Mini Project Report

ON

"MySQL Database GUI Application"

In Partial Fulfilment of the Requirement for the Award of

FOURTH YEAR COMPUTER ENGINEERING

BY

Mr. Sushant Said (BECOMP A-54)
Mr. Mohammad Jaid Mulani (BECOMP A-42)

UNDER THE GUIDANCE OF
Prof.Dnyaneshwar Choudhari

Department of Computer Engineering
Pimpri Chinchwad College of Engineering and Research
Pune - 412101
[2020-2021]

PIMPRI CHINCHWAD EDUCATION TRUST'S

Pimpri Chinchwad College of Engineering and Research, Pune - 412101

CERTIFICATE

This is certify that the mini project report entitled

"MySQL Database GUI Application"

submitted by

Mr. Sushant Said (BECOMP-A54)
Mr. Mohammad Jaid Mulani (BECOMP-A42)

has successfully completed the mini project entitled "***MySQL Database GUI Application***" in the fulfillment of B.E.(Computer Engineering) LP II and this work has been carried out in my presence.

Date: / /
Place:

Prof.Dnyaneshwar Choudhari
Project Guide
Department of Computer Engineering

Prof.Dr.Archana Chaugule
HOD,(Computer Department)
PCCOE&R, Ravet

Prof. Dr. Tiwari H.U.
Principal,PCCOE&R, Ravet

ACKNOWLEDGEMENT

It gives me great pleasure to present mini project on **"MySQL Database GUI Application"**. In preparing this report number of hands helped me directly and indirectly. Therefore, it becomes my duty to express my gratitude towards them.

I am very much obliged to subject guide **Prof. Dnyaneshwar Choudhari** in Computer Engineering Department, for helping me and giving me proper guidance. I will fail in my duty if I won't acknowledge a great sense of gratitude to the Head of Department **Dr. Archana Chaugule** and the entire staff members in for their cooperation.

I am also thankful to my family for their whole hearted blessings are always for me support and constant encouragement towards the fulfillment of the work. I wish to record the help extended to be my friends in all possible ways and active support and constant encouragement.

Place: Ravet, Pune **Sushant Said (A-54)**
Date: **Mohammad Jaid Mulani (A-42)**

ABSTRACT

The MySQL Workbench GUI Application is small window based application designed using tkinter that is tk interface package which helps building GUI features & also mysql-connector package as MySQL driver to connect with GUI Application as backend. The necessary communication with mysql database along PORT 3306 for CRUD operations and other features. The application is then tested using Pyunit or unittest and the test cases are manually designed to perform testing whether the application runs properly or not.

Thus, the buttons,labels, textfields used are checked using unittest and assured that the window-based application works perfectly.

Contents

List of Figures

Chapter 1

Introduction

1.0.1 MySQL Database GUI Application

The MySQL Database GUI Application is a window based application where the user can interact with MySQL database running on PORT 3306 using credentials of MySQL Server. Here list of GUI features helps user to interact with Databases content Tables Content and all Data inside them Using dropdown listview buttons notebook tab etc GUI features which help user to communicate fast with database which would ultimately help in rapid development process of other applications. For higher power features of SQL example Joins another SQL Shell GUI is developed for performing the same Query on Query Engine of MySQL Server.

1.1 Test objectives

The general objective of this study is to create an application for Optimizing User Interaction with MySQL database faster and effectively.

The specific objectives of this study are to:

(1) Implement a Interact with Content of Server with list of numerous Databases, Tables and Data.
(2) Implementation of the SQL Shell Optimisation Query generation execution.
(3) Test functionality of the whole system.

1.2 Testing

Testing is the process of checking the functionality of an application to ensure it runs as per requirements. Unit testing comes into picture at the developers' level; it is the testing of single entity (class or method). Unit testing plays a critical role in helping a software company deliver quality products to its customers.Unit Testing can be done in two ways.
Manual Testing.
Automated Testing.

1.2.1 Manual Testing

Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user whereby they use most of the application's features to ensure correct behavior. To guarantee completeness of testing, the tester often follows a written test plan that leads them through a set of important test cases. Executing a test cases manually without any tool support is known as manual testing. No programming can be done to write sophisticated tests to fetch hidden information. As test cases need to be executed manually, more testers are required in manual testing. Less reliable Manual testing is less reliable

GUI objects size difference and color combination etc is not easy to find out in manual testing. Load testing and performance testing is not possible in manual testing. Running test manually is very time consuming job.Regression Test cases are time consuming if it is manual testing. Manual testing is the oldest and most rigorous type of software testing. Manual testing requires a tester to perform manual test operations on the test software without the help of Test automation. Things such as device drivers and software libraries must be tested using test programs. In addition, testing of large numbers of users (performance testing and load testing) is typically simulated in software rather than performed in practice.

1.2.2 Automated Testing

Automation testing is a Software testing technique to test and compare the actual outcome with the expected outcome. This can be achieved by writing test scripts or using any automation testing tool. Test automation is used to automate repetitive tasks and other testing tasks which are difficult to perform manually. What is Automation Testing? Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

Automation Testing means using an automation tool to execute your test case suite.

The automation software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Test Automation demands considerable investments of money and resources.

Successive development cycles will require execution of same test suite repeatedly. Using a test automation tool, it's possible to record this test suite and re-play it as required. Once the test suite is automated, no human intervention is required. This improved ROI of Test Automation. The goal of Automation is to reduce the number of test cases to be run manually and not to eliminate Manual Testing altogether.Manual Testing of all workflows, all fields, all negative scenarios is time and money consuming It is difficult to test for multilingual sites manually.Automation does not require Human intervention. You can run automated test unattended (overnight). Automation increases the speed of test execution. Automation helps increase Test Coverage. Manual Testing can become boring and hence error-prone.

1.3 Pyunit

1.3.1 Introduction

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit that originated with JUnit.

The Python unit testing framework, sometimes referred to as "PyUnit," is a Python language version of JUnit developed by Kent Beck and Erich Gamma. PyUnit forms part of the Python Standard Library as of Python version 2.1.

Python unit testing framework supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework. The unittest module provides classes that make it easy to support these qualities for a set of tests.

Features of PyUnit The definitions of test cases and test suites in the same modules as the code they are to test (e.g. 'widget.py'), but there are several advantages to placing the test code in a separate module, such as 'widgettests.py':

The test module can be run standalone from the command line.

The test code can more easily be separated from shipped code.

There is less temptation to change test code to fit the code it tests without a good reason.

Test code should be modified much less frequently than the code it tests.

Tested code can be refactored more easily.

Tests for modules written in C must be in separate modules anyway, so why not be consistent? If the testing strategy changes, there is no need to change the source code You can access to all funtionalities and can find more information at https://pypi.org/project/pytheons.pyunit/

Chapter 2

Functional Requirements

In software engineering, a functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements are also called Functional Specification. Functional software requirements help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is required to perform.
Functional Requirements should include the following things:
Details of operations conducted in every screen
Data handling logic should be entered into the system
It should have descriptions of system reports or other outputs
Complete information about the workflows performed by the system
It should clearly define who will be allowed to create/modify/delete the data in the system
How the system will fulfill applicable regulatory and compliance needs should be captured in the functional document
Benefits of Functional Requirement
Helps you to check whether the application is providing all the functionalities that were mentioned in the functional requirement of that application A functional requirement document helps you to define the functionality of a system or one of its subsystems. Functional requirements along with requirement analysis help identify missing requirements. They help clearly define the expected system service and behavior. Errors caught in the Functional requirement gathering stage are the cheapest to fix.

During functional testing, Black Box Testing technique is used in which the internal logic of the system being tested is not known to the tester.Functional testing is normally performed during the levels of System Testing and Acceptance Testing. Typically, functional testing involves the following steps:

Identify functions that the software is expected to perform. Create input data based on the function's specifications. Determine the output based on the function's specifications. Execute the test case. Compare the actual and expected outputs. Functional testing is more effective when the test conditions are created directly from user/business requirements. When test conditions are created from the system documentation (system requirements/ design documents), the defects in that documentation will not be detected through testing and this may be the cause of end-users' wrath when they finally use the software

2.0.1 Best practice of Functional Requirement

- Do not combine two requirements into one. Keep the requirements granular.
- You should make each requirement as complete and accurate as possible.
- The document should draft all the technical requirements.
- Map all requirements to the objectives and principles which contributes to successful software delivery Elicit requirements using interviews, workshops and casual communications.
- If there is any known, verified constraint which materially affects a requirement then it is a critical state that should be documented.
- It is necessary that you document all the assumption in the document.

Chapter 3

Non Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture, because they are usually architecturally significant requirements.Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs. Proper definition and implementation of NFRs is critical. Over-specify them, and the solution may be too costly to be viable; under-specify or underachieve them, and the system will be inadequate for its intended use. An adaptive and incremental approach to exploring, defining, and implementing NFRs is a vital skill for Agile teams.This does not mean the latter are more important, but most requirement gathering techniques focus on functional requirements, so large gaps in non-functional requirements are common.So what exactly are we looking for here? Well, here are four examples of Non-Functional requirement groups; usability, reliability, performance. A non-functional requirement is essential to ensure the us- ability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are more than 10000. Description of non-functional requirements is just as critical as a functional requirement.

3.1 Benefits of Non-Functional Requirement

Helps you to check whether the application is providing all the functionalities that were mentioned in the functional requirement of that application. A functional requirement document helps you to define the functionality of a system or one of its subsystems.Functional requirements along with requirement analysis help identify missing requirements The non-functional requirements ensure the software system fol- low legal and compliance rules.They ensure the reliability, availability, and performance of the software system. They ensure good user experience and ease of operating the software.

Chapter 4

Block Diagram

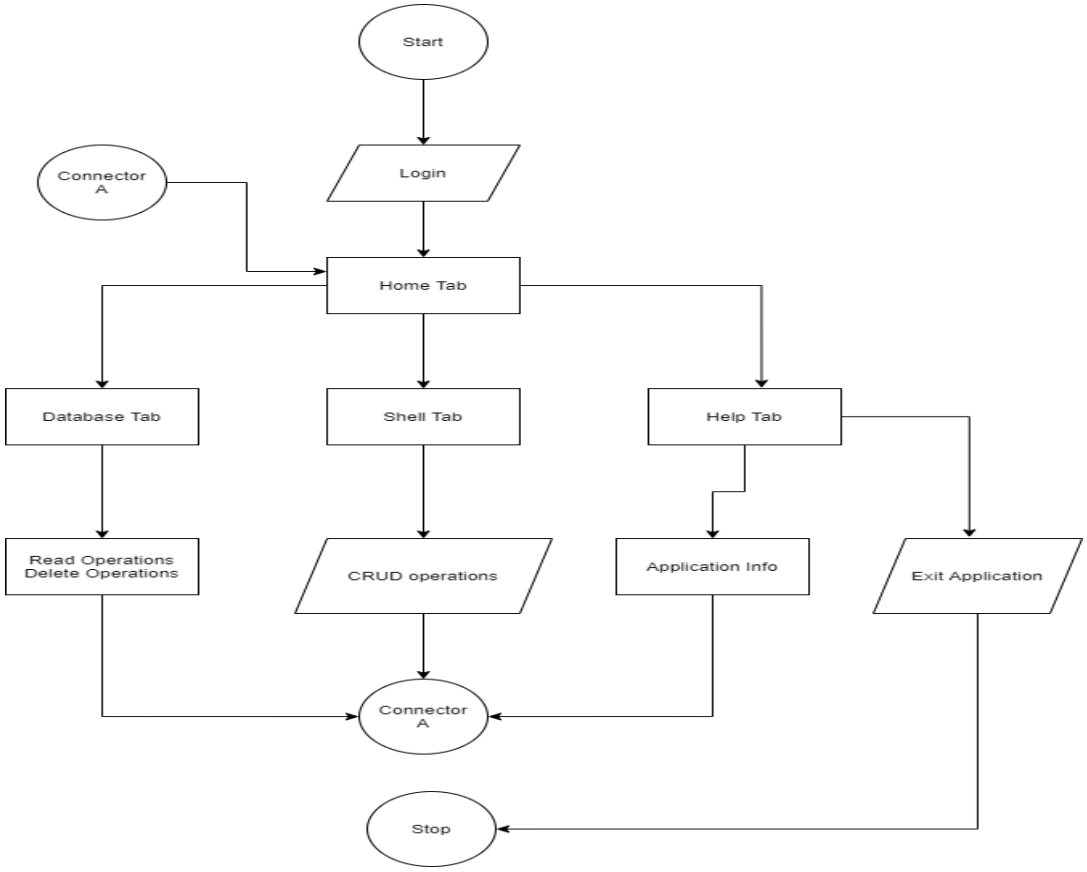

Figure 4.1: MySQL GUI Application

4.0.1 Test Strategy

A test strategy is an outline that describes the testing approach of the software development cycle. It is created to inform project managers, testers, and developers about some key issues of the testing process. This includes the testing objective, methods of testing new functions, total time and resources required for the project, and the testing environment.

Test strategies describe how the product risks of the stakeholders are mitigated at the test-level, which types of testing are to be performed, and which entry and exit criteria apply. They are created based on development design documents. System design documents are primarily used, and occasionally conceptual design documents may be referred to. Design documents describe the functionality of the software to be enabled in the upcoming release. For every stage of development design, a corresponding test strategy should be created to test the new feature sets.

The test strategy describes the test level to be performed. There are primarily three levels of testing: unit testing, integration testing, and system testing. In most software development organizations, the developers are responsible for unit testing. Individual testers or test teams are responsible for integration and system testing.

In Software Engineering, software release goes through Test Strategy documents from time to time to map the progress of testing in the right direction. When the release date is close many of these activities will be skipped, it is desirable to discuss with team members whether cutting down any particular activity will help for release without any potential risk.

A strategy plan for defining the testing approach, what you want to accomplish and how you are going to achieve it. This document removes all uncertainty or vague requirement statements with a clear plan of approach for achieving the test objectives. Test Strategy is one of the most important documents for the QA team. It helps Test managers to get the clear state of the project at any point. The chances of missing any test activity are very low when there is a proper test strategy in place.

Test execution without any plan rarely works. I know teams who write strategy document but never refer it back while test execution. The Testing Strategy plan must be discussed with the whole team so that the team will be consistent with the approach and responsibilities.

In tight deadlines, you can't just waive any testing activity due to time pressure. At least it must go through a formal process before doing so.

Chapter 5

Test Plan

Srno.	Test Case	Result	Function
1	SQL Object Creation	Pass	assertNotNull()
2	Login	Pass	assertEquals()
3	Connect To MySQL db	Pass	assertTrue()
4	Pass Invalid Database Connection	Pass	assertFalse()
5	Select valid Database	Pass	assertTrue()
6	Listing all databases	Pass	assertTrue()
7	Creating Database	Pass	assertTrue()
8	Select valid Database	Pass	assertTrue()
9	Describe Table Data	Pass	assertTrue()
10	Show Table Data	Pass	assertTrue()
11	Drop Table Data	Pass	assertTrue()
12	Drop Database	Pass	assertTrue()

Chapter 6

Output Screenshots

6.0.1 SQL Object Creation

The test case verifies whether the object is created successfully or not.

6.0.2 Login Button

The test case verifies whether the login button is clicked and verified credential is performed or not.

6.0.3 connecting to MySQL db

Home Tab navigation to various function is shown and displayed or not is tested.

6.0.4 SQL Databases Tab

The SQL Server functionality accessing Data through GUI content testing frontend, backend and displayed or not is tested.

6.0.5 SQL Shell Tab

The SQL Shell Server functionality handling Data content testing frontend, backend and displayed or not is tested.

6.0.6 Screenshots

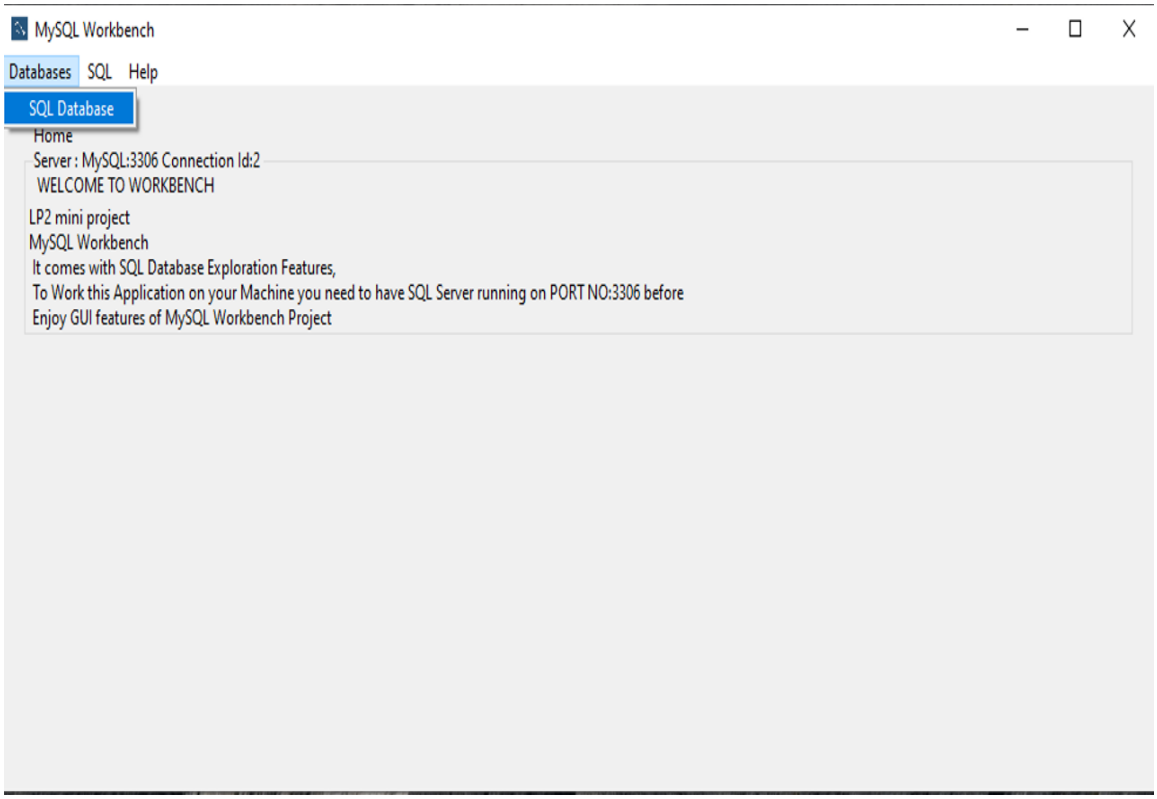

Figure 6.1: Application Main Screen

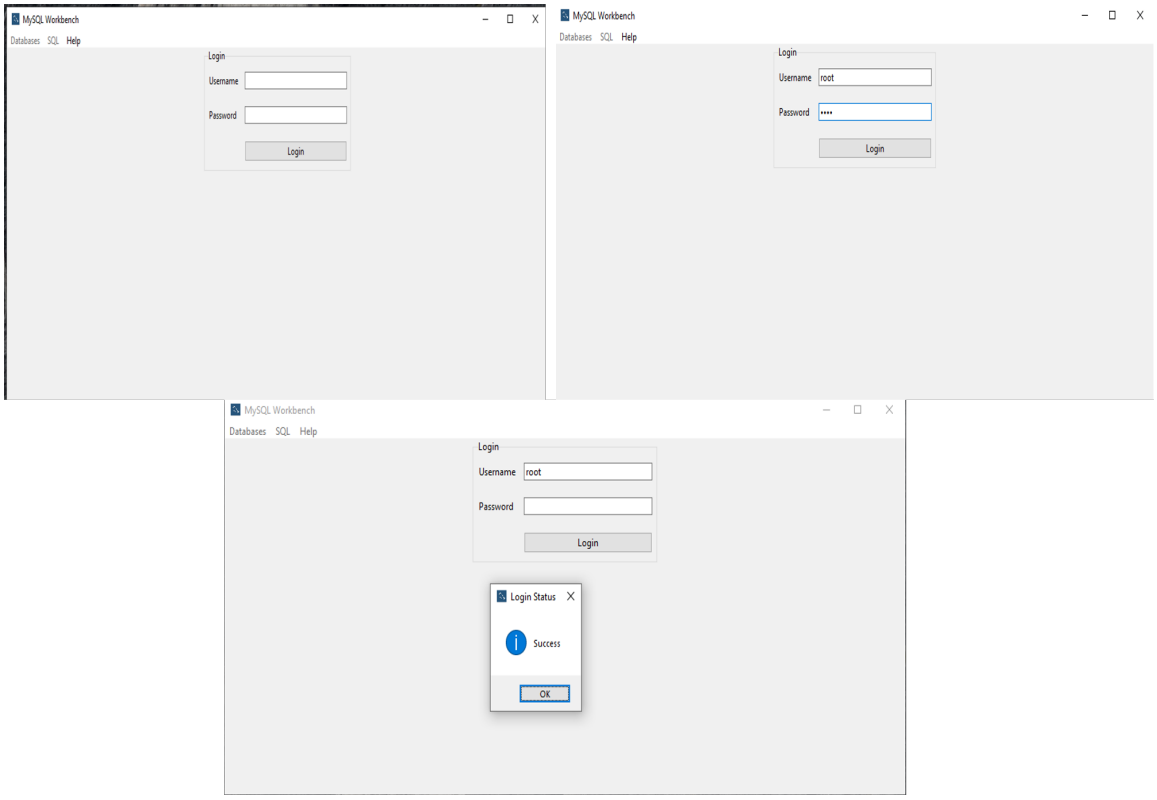

Figure 6.2: Login

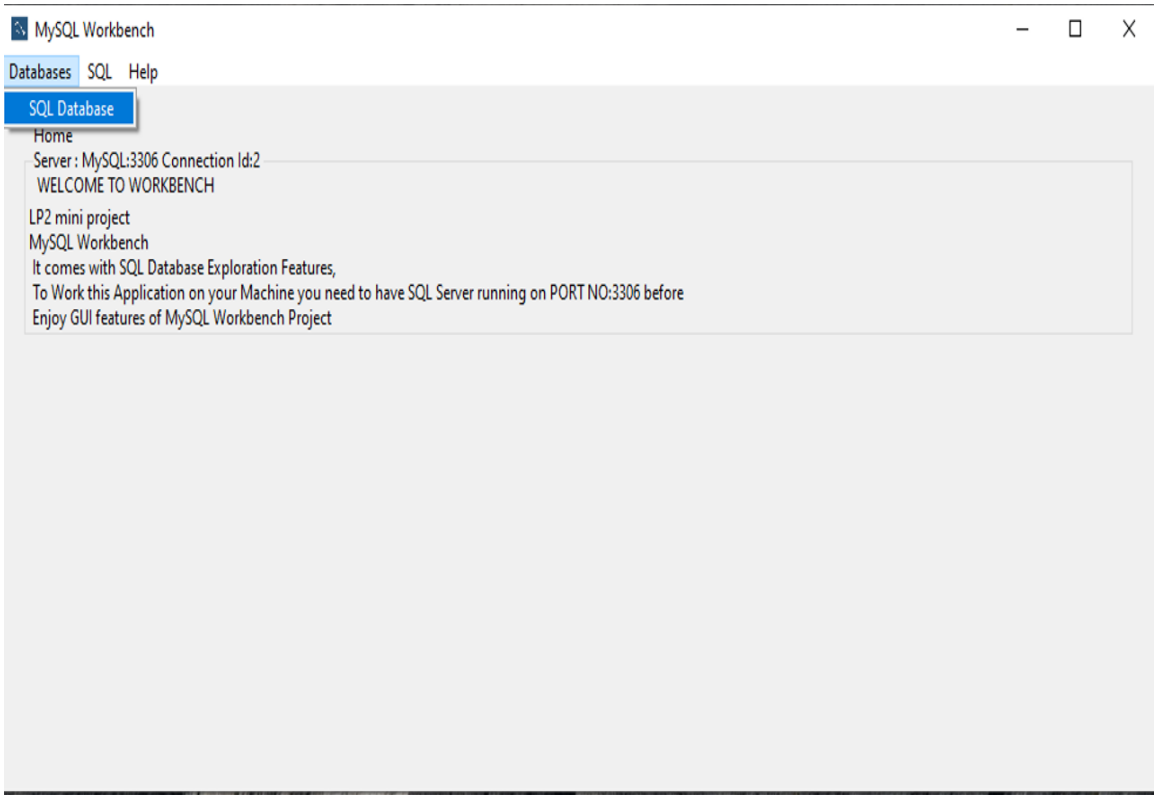

Figure 6.3: Home Page

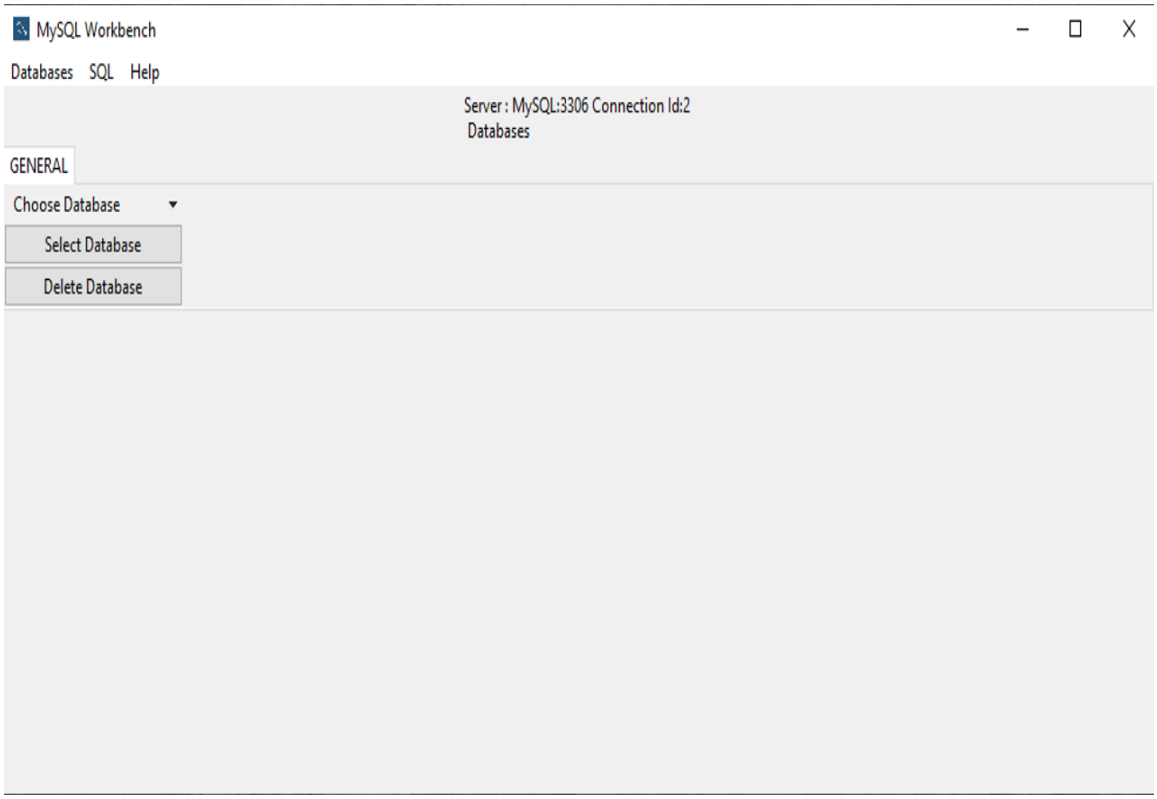

Figure 6.4: Database Tab

Figure 6.5: Choose Database

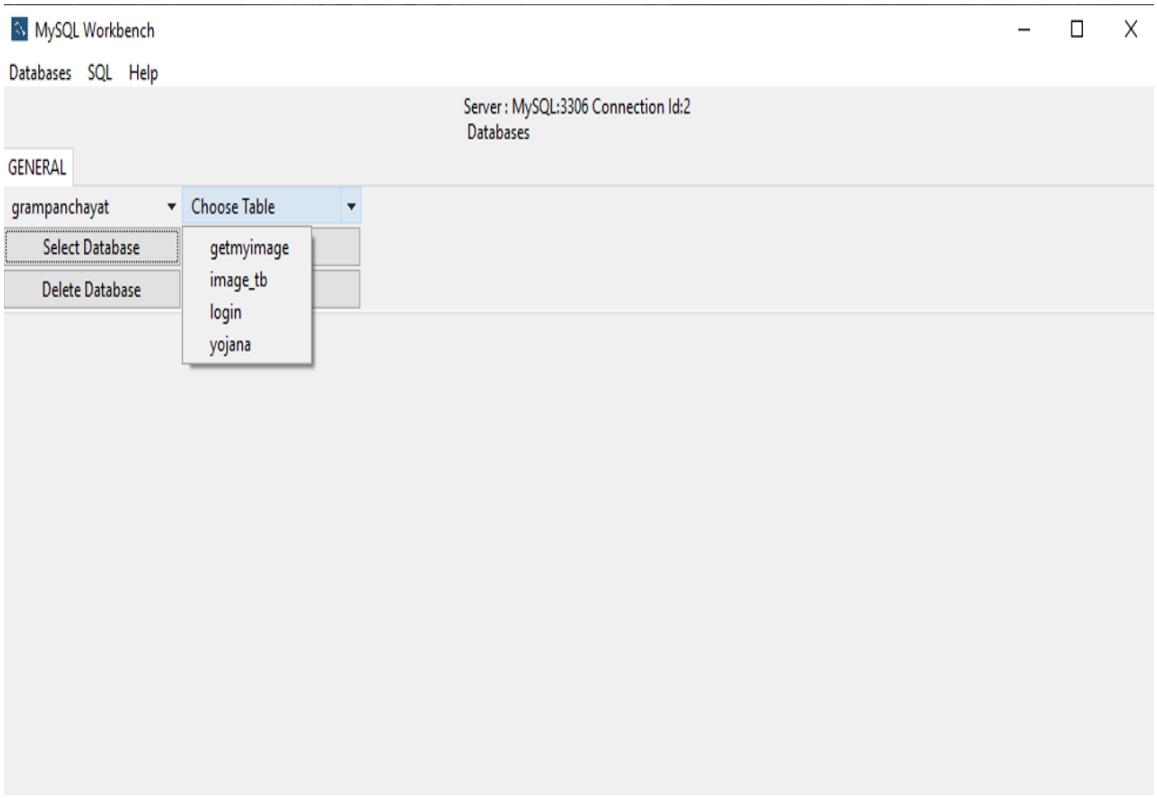

Figure 6.6: Choose Table

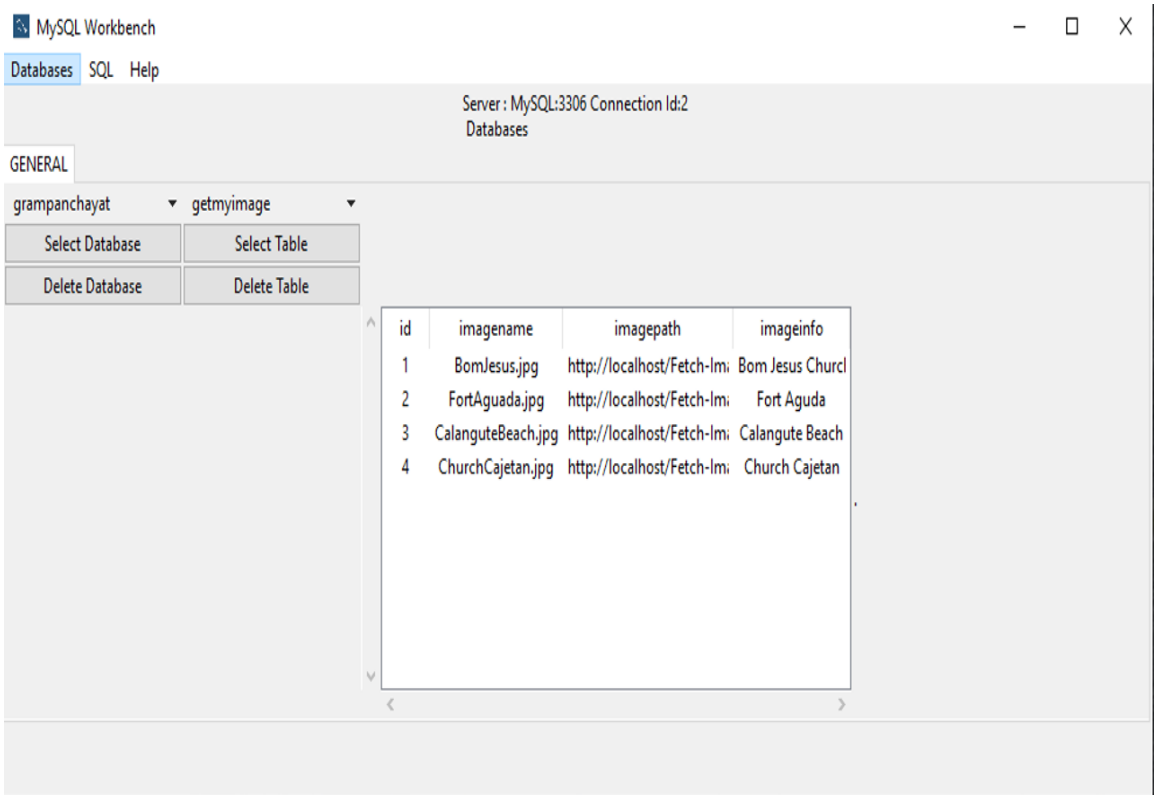

Figure 6.7: Display Data

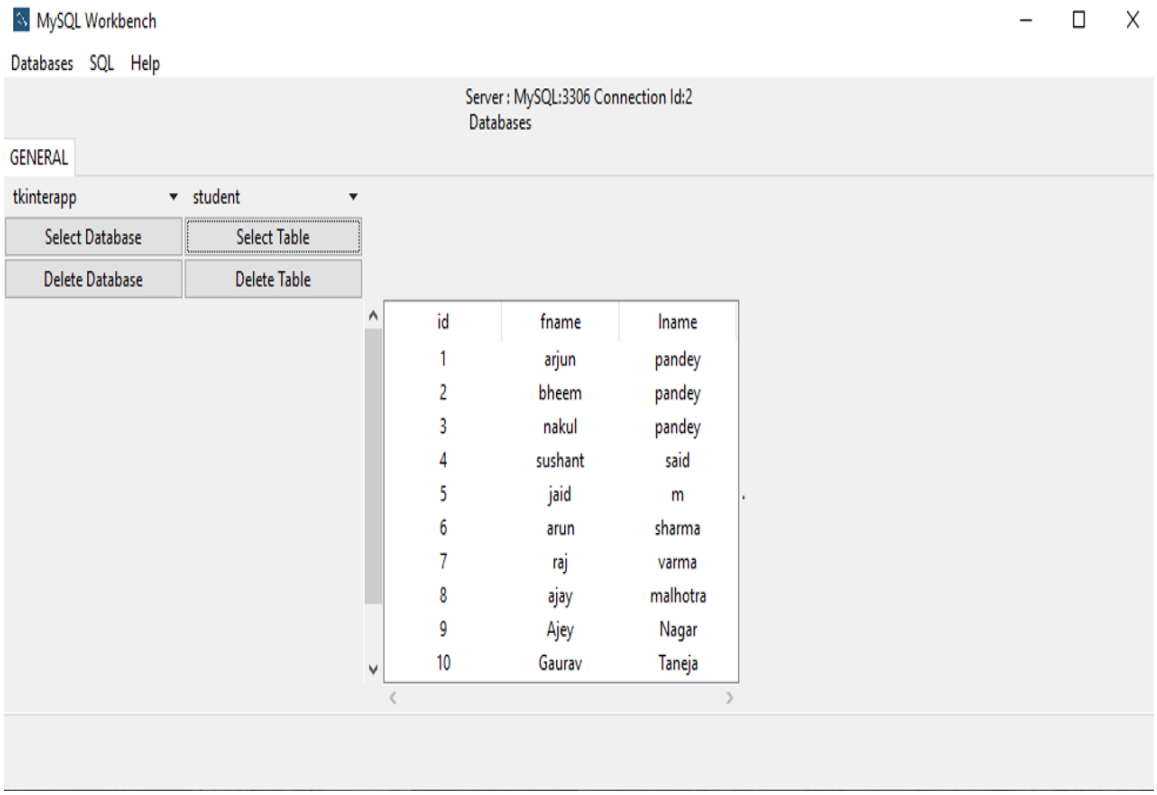

Figure 6.8: Display Data from another Database : Table Part 1

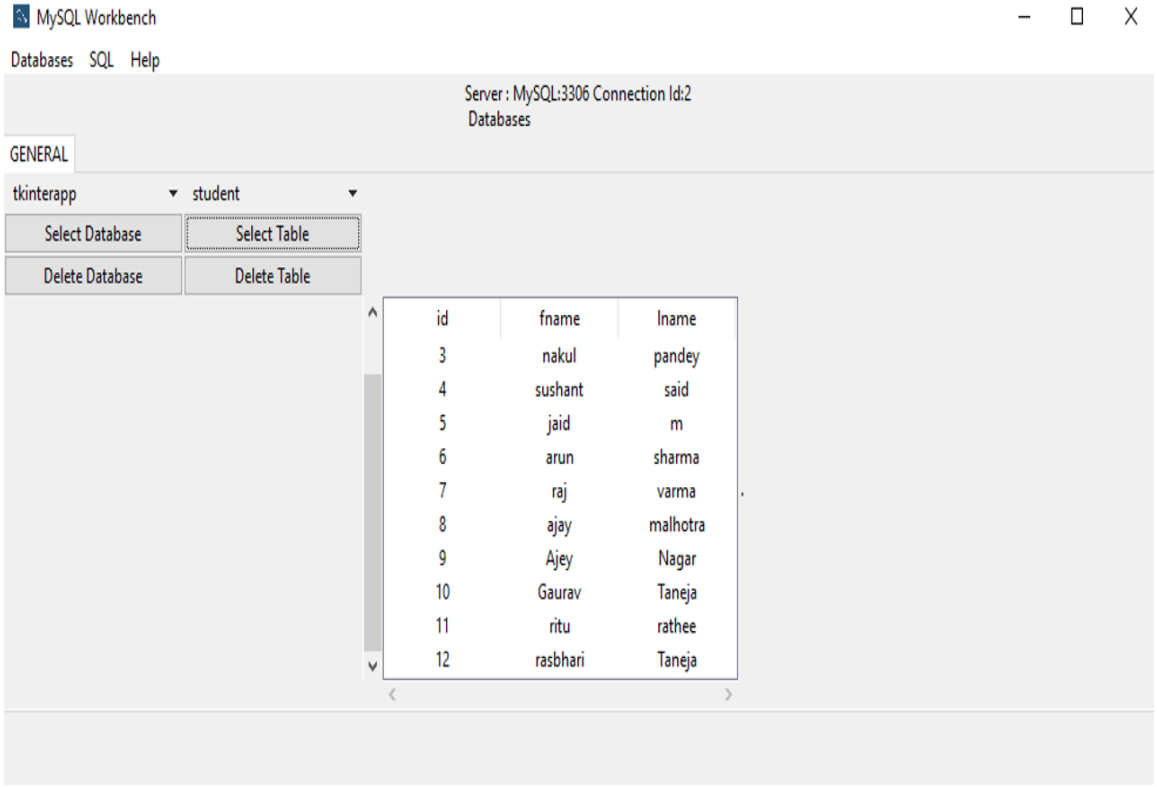

Figure 6.9: Display Data from another Database :Table Part 2

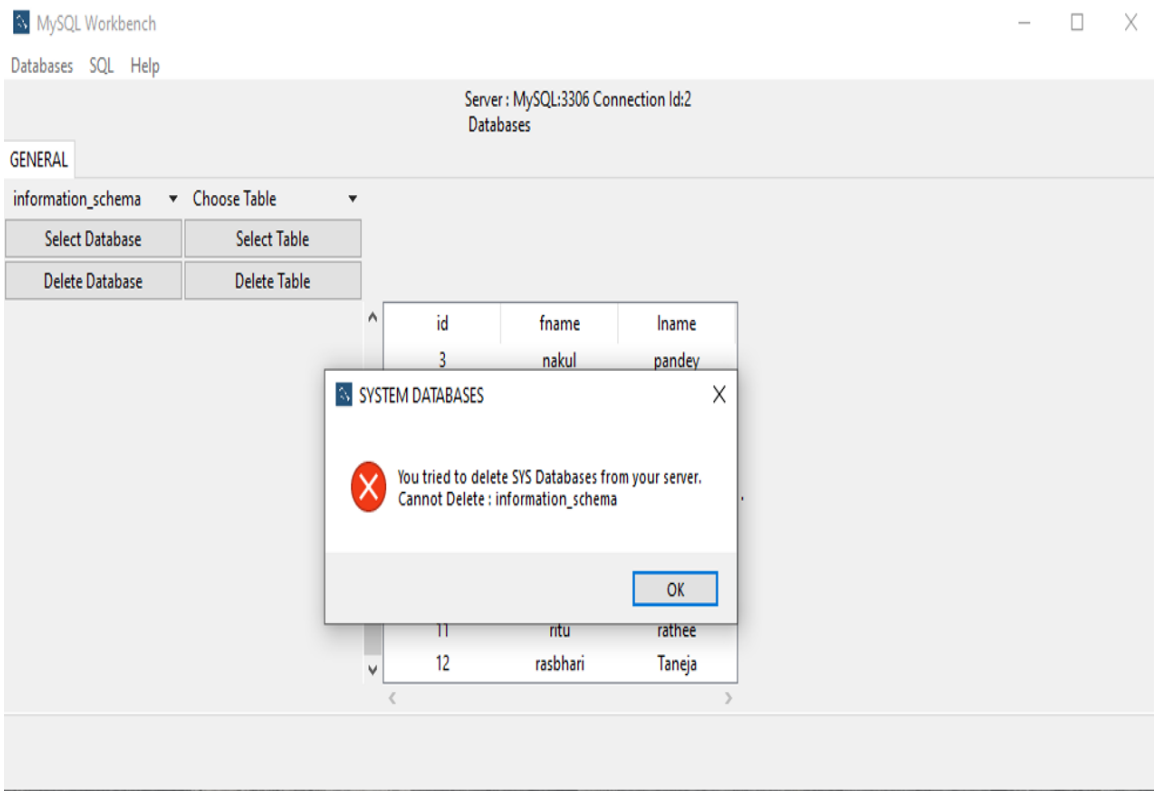

Figure 6.10: Deleting System Database

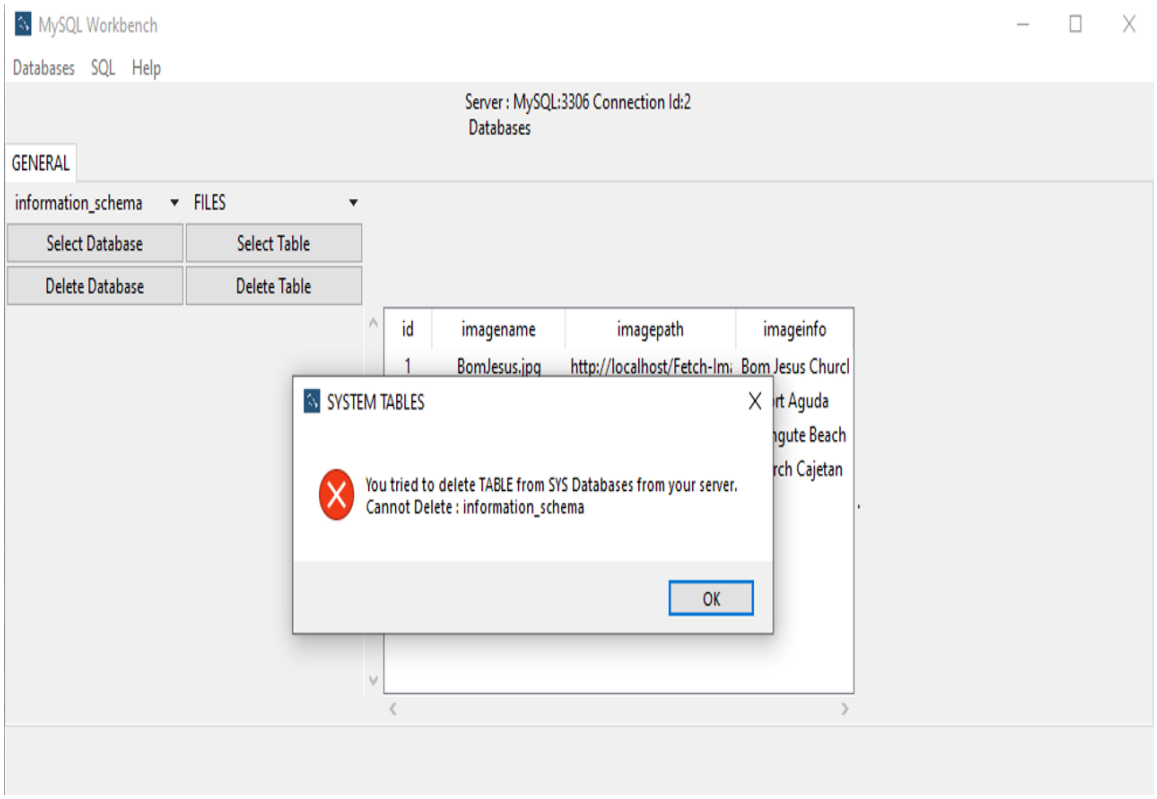

Figure 6.11: Deleting System Tables

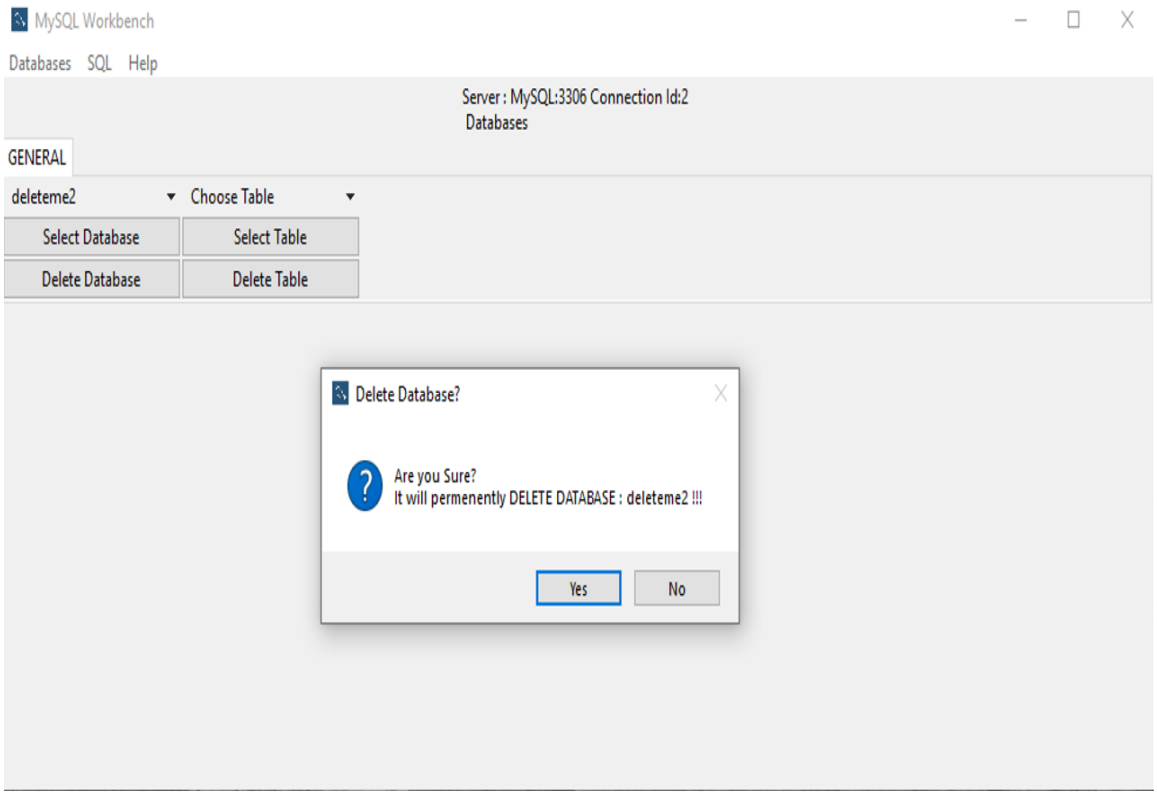

Figure 6.12: Delete Database

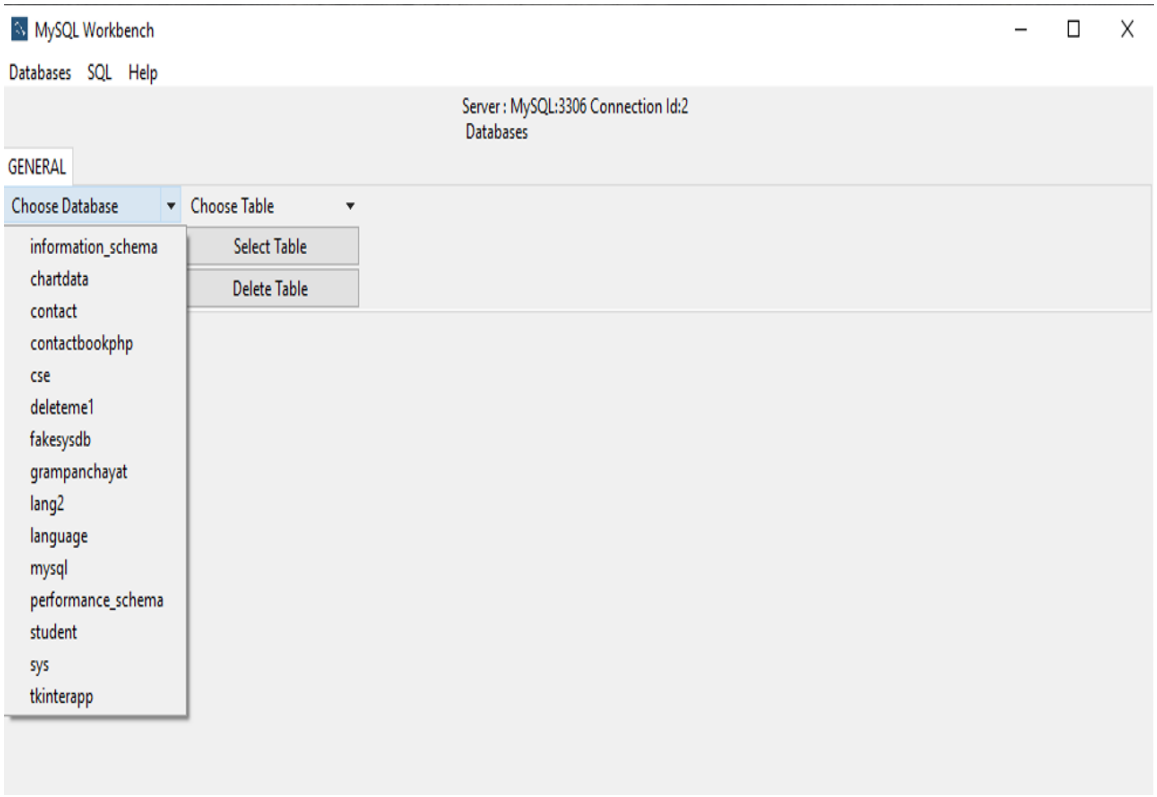

Figure 6.13: Check deleted Success

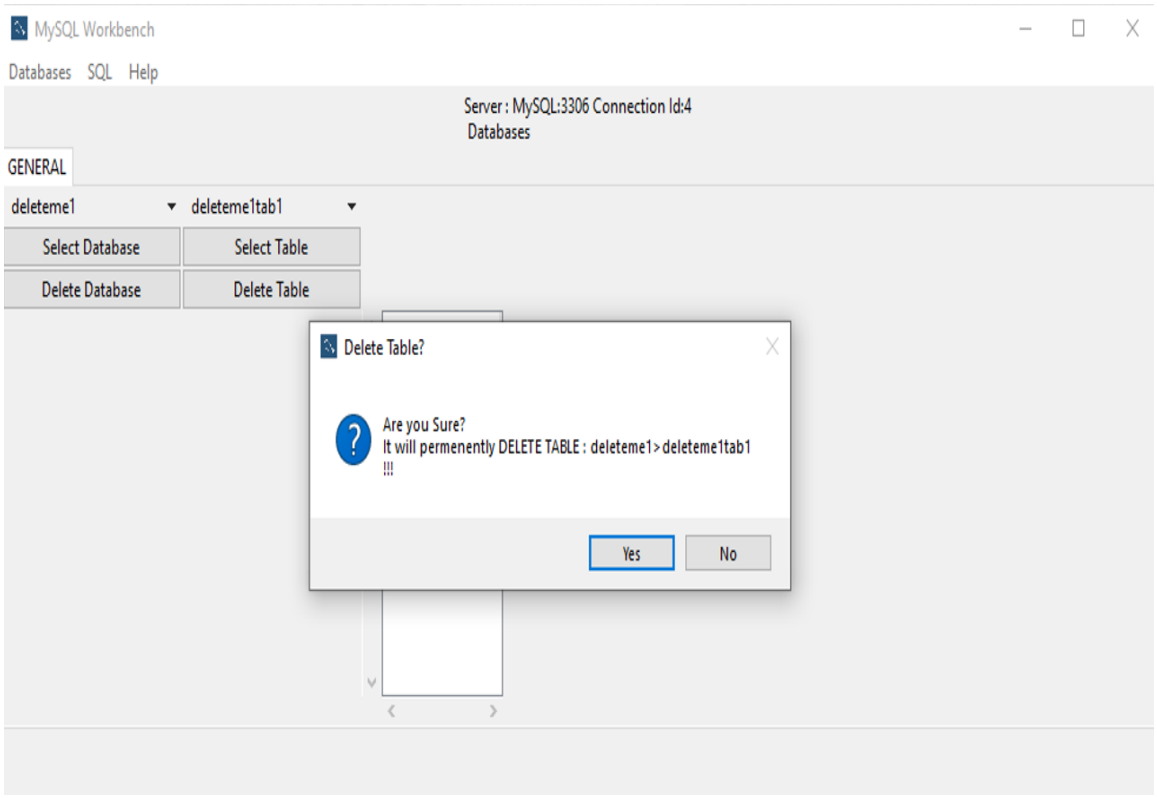

Figure 6.14: Delete Table

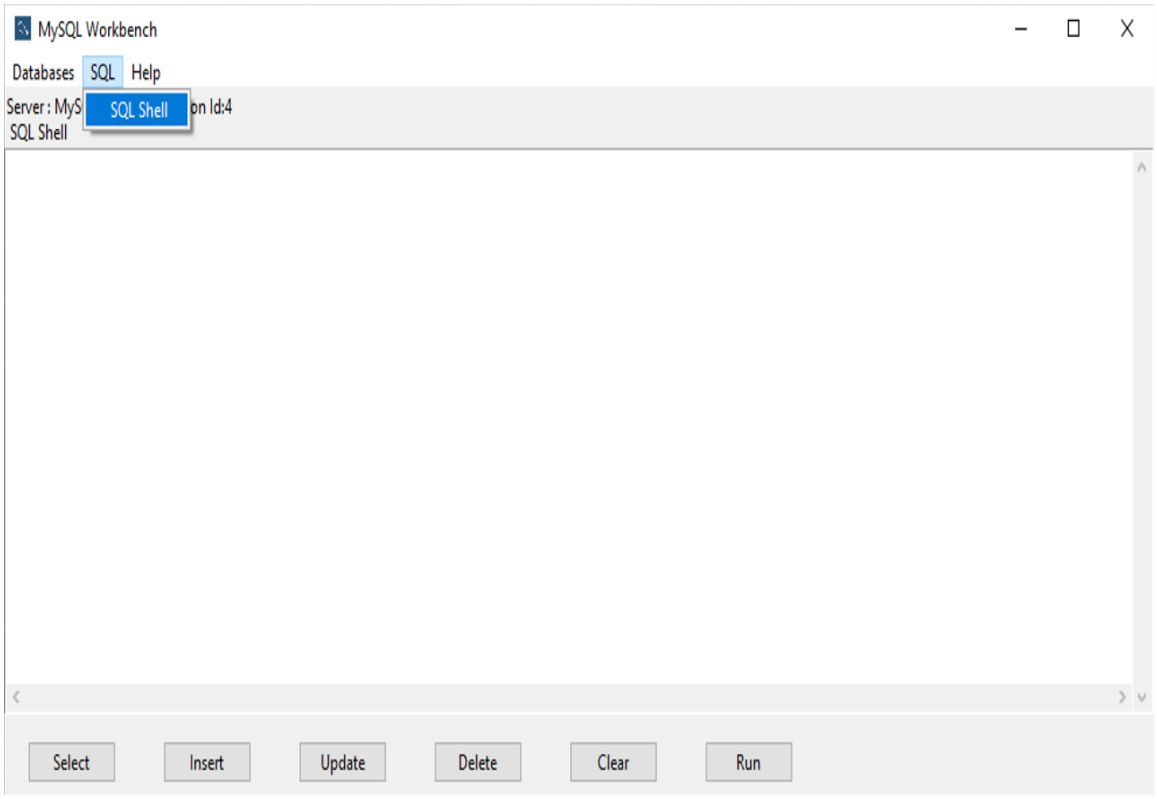

Figure 6.15: SQL Shell Tab

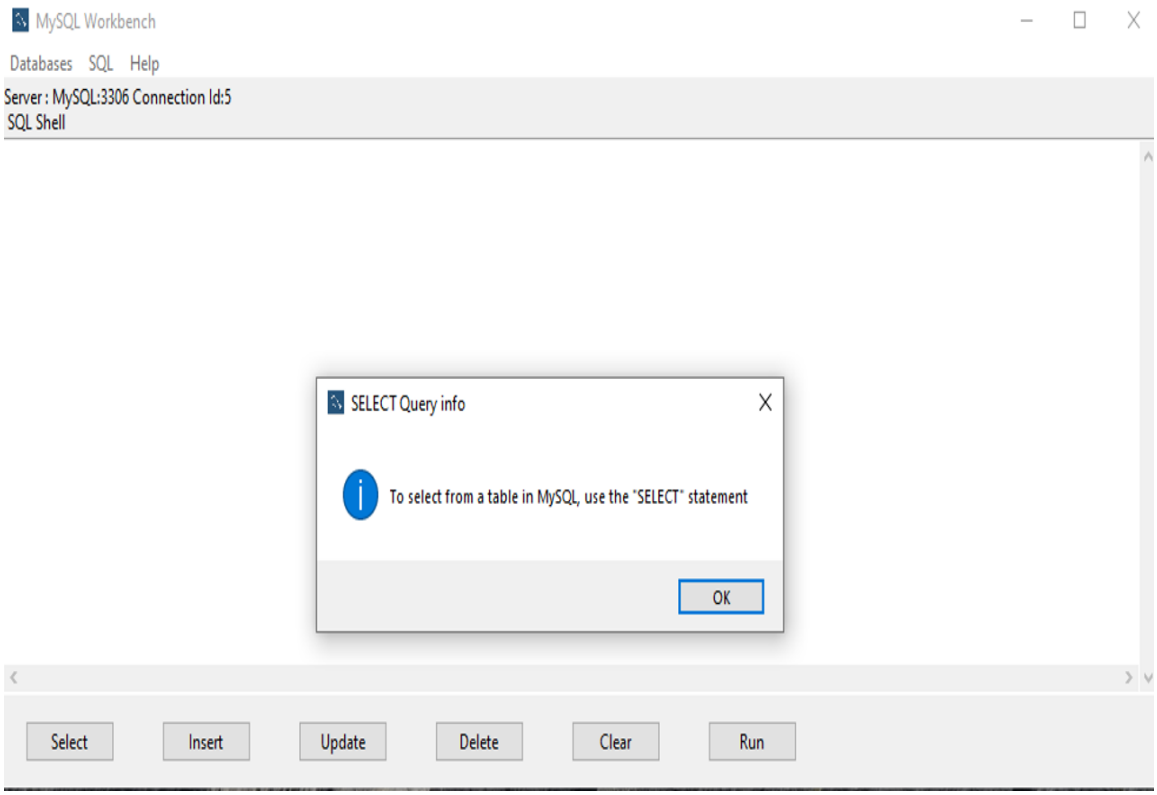

Figure 6.16: SQL Shell : Select Query info

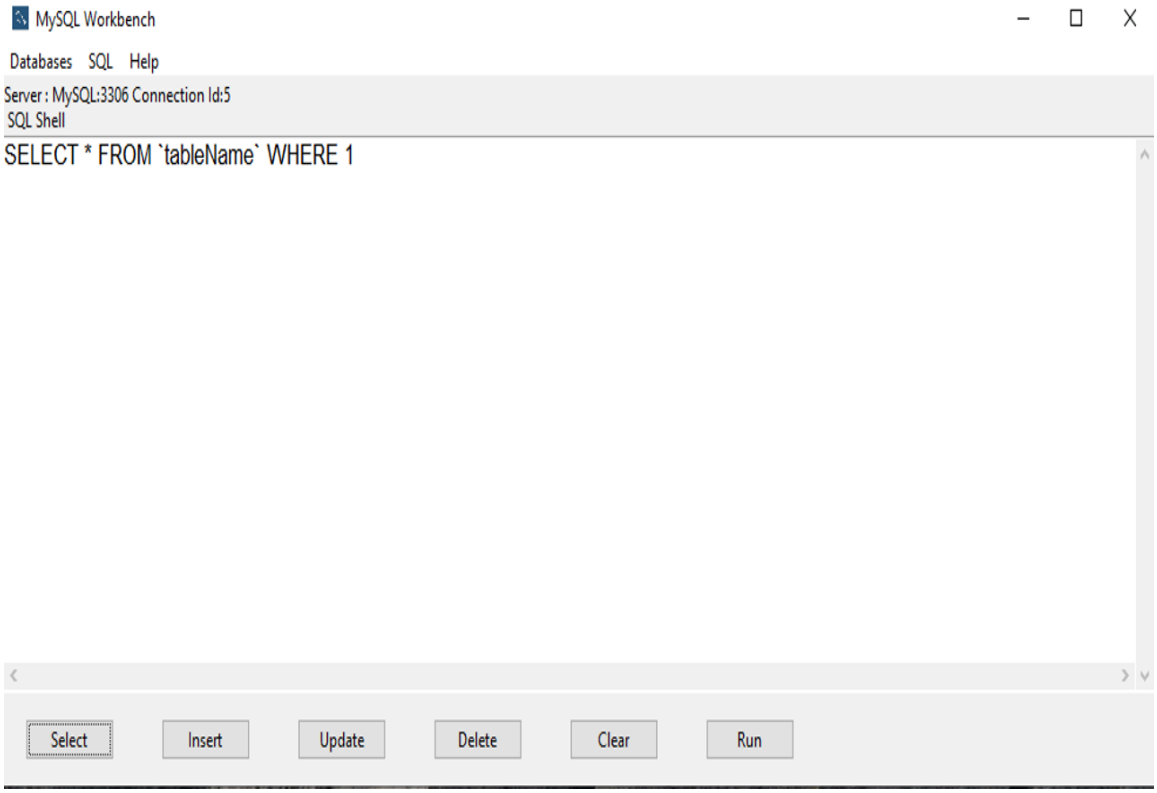

Figure 6.17: SQL Shell : Select Query

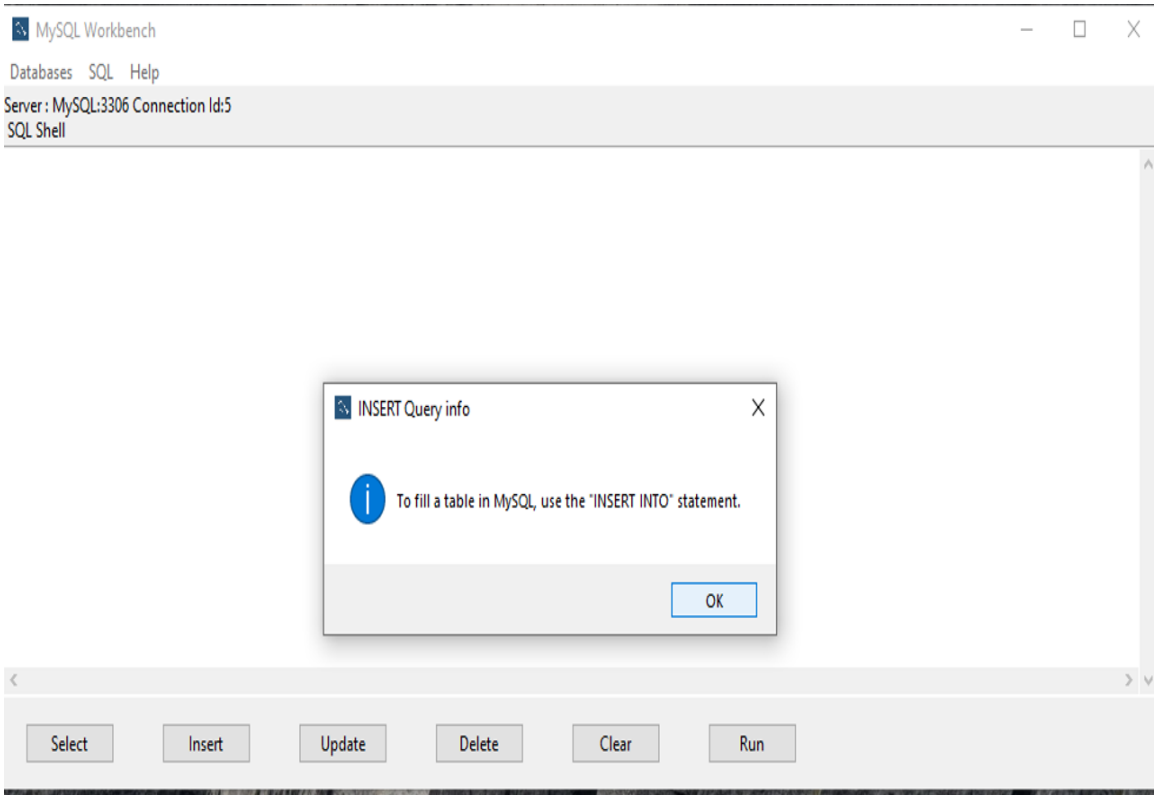

Figure 6.18: SQL Shell : Insert Query info

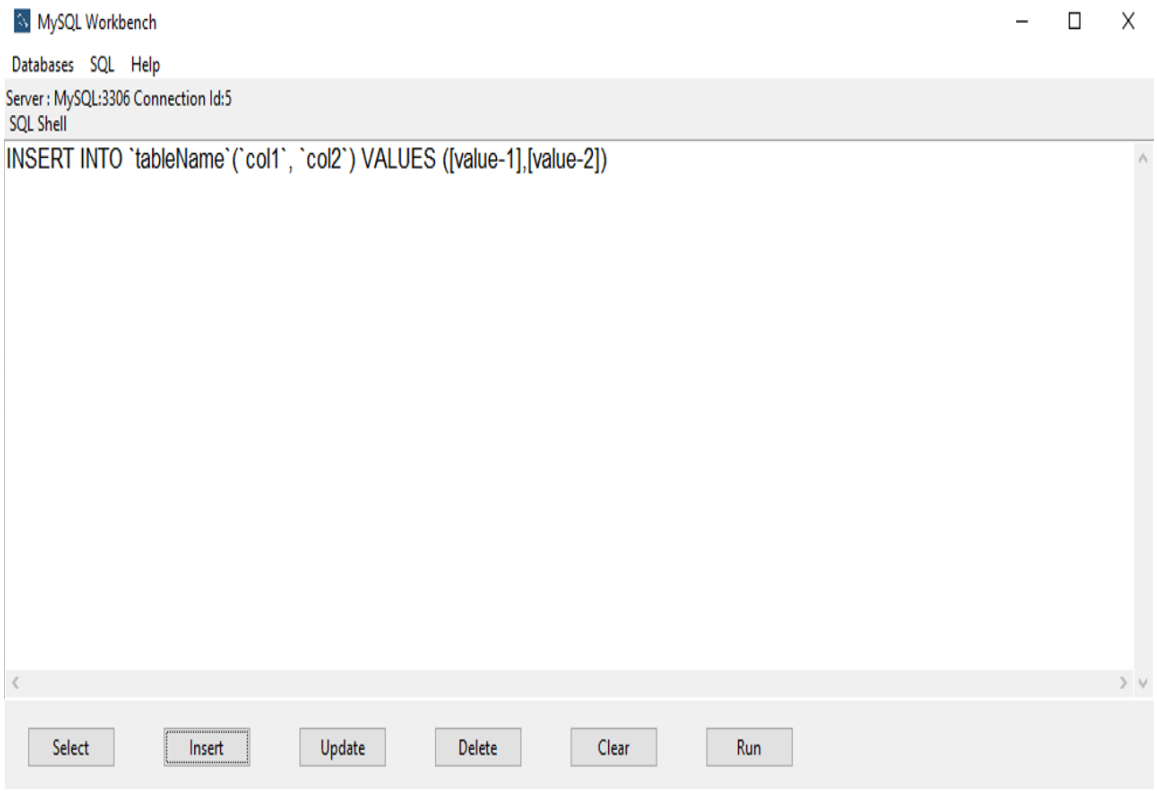

Figure 6.19: SQL Shell : Insert Query

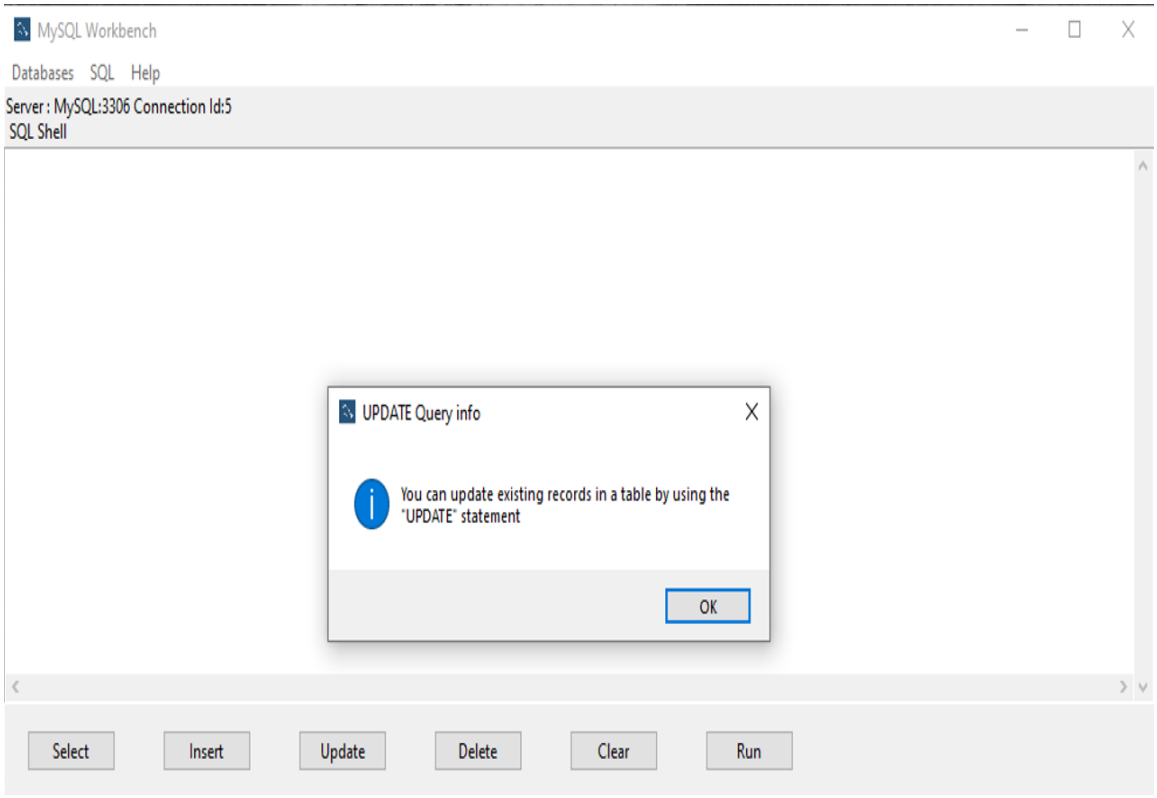

Figure 6.20: SQL Shell : Update Query info

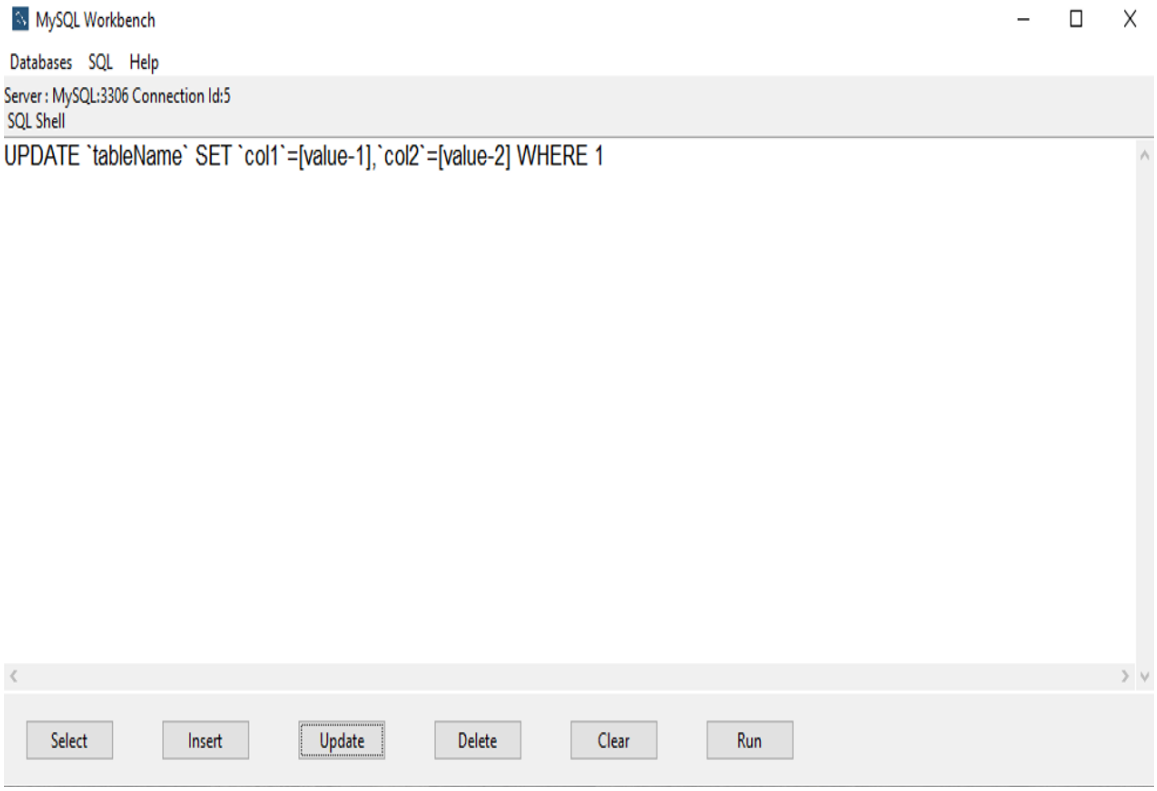

Figure 6.21: SQL Shell : Update Query

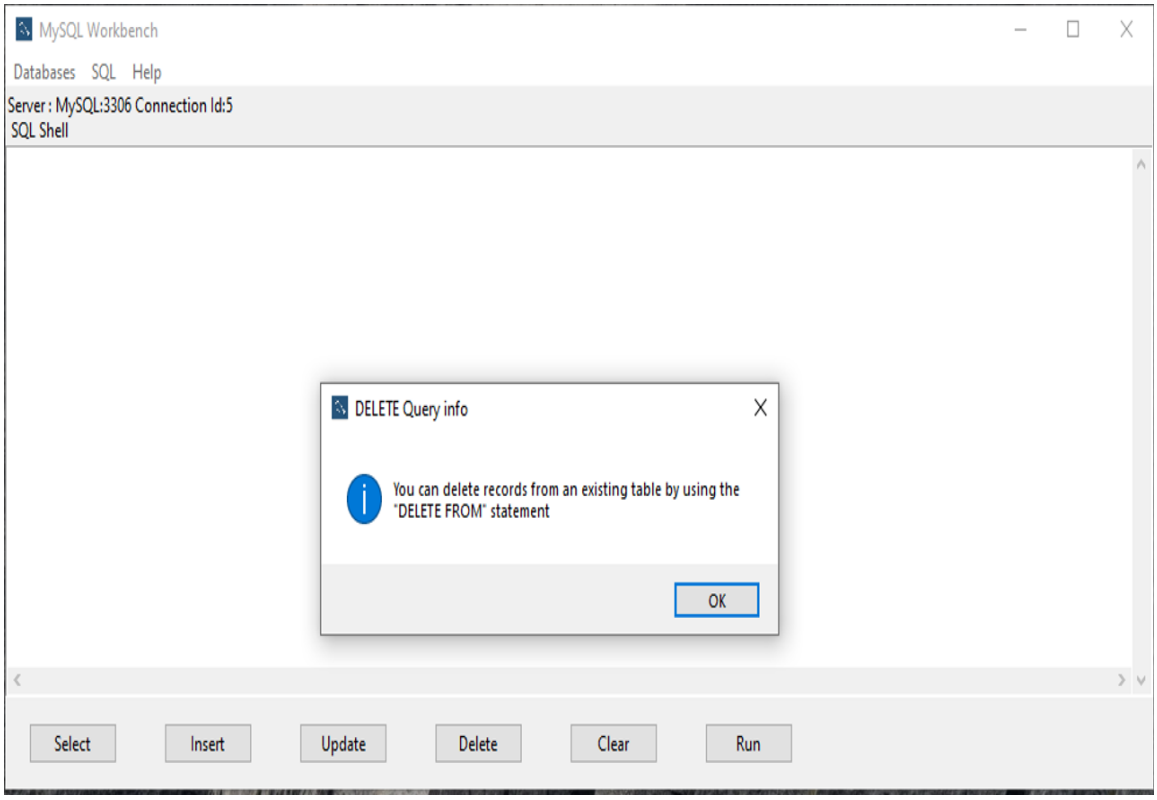

Figure 6.22: SQL Shell : Delete Query info

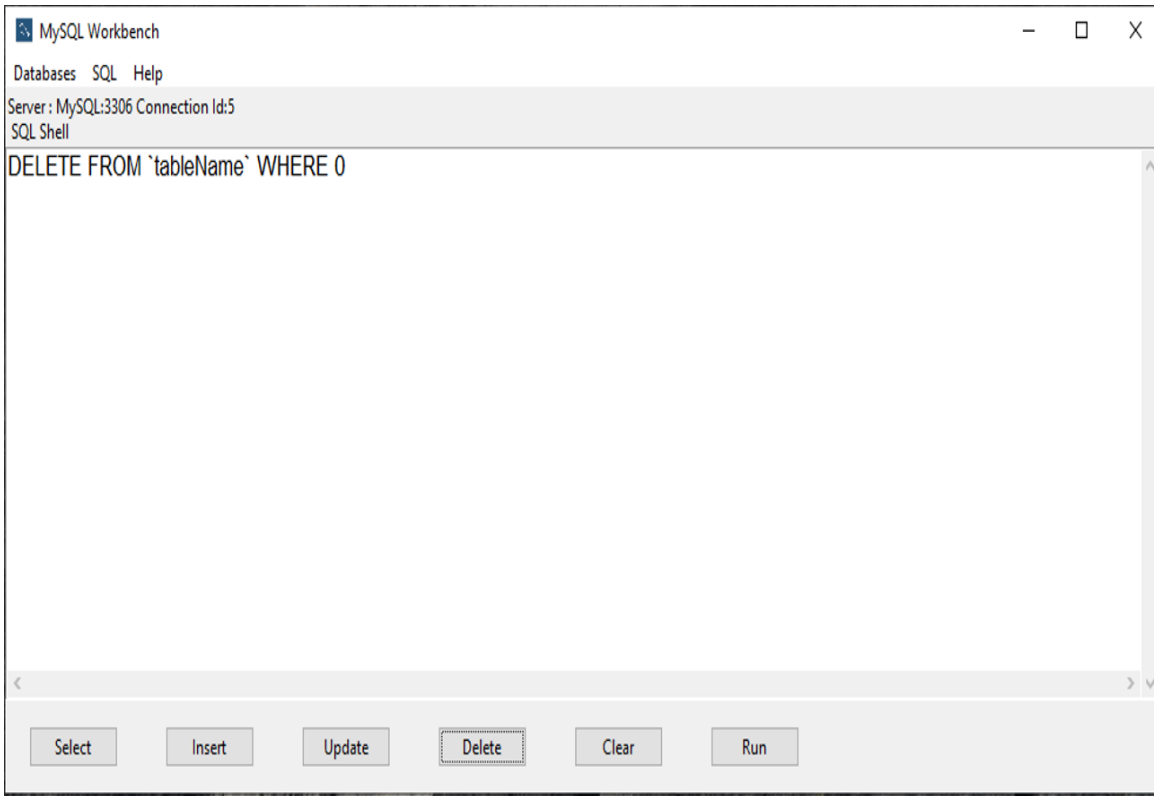

Figure 6.23: SQL Shell : Delete Query

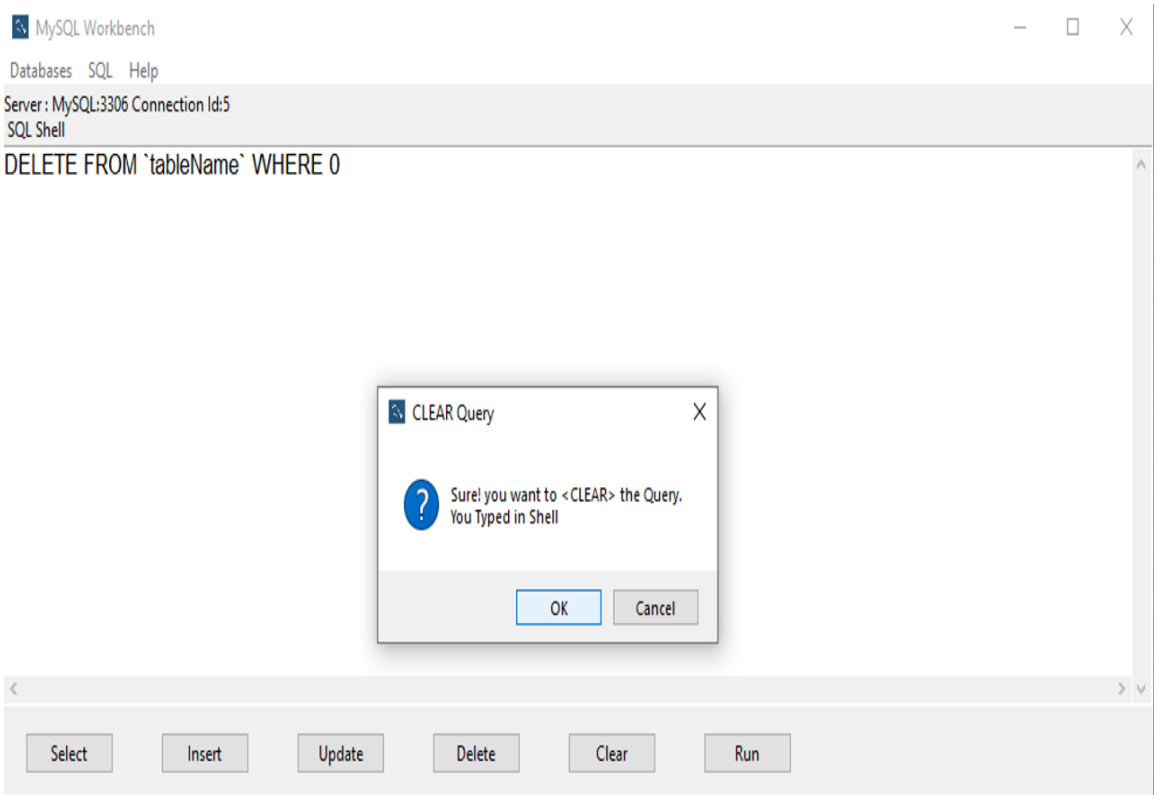

Figure 6.24: SQL Shell : Clear Query info

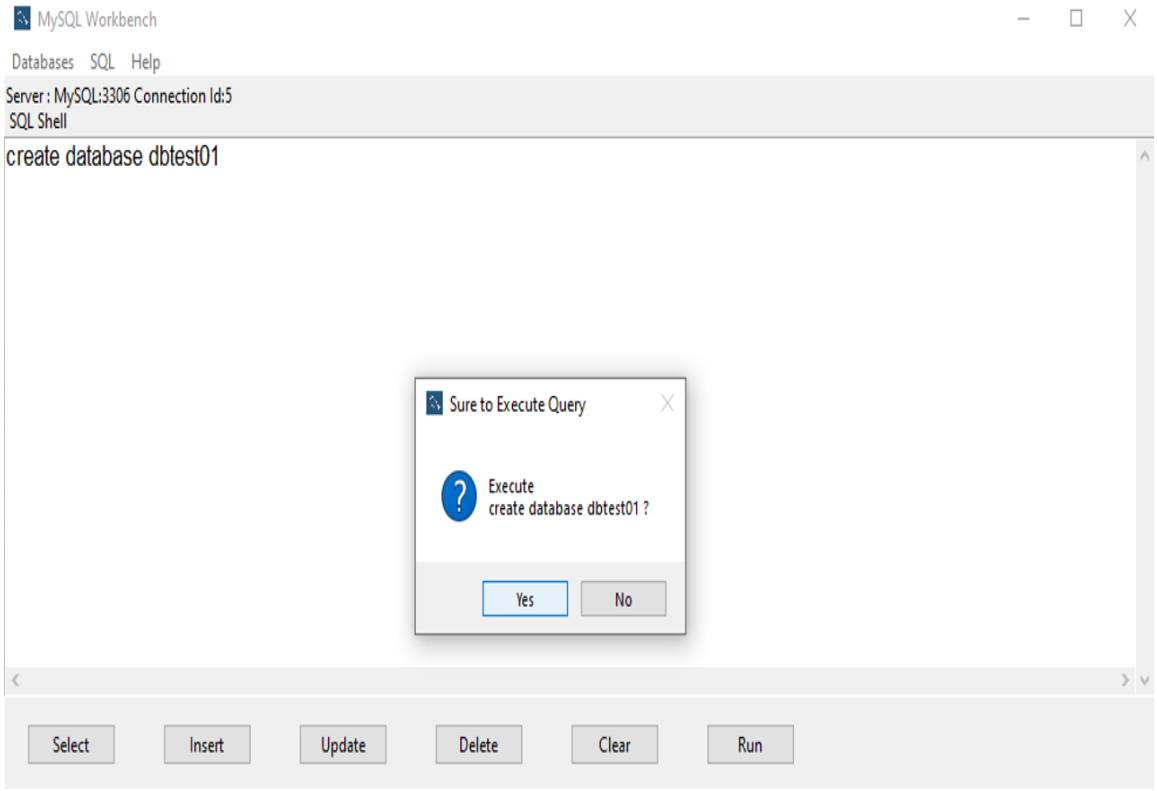

Figure 6.25: SQL Shell : Execute Query Confirmation

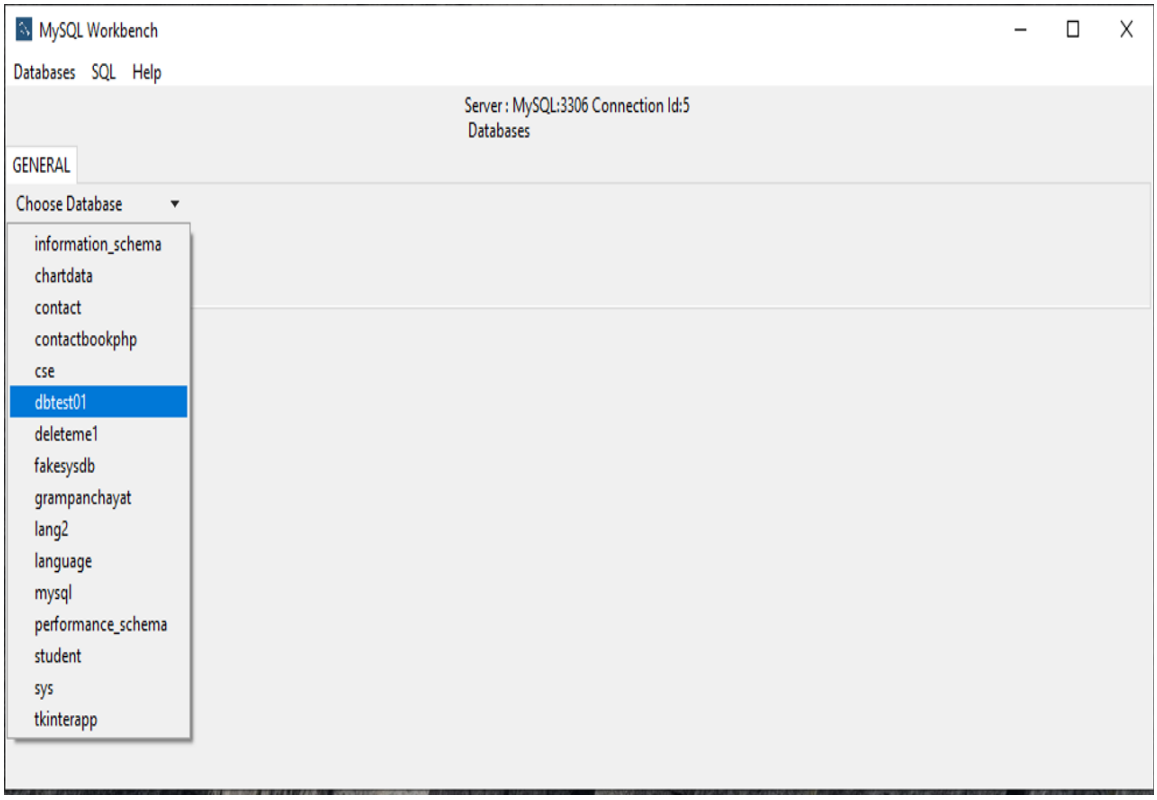

Figure 6.26: SQL Shell : Check database created

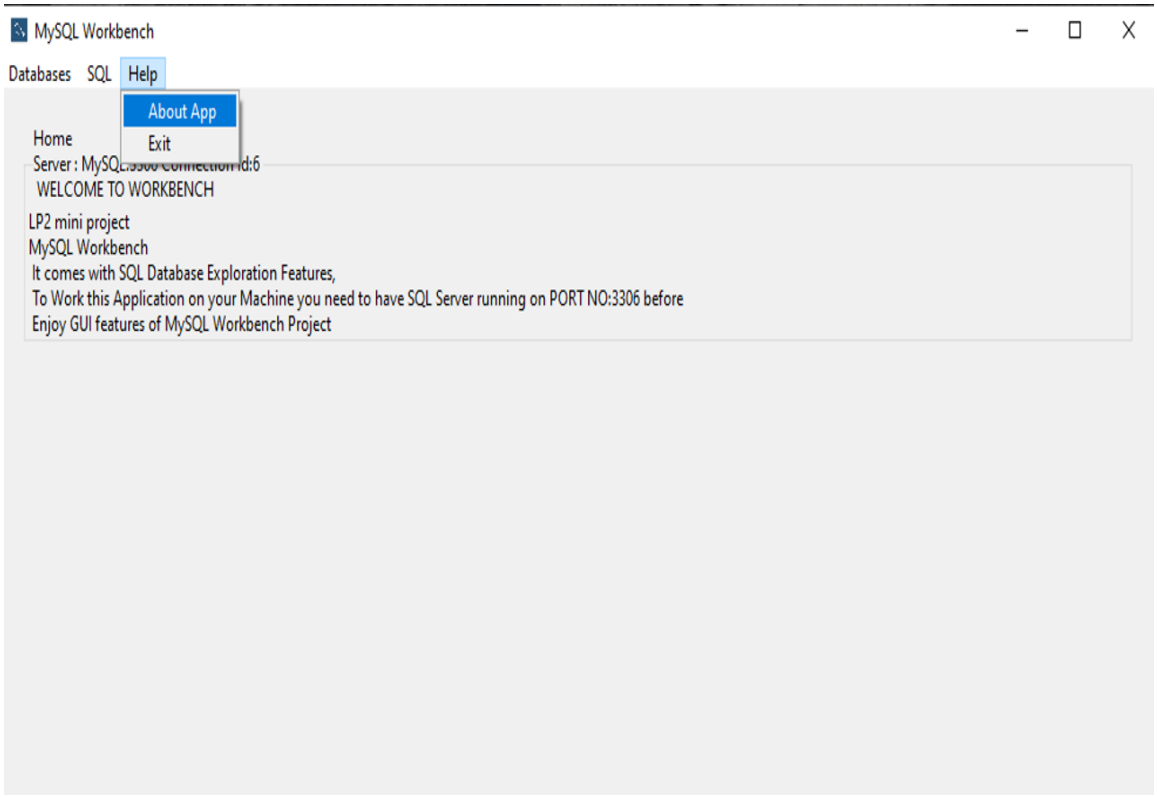

Figure 6.27: Help Tab : Application info

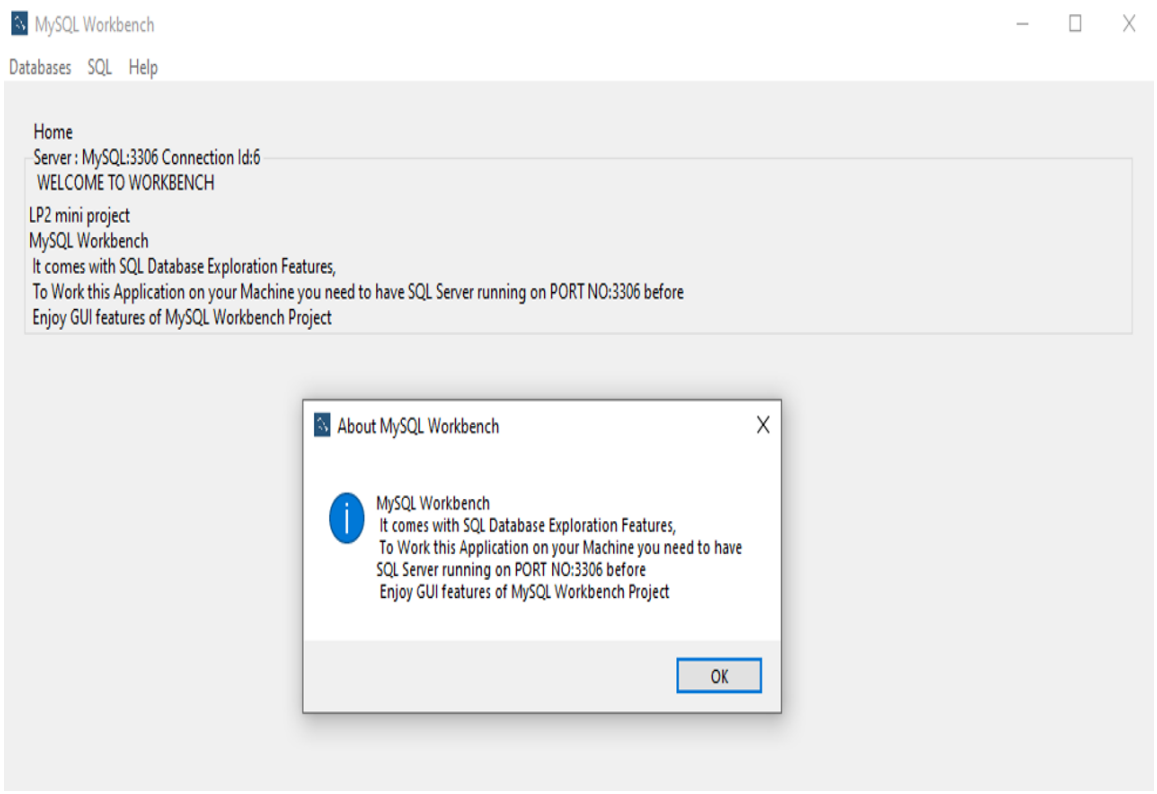

Figure 6.28: About Application

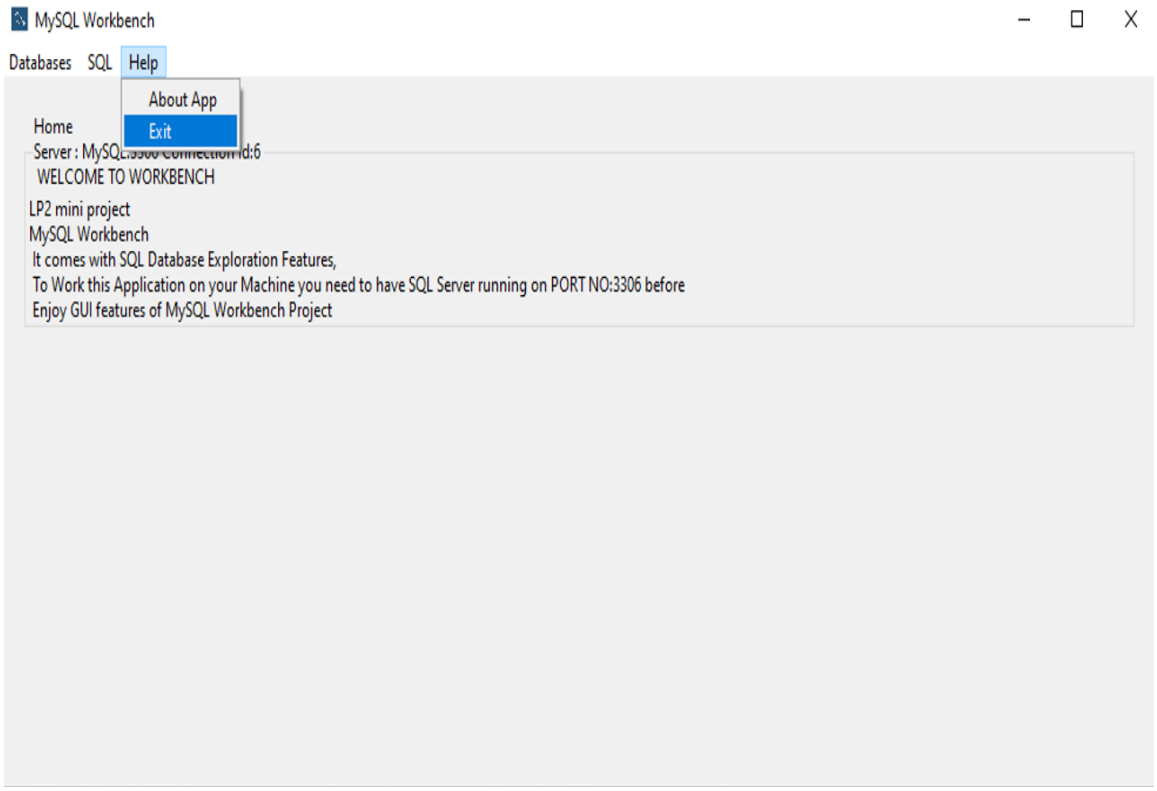

Figure 6.29: Help Tab : Exit

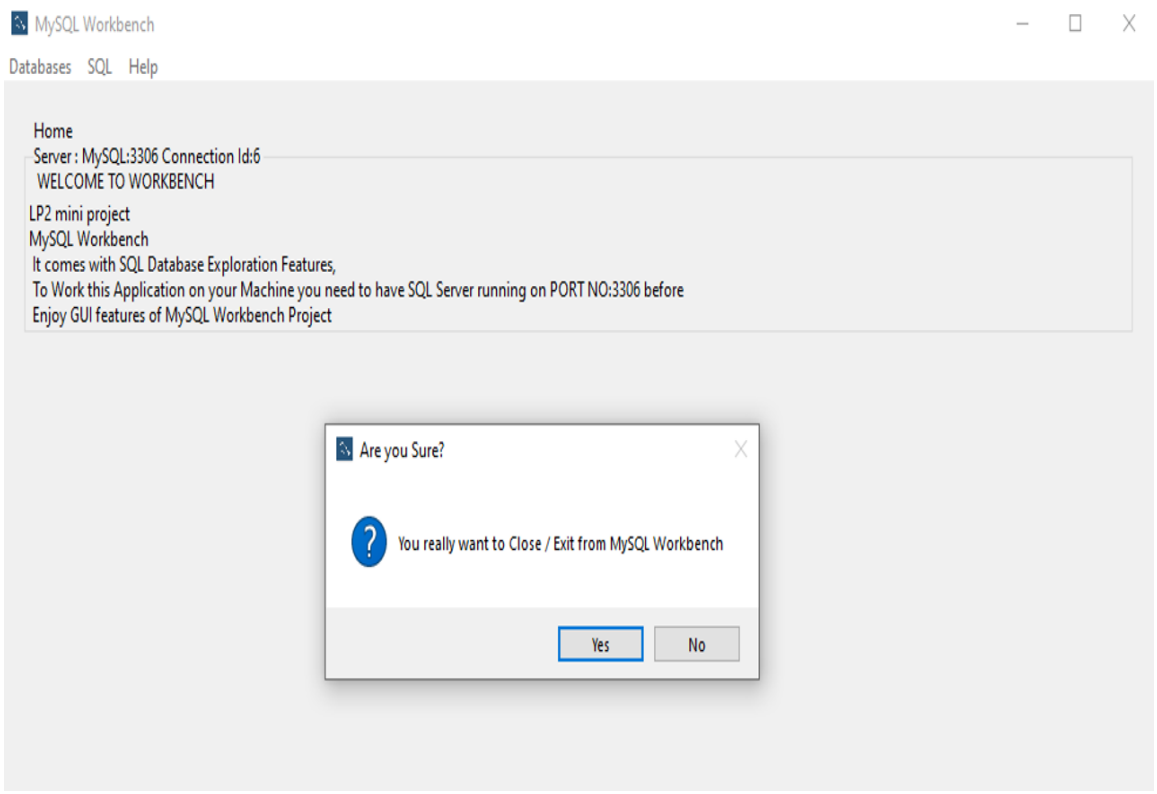

Figure 6.30: Help Tab : Application Exit Confirmation

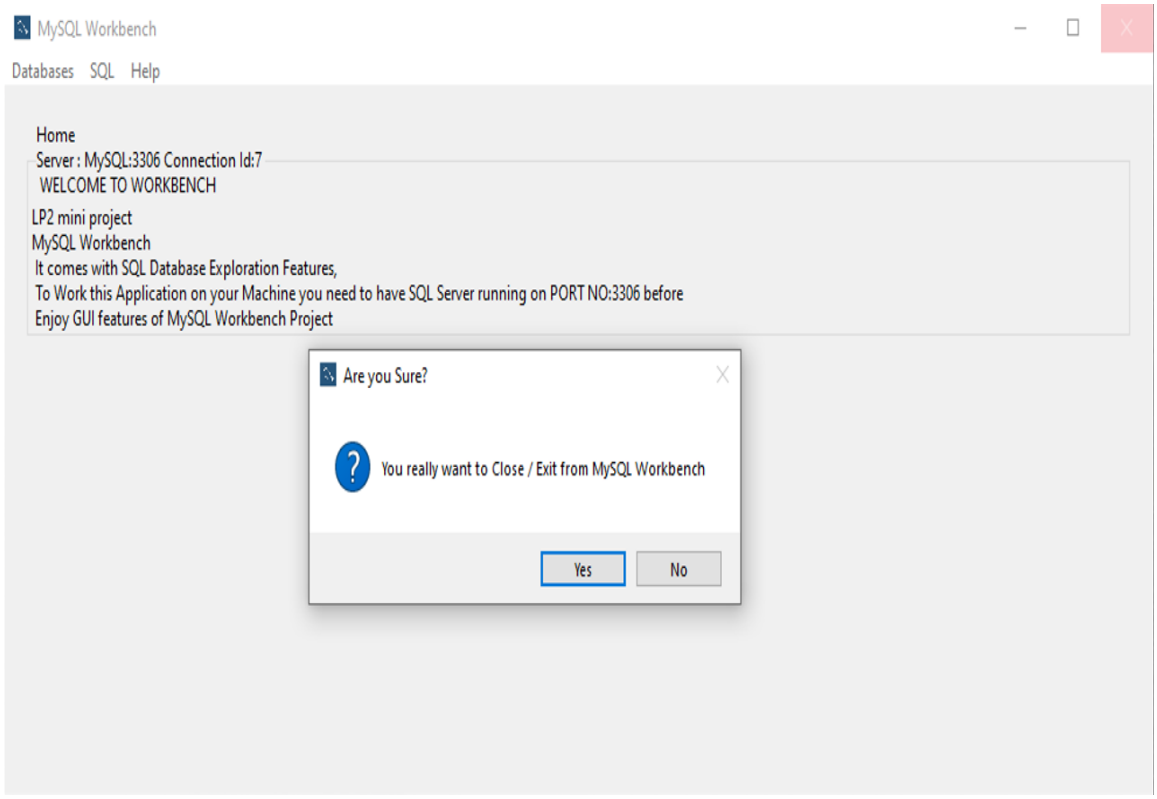

Figure 6.31: Window : Application Exit Confirmation

Chapter 7

PyUnit Test Cases Screenshots

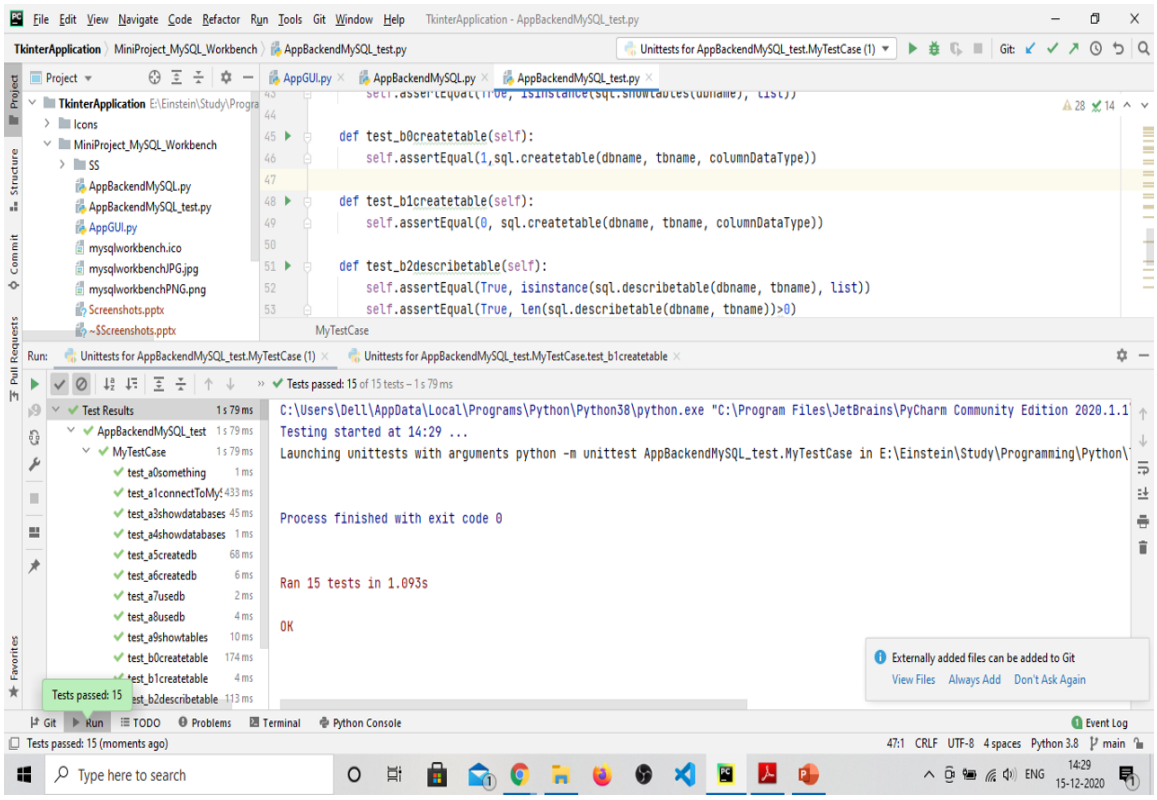

Figure 7.1: Running Test case

Chapter 8

Conclusion

In this way we have successfully completed the implementation of MySQL Workbench GUI Application . Also applied the PyUnit test cases to the application and the result is positive.