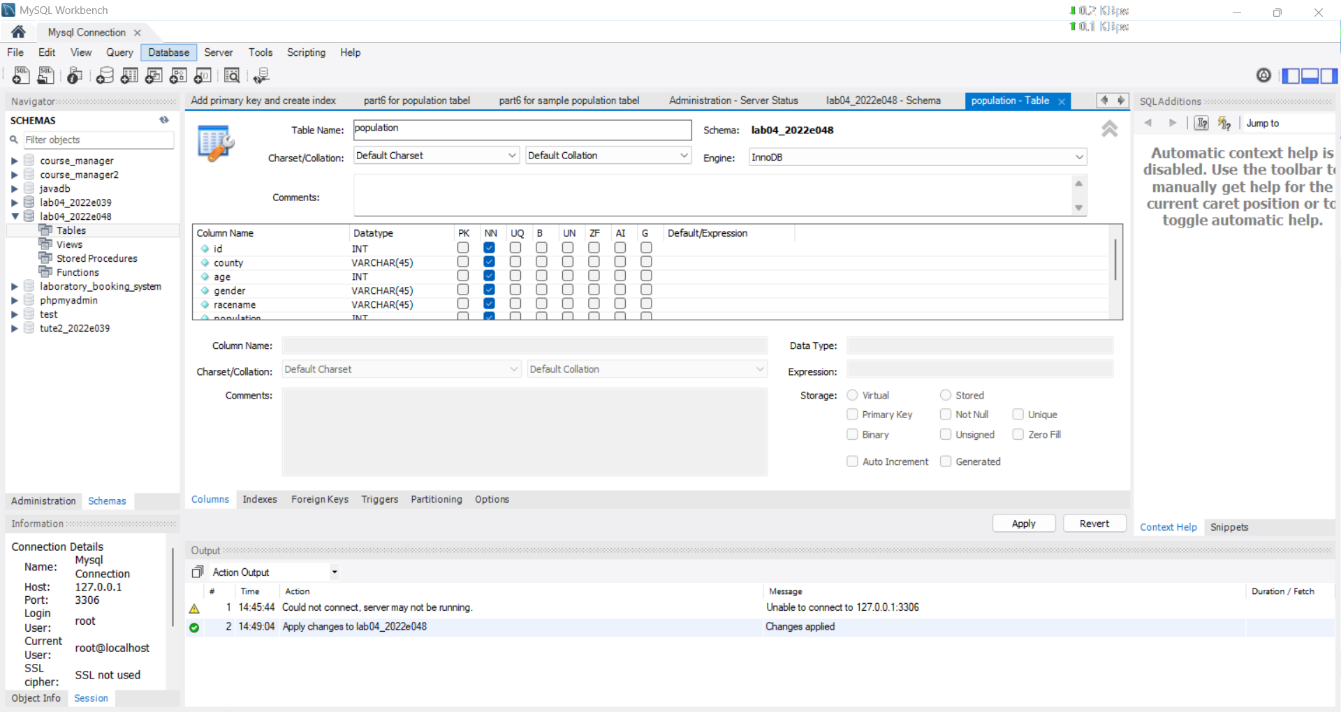
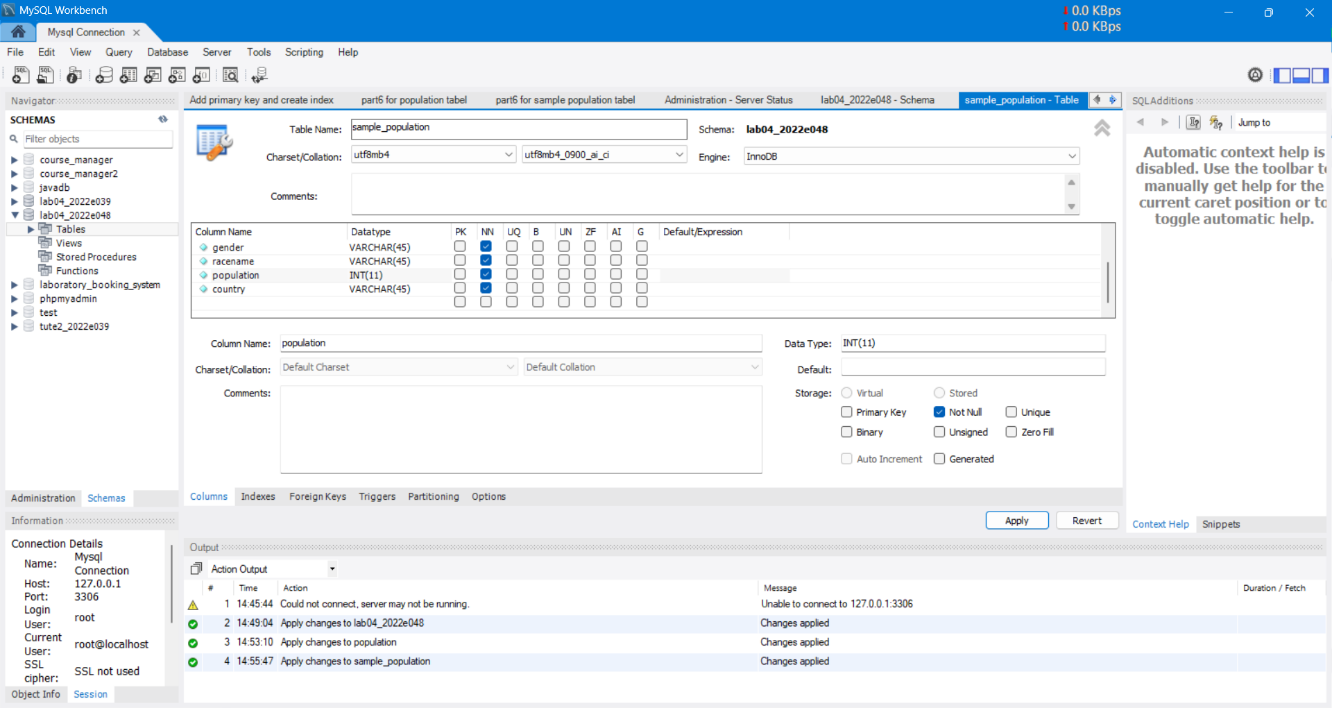


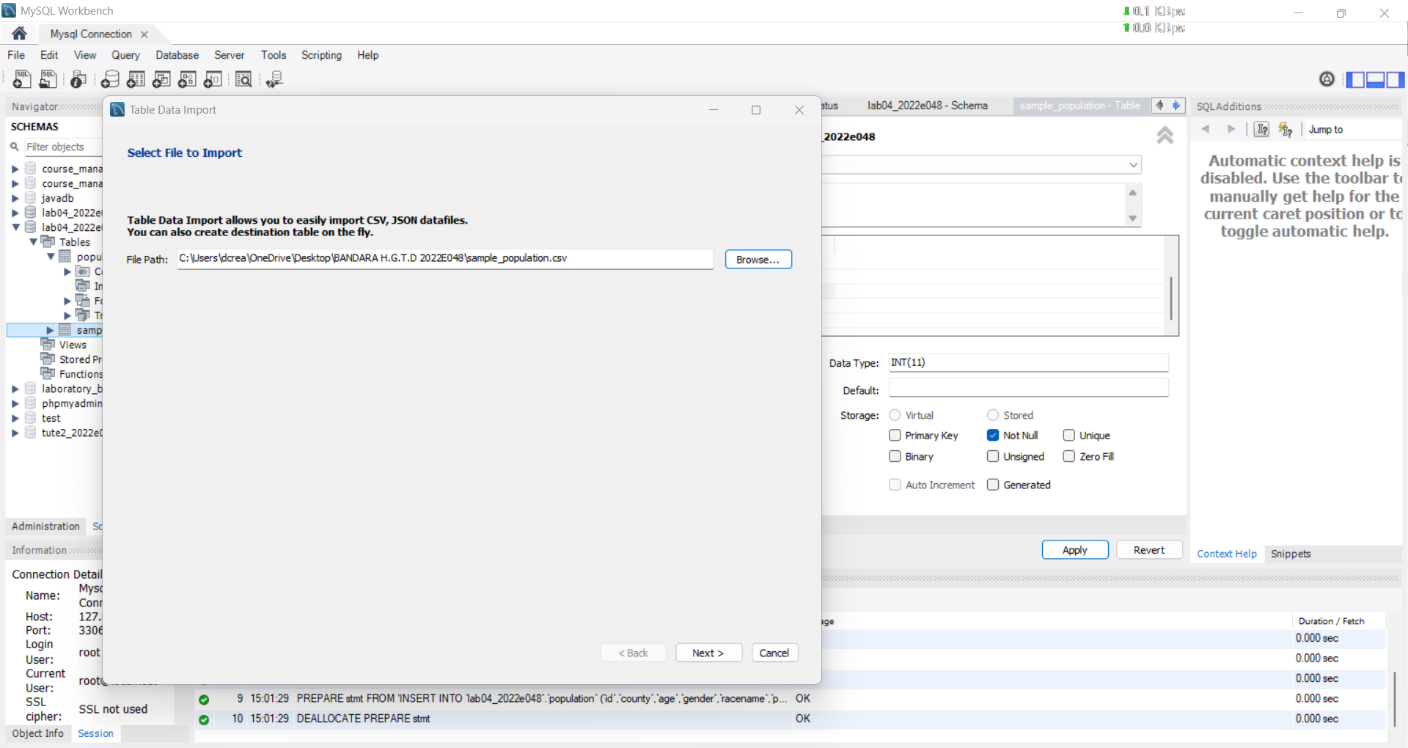
FIGURE 02:CREATE TABLE FOR POPULATION

FIGURE 03:CREATE TABLE FOR SAMPLE POPULATION

A screenshot of a computer

AI-generated content may be incorrect.Q3. Importing .csv files using import data wizard.

FIGURE 04: IMPORT population.csv FILE USING DATA WIZARD

FIGURE 04: IMPORT sample\_population.csv FILE USING DATA WIZARD

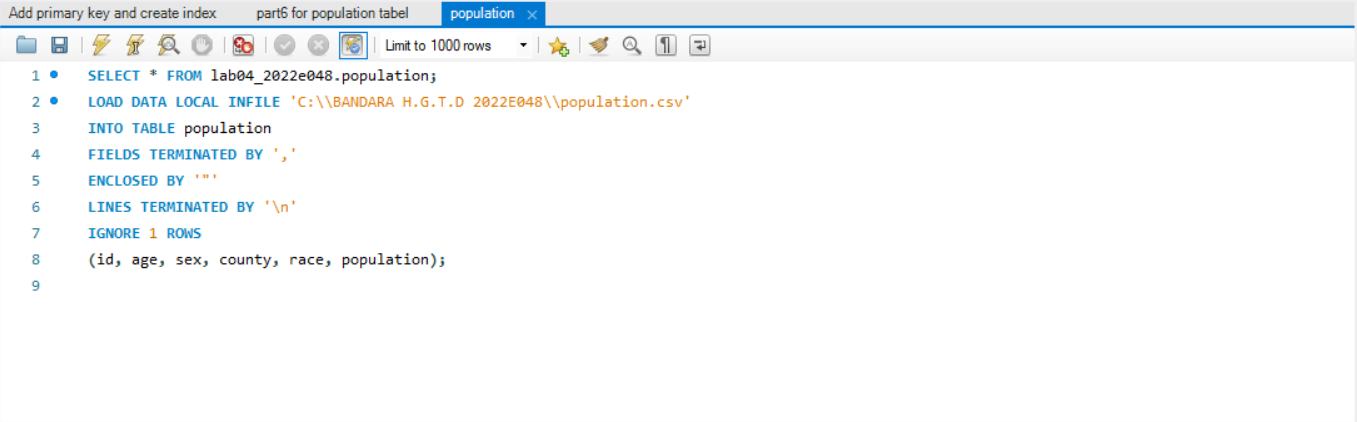
Q4. Importing .csv files using queries

FIGURE 04: IMPORT population.csv FILE USING QUERIES

Q5. Explanation of your observation for question 3,4.

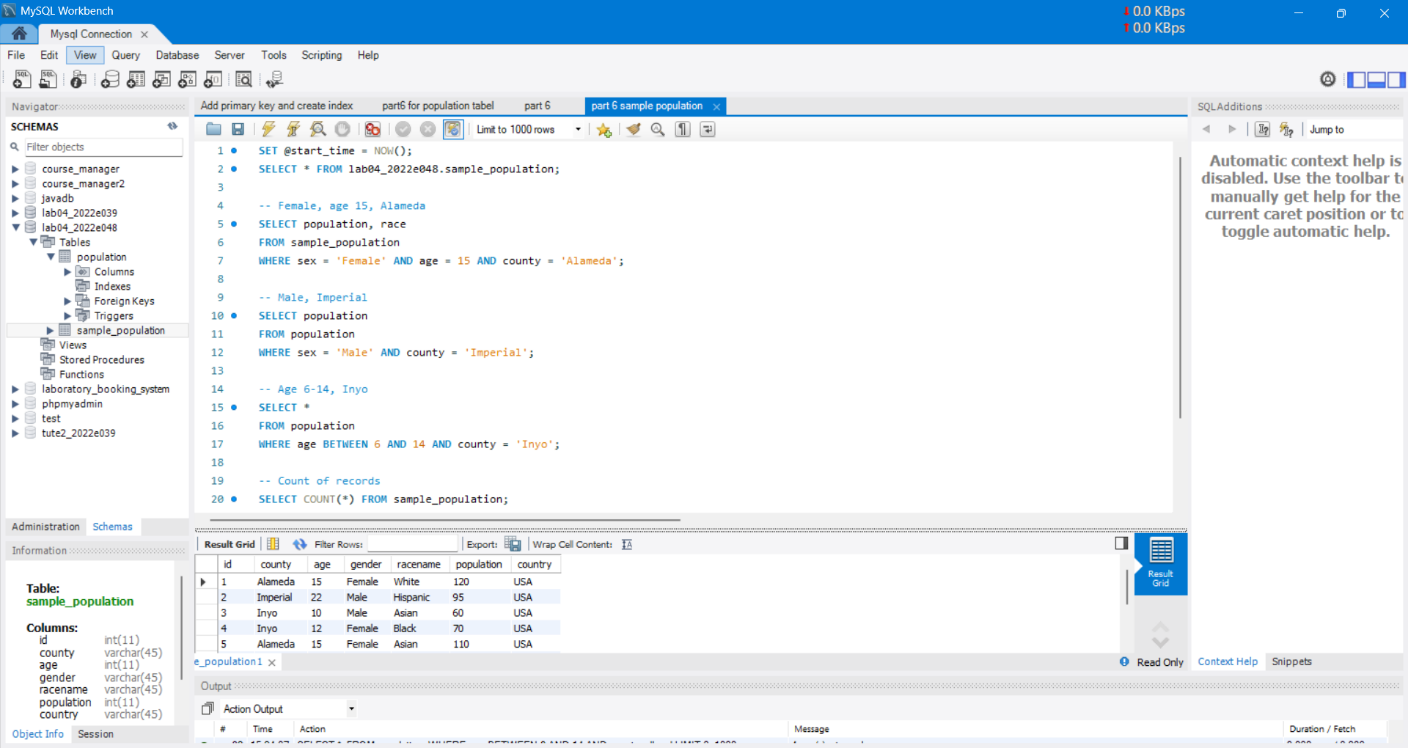
Observation for Q3 & Q4:

The import using the wizard was easier and good for small files. However, the query-based import (LOAD DATA INFILE) was faster for larger files. It required correct file path and permissions but gave better control and performance for repeated use.

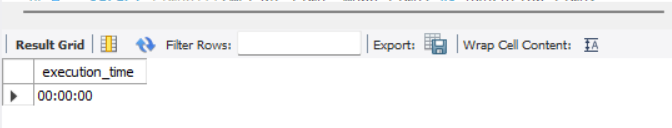
**Observations on Data Import (Wizard vs. Queries):**

* **Ease of Use for Beginners:** The **Data Import Wizard** (Question 3) is generally more user-friendly for those new to databases or SQL. Its graphical interface and step-by-step guidance reduce the barrier to entry, making it easy to perform imports without memorizing SQL syntax or command structures. You visually select files, tables, and map columns.
* **Control and Automation:** Importing data using **SQL queries (LOAD DATA INFILE)** (Question 4) offers much greater control and is essential for automation. With LOAD DATA INFILE, we have precise command over delimiters, line endings, error handling (e.g., REPLACE or IGNORE), and column mapping. This method is highly scriptable, meaning we can include it in shell scripts, Python scripts, or other batch processes to automate regular data imports without manual intervention.
* **Troubleshooting:** Errors in the wizard are often presented in a more digestible format, but troubleshooting can be a "black box" if we don't understand the underlying SQL. With LOAD DATA INFILE, errors might be less descriptive but are often directly related to the SQL syntax or file issues, which is more transparent for a user familiar with SQL.
* **Performance (for large files):** For very large CSV files, LOAD DATA INFILE is typically much faster and more resource-efficient than inserting row by row (which some wizards might do internally, though many now use optimized methods). It's designed for bulk loading.

Q6. Attach the screen shots for each queries given in question 6.

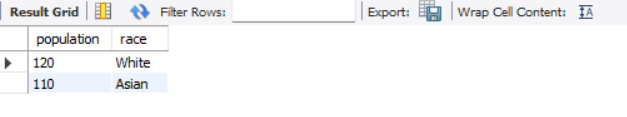
**CODE**

**EXECUTION TIME FOR POPULATION TABLE**

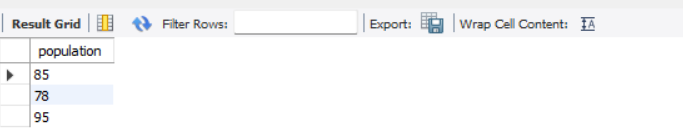
****

**FOR POPULATION TABLE**

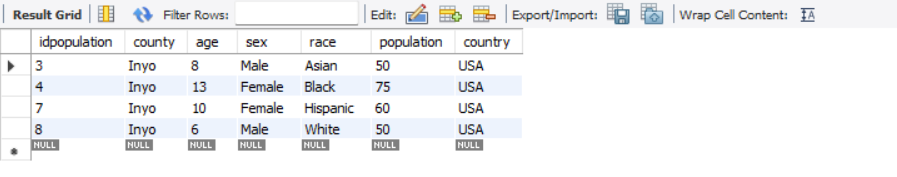
6-i



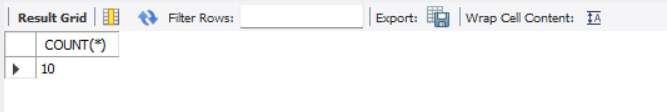
6.ii



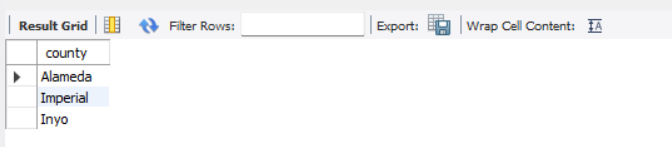
6.iii



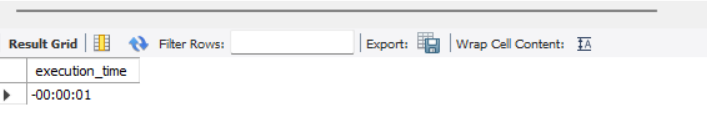
6.iv)



6.v)

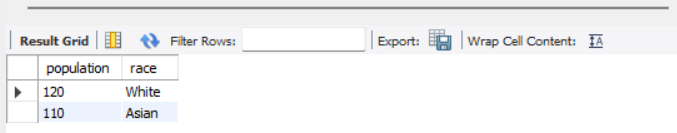


**EXECUTION TIME SAMPLE\_POPULATION TABLE**

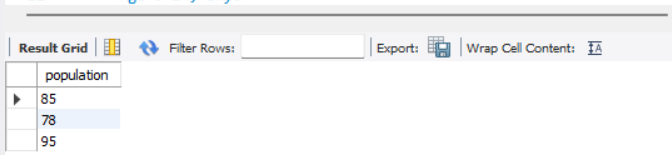


**FOR SAMPLE\_POPULATION TABLE**

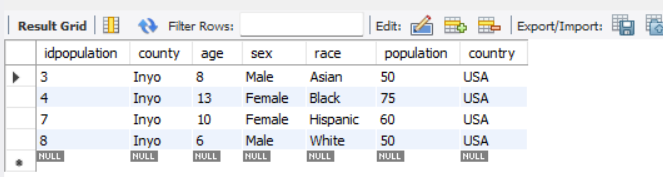
6.i



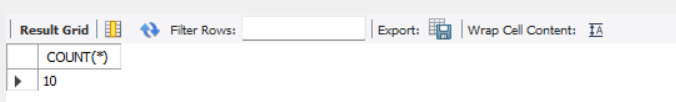
6.ii



6.iii



6.iv



6.v

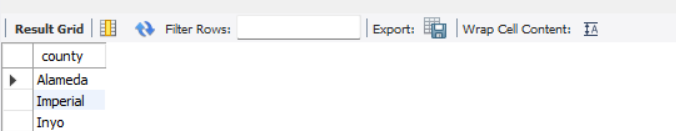


FIGURE 05: OUTPUT

A screenshot of a computer

AI-generated content may be incorrect.Q7. Screenshot for create the primary key for two tables and the listed queries in question 4.

FIGURE 06: IMPORT PRIMARY KEYS FOR TWO TABLES

A screenshot of a computer

AI-generated content may be incorrect.

FIGURE 07: AGAIN GET THE TIME PART 06

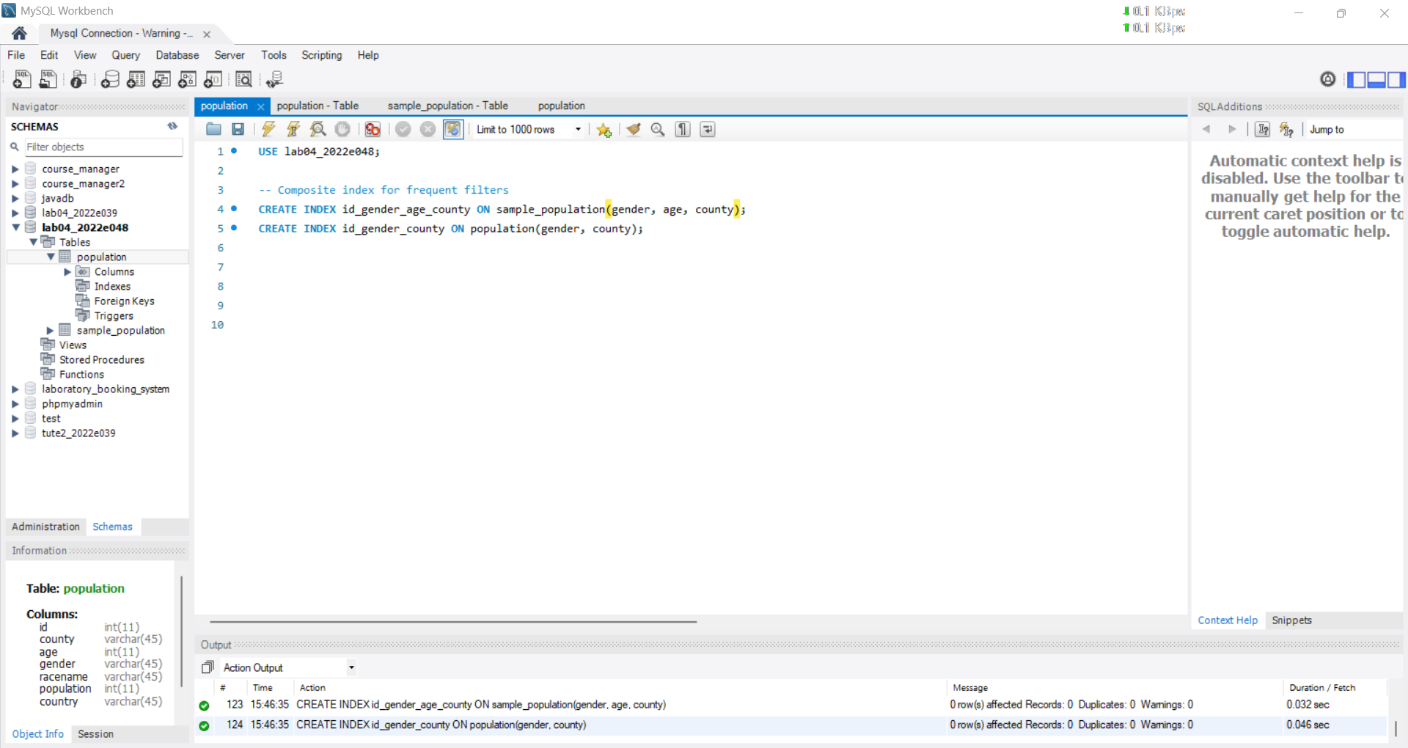
Q8. Screenshot for create the secondary index for these two tables and do the queries again (in question 4) and the time duration.

FIGURE 08: CREATE THE SECONDARY INDEX FOR THESE TWO TABLES

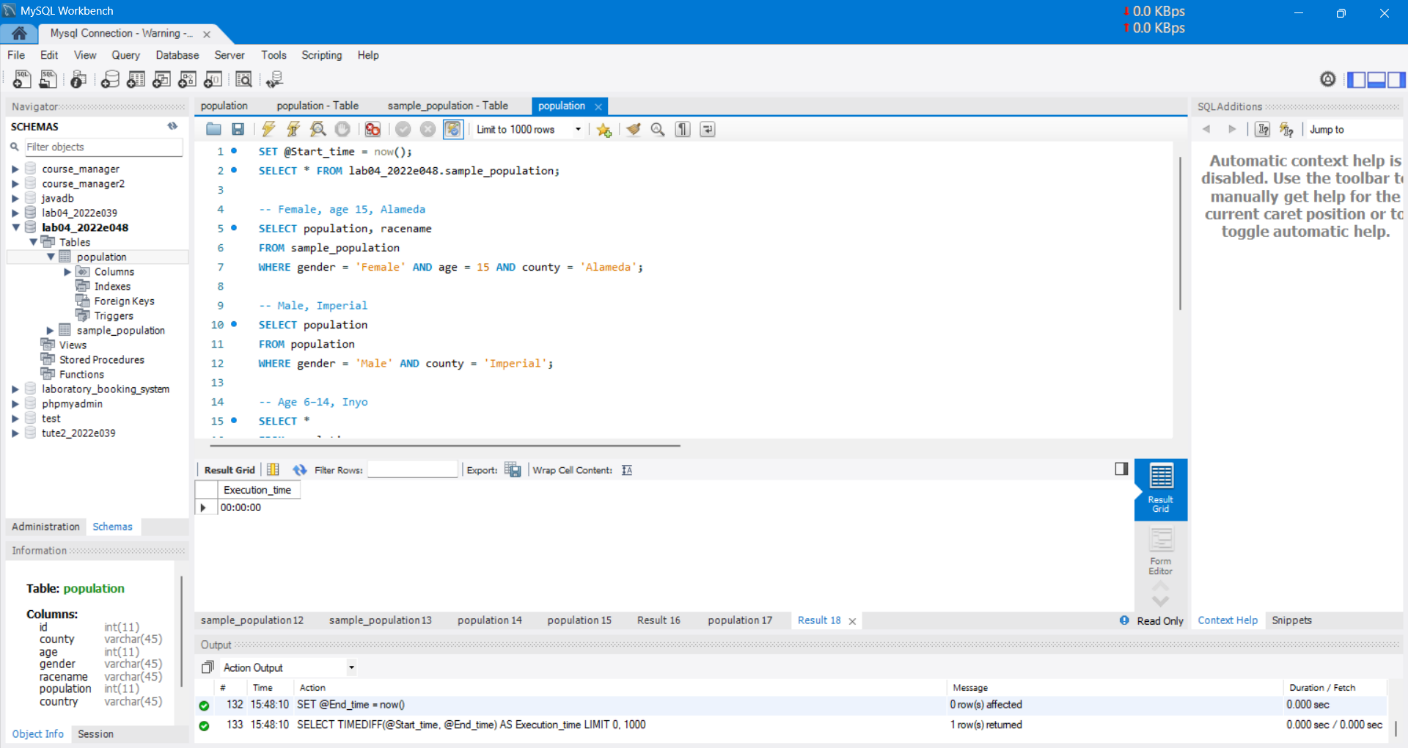


FIGURE 09: AGAIN GET THE TIME PART 06

Q9. Explanation for question 4,5,6.

**Final Observations (Q4, Q5, Q6):**  
Without indexing, the queries took longer, especially when filtering on multiple columns. After adding a **primary key**, we noticed a slight performance improvement on queries using id. However, the **secondary indexes** on gender, county, and age drastically reduced execution time, especially for complex WHERE clauses. This shows the effectiveness of indexes for large datasets in real-time querying.