Module-7

28. Database.

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

MYSQL:

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company.

A relational database is a type of database that stores and provides access to data points that are related to one another. Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables.

Keys are very important part of Relational database model. They are used to establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table.

A Key can be a single attribute or a group of attributes, where the combination may act as a key.

Super Key is defined as a set of attributes within a table that can uniquely identify each record within a table. Super Key is a superset of Candidate key.

A super key could include student_id, (student_id, name), phone etc.

The first one is pretty simple as student_id is unique for every row of data, hence it can be used to identity each row uniquely.

Next comes, (student_id, name), now name of two students can be same, but their student_id can't be same hence this combination can also be a key.

Similarly, phone number for every student will be unique, hence again, phone can also be a key.

So they all are super keys.

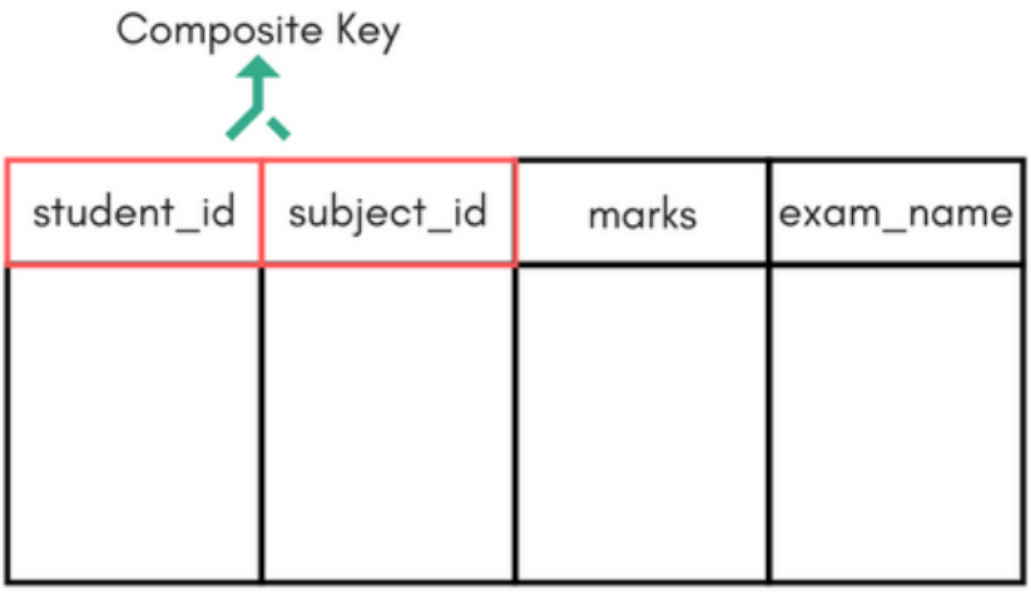
Candidate keys are defined as the minimal set of fields which can uniquely identify each record in a table. It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table. There can be more than one candidate key.

In our example, student_id and phone both are candidate keys for table Student.

- A candiate key can never be NULL or empty. And its value should be unique.
- There can be more than one candidate keys for a table.
- A candidate key can be a combination of more than one columns (attributes).

Primary key is a candidate key that is most appropriate to become the main key for any table. It is a key that can uniquely identify each record in a table.

Key that consists of two or more attributes that uniquely identify any record in a table is called **Composite key**. But the attributes which together form the Composite key are not a key independently or individually.



Score Table - To save scores of the student for various subjects.

In the above picture we have a Score table which stores the marks scored by a student in a particular subject.

In this table student_id and subject_id together will form the primary key, hence it is a composite key.

The candidate key which are not selected as primary key are known as **secondary keys** or **alternative keys**.

Workbench Overview:

MySQL Workbench is a designing or a graphical tool, which is used for working with MySQL servers and databases. This tool compatible with the older server 5.x versions and does not support the 4.x server versions.

The functionalities of MySQL Workbench are as follows:

• **SQL Development:** This functionality provides the capability to execute SQL queries, create and manage connections to database servers using the built-in SQL Editor.

• **Data Modeling (Design):** This functionality enables you to create models of your database schema graphically, perform reverse and forward engineer between a schema and a live database, and edit all aspects of your database using the comprehensive Table Editor.

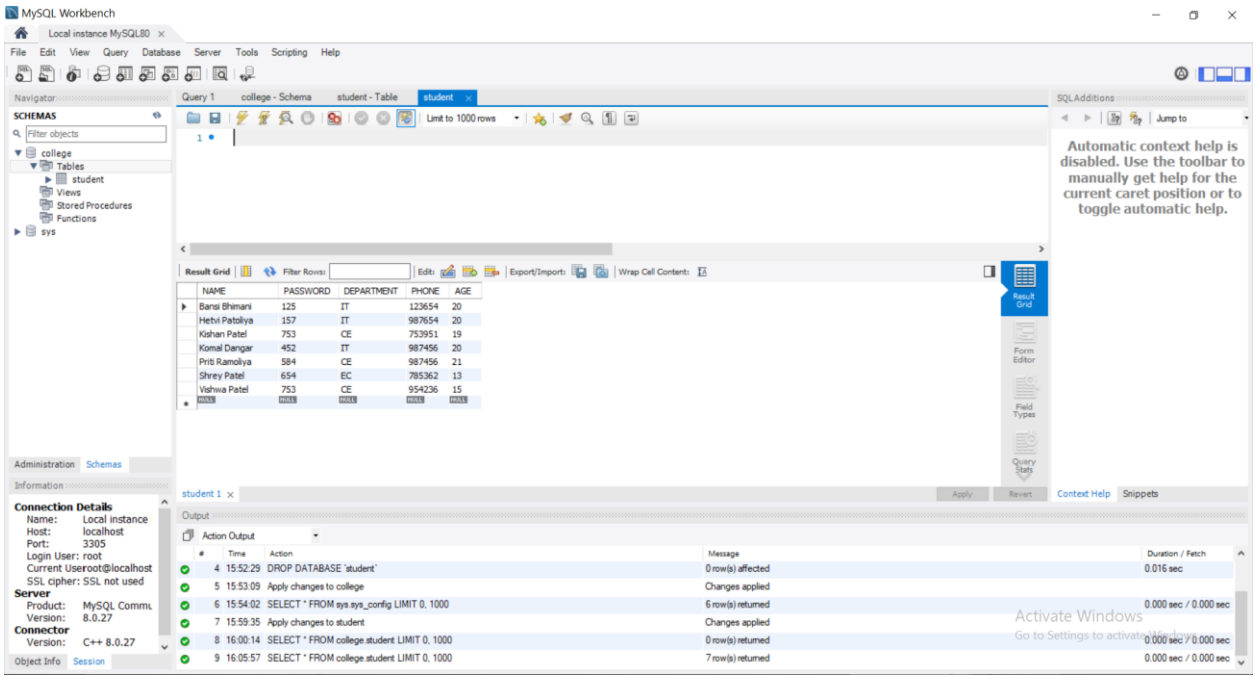
• **Server Administration:** This functionality enables you to administer MySQL server instances by administering users, performing backup and recovery, inspecting audit data, viewing database health, and monitoring the MySQL server performance.

• **Data Migration:** This functionality allows you to migrate from Microsoft SQL Server, Microsoft Access, and other RDBMS tables, objects, and data to MySQL.

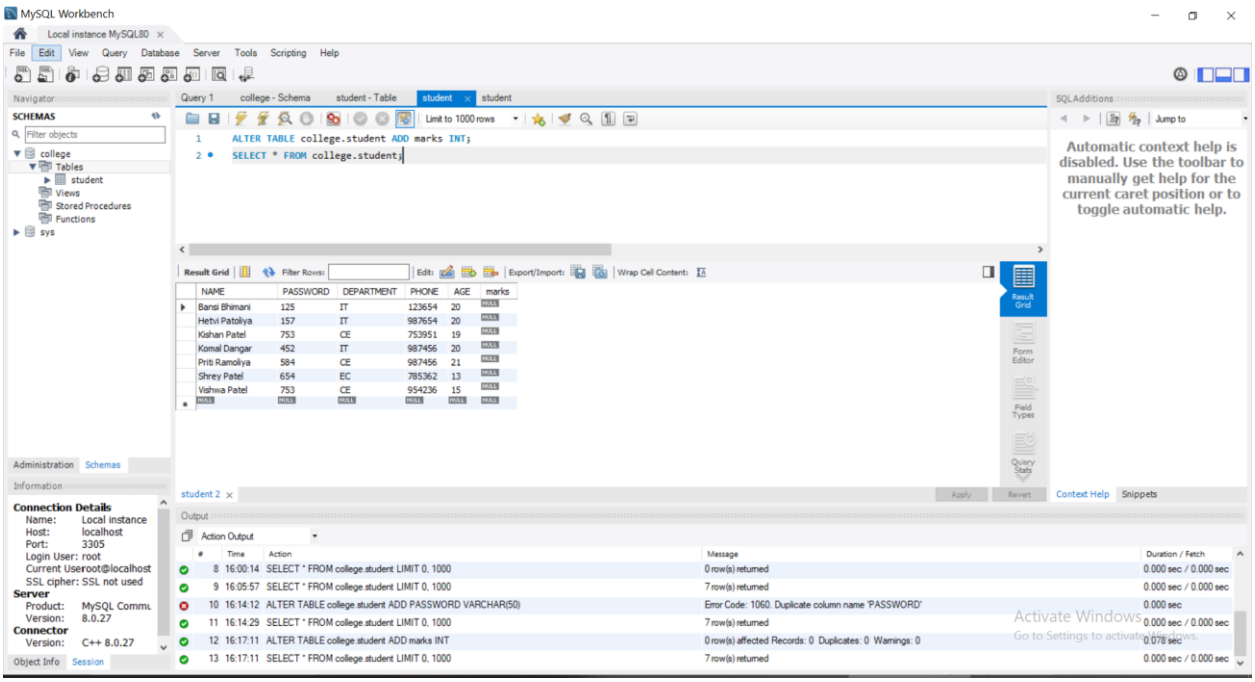
• **MySQL Enterprise Support:** This functionality provides support for Enterprise products such as MySQL Enterprise Backup, MySQL Firewall, and MySQL Audit.

In sql development I have fired various queries on student table of college database which is as shown below.

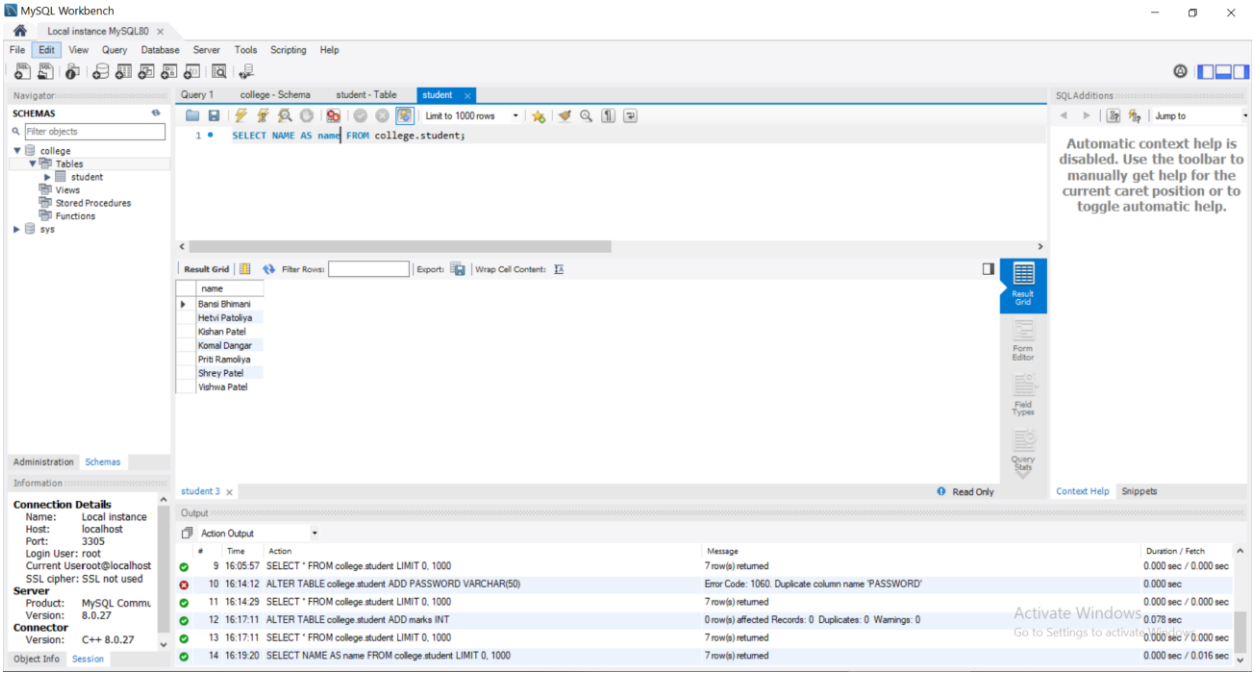
Table data:



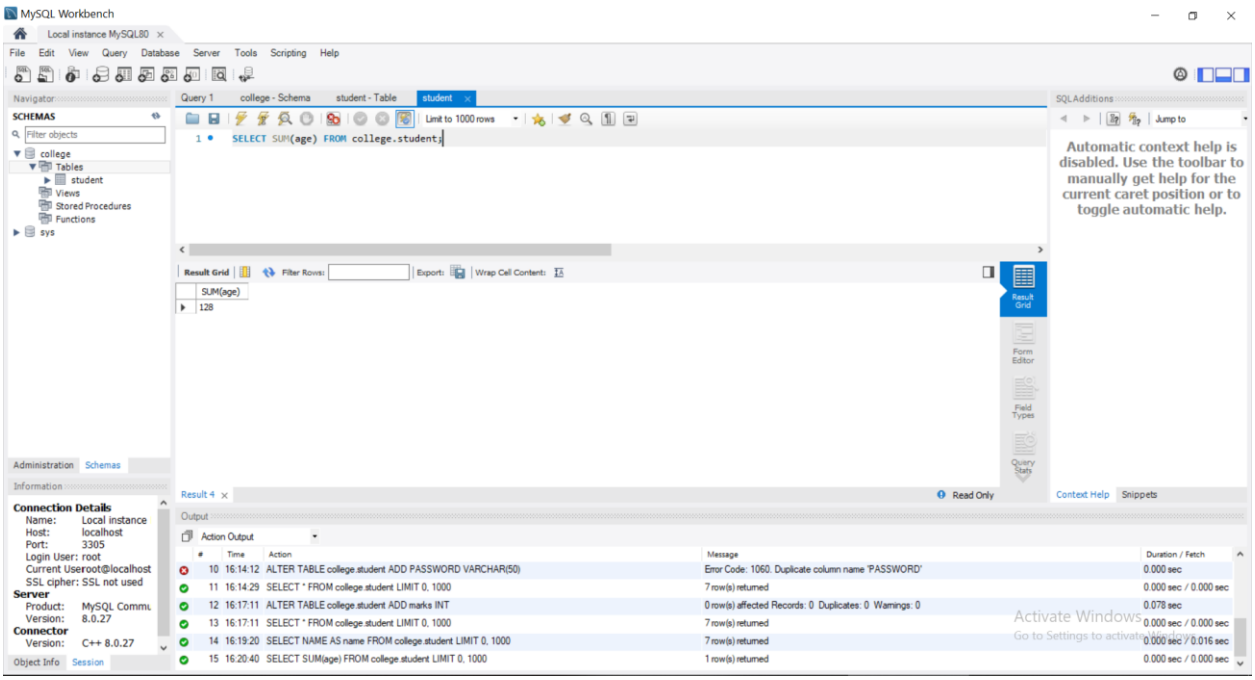
1. ALTER keyword



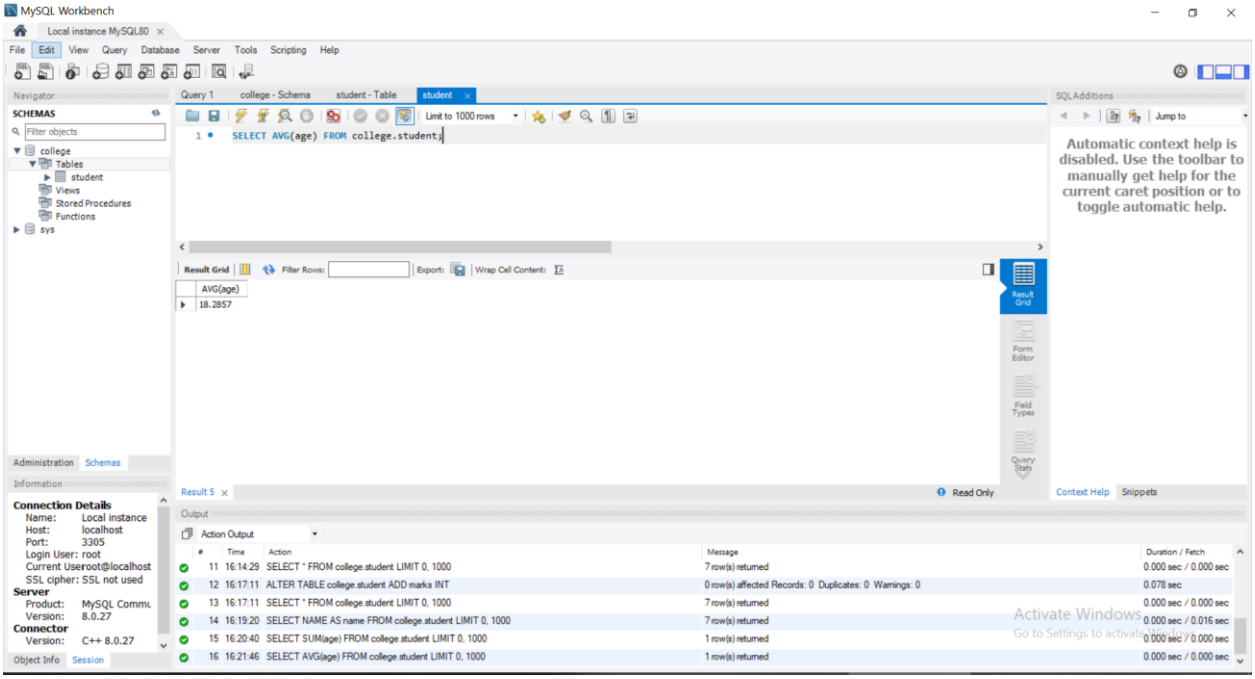
2. AS keyword



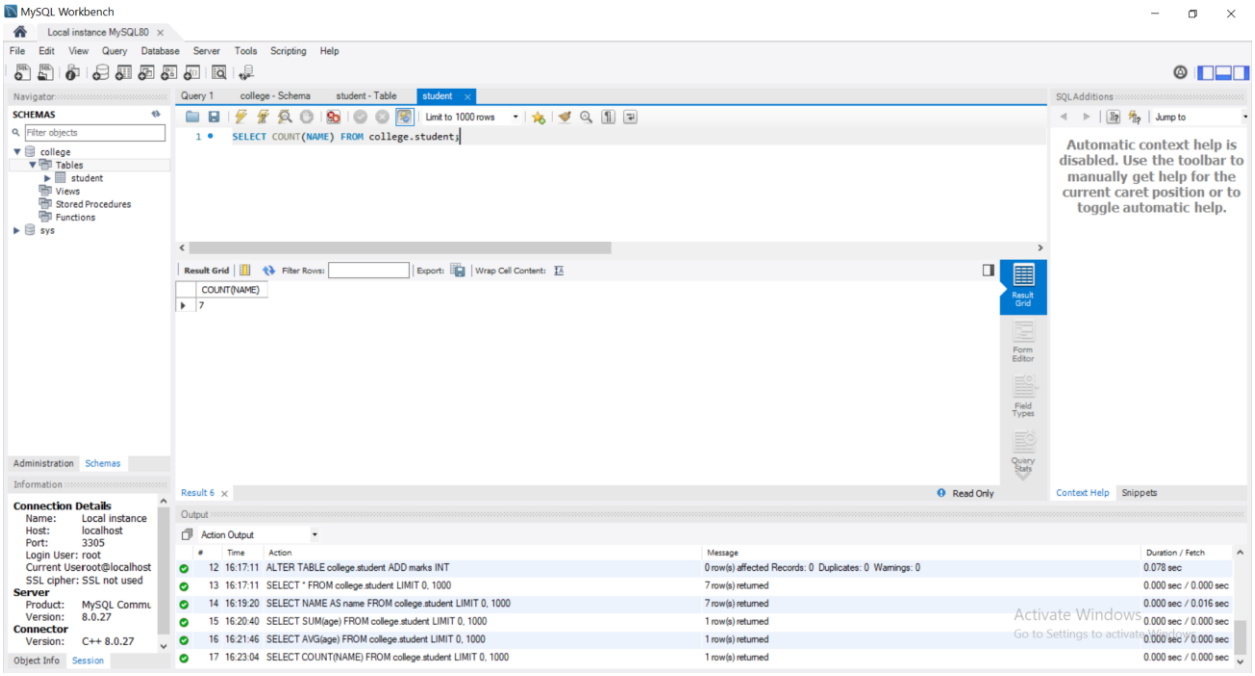
3. SUM keyword



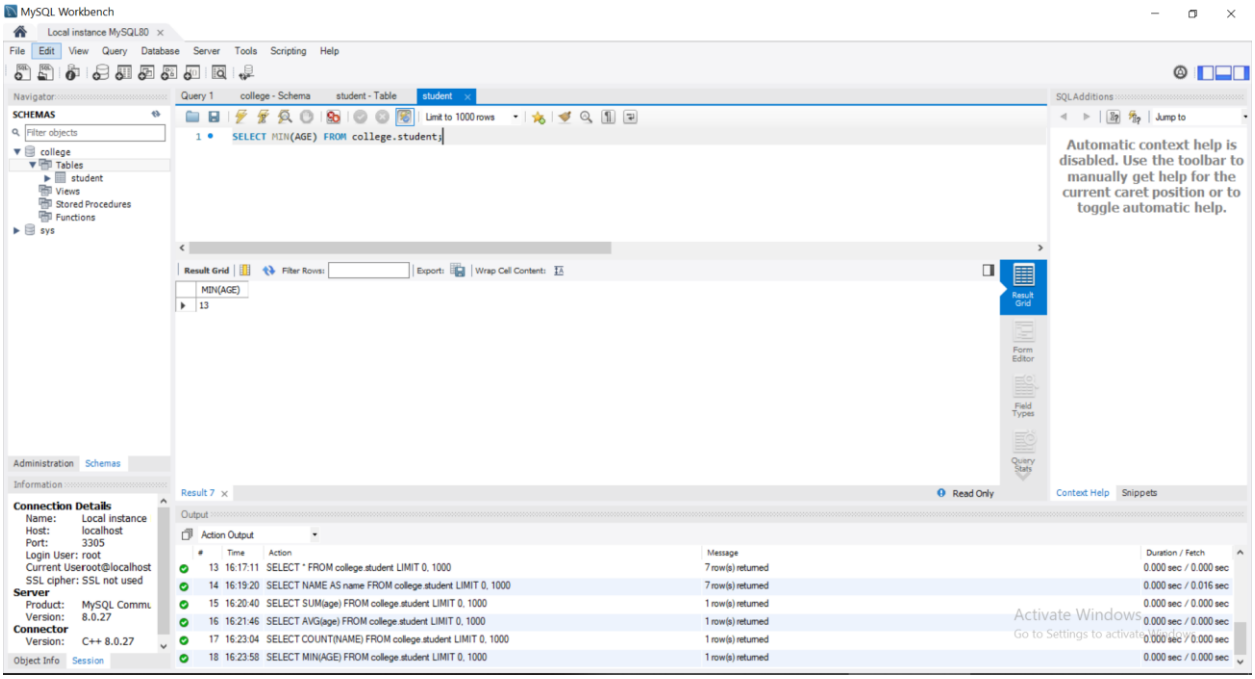
4. AVG keyword



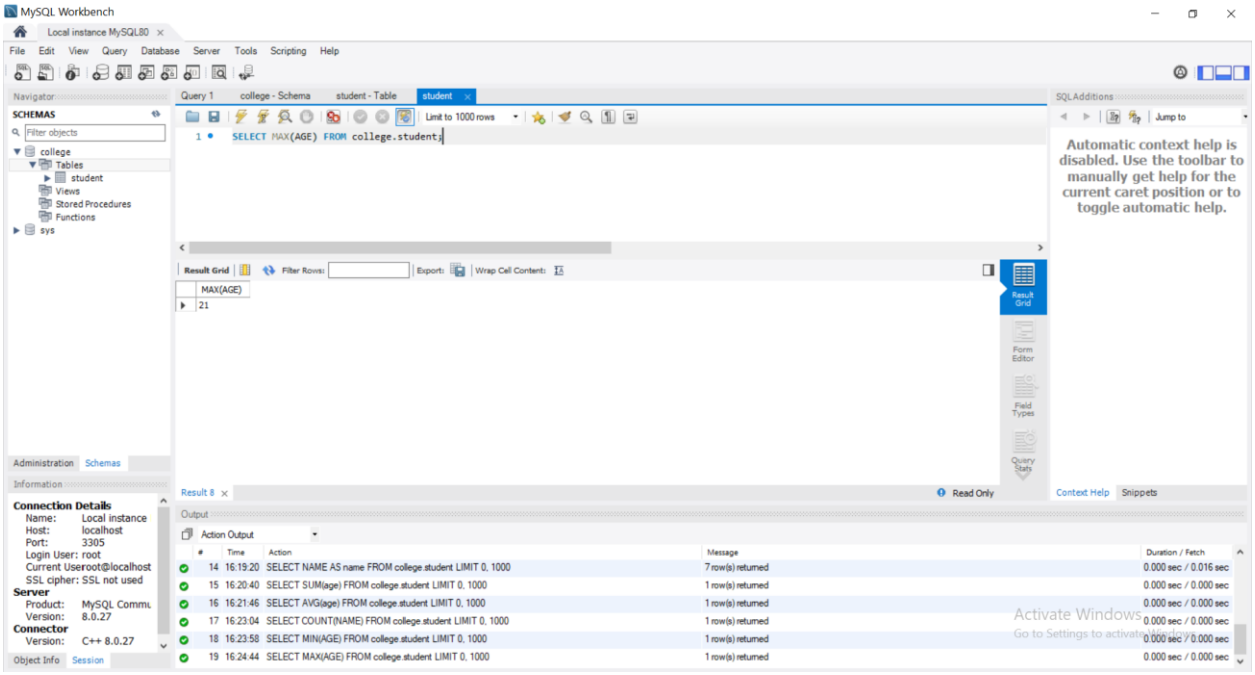
5. COUNT keyword



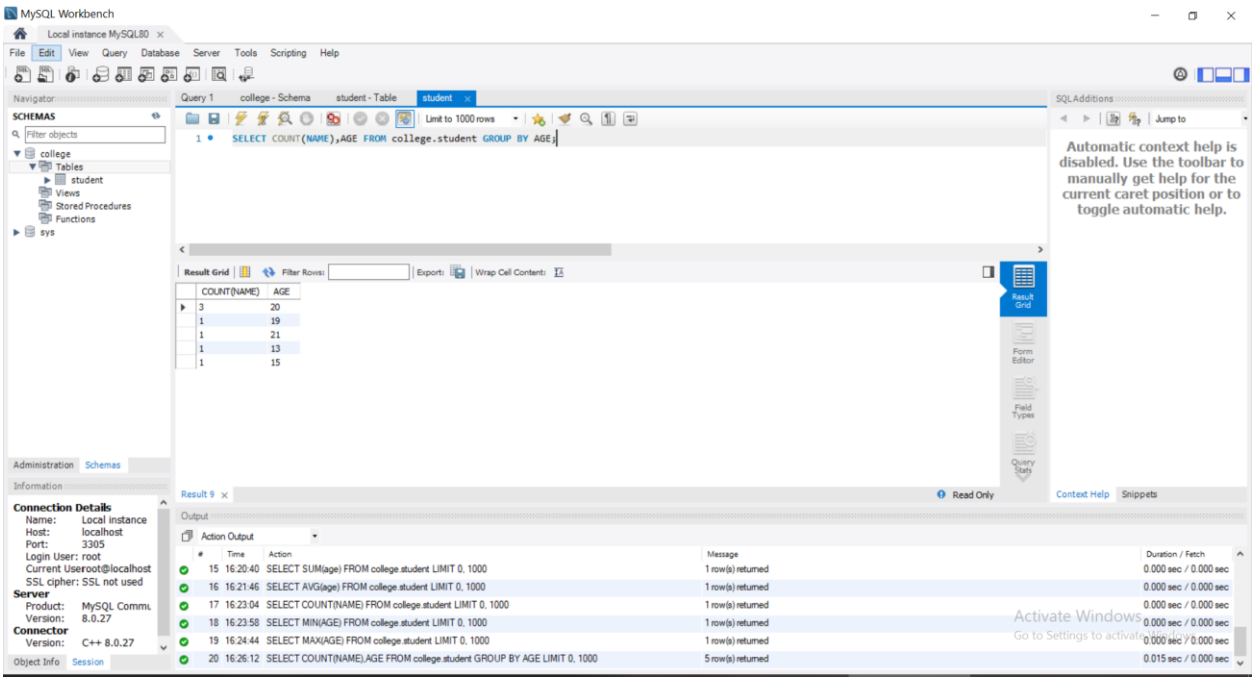
6. MIN keyword



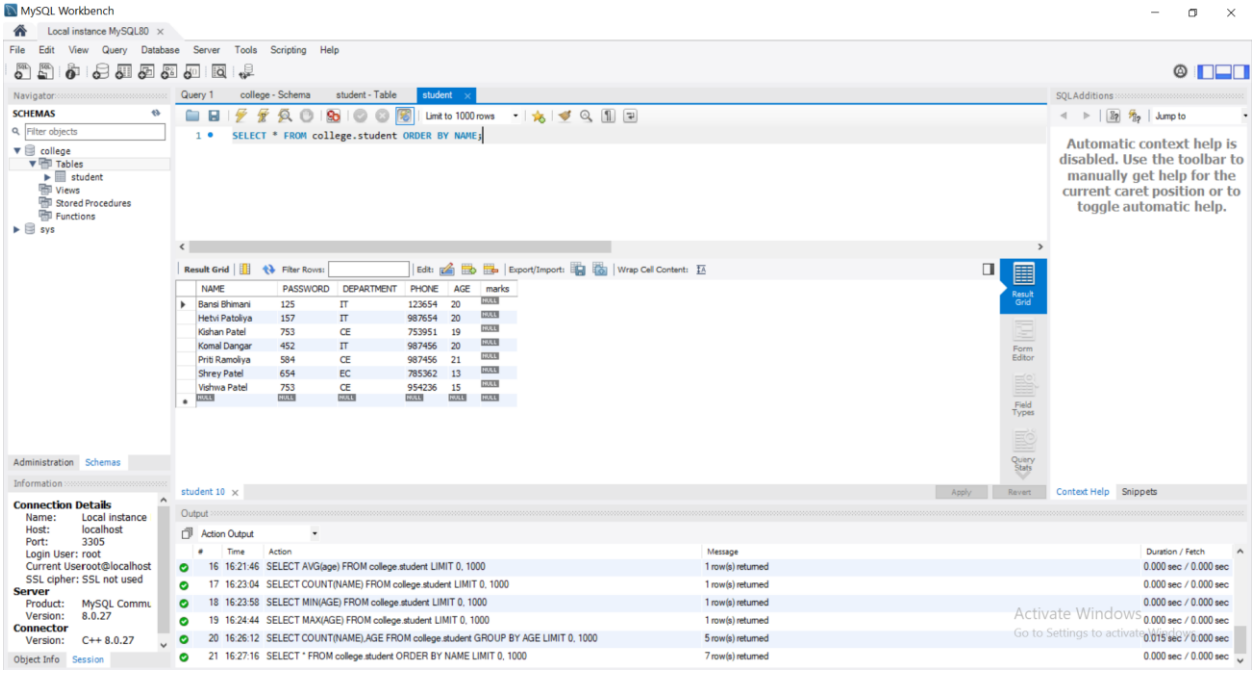
7. MAX keyword



8. GROUP BY keyword



9. ORDER BY keyword



Crude Operation:

I have made a very simple table in MYSQL Workbench. Then, I have connected MySql with web api and performed crud operations. Its demonstration is shown here.

Friend.cs:

```
using System.ComponentModel.DataAnnotations;

namespace CrudDemo.Models
{
   public class Friend
   {
      /// <summary>
      /// It is friend id
      /// </summary>
      [Required(ErrorMessage = "Friend id is required")]
      [Range(1, 100)]
```

```
        public int id { get; set; }
        /// <summary>
        /// It is firstname of friend
        /// </summary>
        public string firstname { get; set; }
        /// <summary>
        /// It is lastname of friend
        /// </summary>
        public string lastname { get; set; }
        /// <summary>
        /// It is location of friend
        /// </summary>
        public string location { get; set; }
        /// <summary>
        /// It is salary of friend
        /// </summary>
        [Required(ErrorMessage = "Salary is required")]
        public int salary { get; set; }

    }
}
```

BLfriend.cs:

```
using System;
using System.Collections.Generic;
using CrudDemo.Models;
using MySql.Data.MySqlClient;

namespace CrudDemo.BL_Class
{/// <summary>
/// This is BL class of Friend controller
/// </summary>
    public class BLfriend
    {
        public static string ConnectionString =
System.Configuration.ConfigurationManager.ConnectionStrings["sqlconnection"].ConnectionString;

        /// <summary>
        /// This is selectAll Method
        /// </summary>
        /// <returns>List<Friend></returns>
        public List<Friend> selectAll()
        {
            List<Friend> friends = new List<Friend>();

            using (MySqlConnection conn = new MySqlConnection(ConnectionString))
            {
                try
                {
                    //open connection
                    conn.Open();
                    MySqlCommand cmd = new MySqlCommand("select * from friend", conn);

                    //read data
                    using (var reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
```

```
                friends.Add(new Friend()
                {
                    id = Convert.ToInt32(reader["id"]),
                    firstname = reader["firstname"].ToString(),
                    lastname = reader["lastname"].ToString(),
                    location = reader["location"].ToString(),
                    salary = Convert.ToInt32(reader["salary"]),
                });
            }
        }
    }
    catch (Exception)
    {
        Console.WriteLine("Can not open connection !");

    }

  }
  return friends;
}

/// <summary>
/// getFriendById Method
/// </summary>
/// <param name="id">Id of friend</param>
/// <returns>Friend</returns>
public Friend getFriendById(int id)
{
  Friend friends = new Friend();

  using (MySqlConnection conn = new MySqlConnection(ConnectionString))
  {
    try
    {
      //open connection
      conn.Open();
      MySqlCommand cmd = new MySqlCommand("select * from friend where id = " + id + ";", conn);

      //read data
      using (var reader = cmd.ExecuteReader())
      {
        while (reader.Read())
        {
          friends.id = Convert.ToInt32(reader["id"]);
          friends.firstname = reader["firstname"].ToString();
          friends.lastname = reader["lastname"].ToString();
          friends.location = reader["location"].ToString();
          friends.salary = Convert.ToInt32(reader["salary"]);
        }
      }
    }
    catch (Exception)
    {
      Console.WriteLine("Can not open connection !");

    }
    finally
    {
      conn.Close();
```

```
            }

        }
        return friends;
    }

    /// <summary>
    /// addFriend Method
    /// </summary>
    /// <returns>string</returns>
    public string addFriend(Friend objFriend)
    {
        List<Friend> friends = new List<Friend>();

        using (MySqlConnection conn = new MySqlConnection(ConnectionString))
        {
            try
            {
                conn.Open();

                MySqlCommand cmd = new MySqlCommand("insert into friend (id,firstname,lastname,location,salary) values('"
+ objFriend.id + "','" + objFriend.firstname + "','" + objFriend.lastname + "','" + objFriend.location + "','" + objFriend.salary +
"');", conn);


                int effect = cmd.ExecuteNonQuery();
                if (effect > 0)
                {
                    return "successfull";
                }
                return "Not successfull query";


            }
            catch (Exception ex)
            {
                return "Cannot open connection with error - " + ex.Message;
            }
            finally
            {
                conn.Close();
            }

        }
    }

    /// <summary>
    /// deleteFriend Method
    /// </summary>
    /// <param name="id">id of friend</param>
    /// <returns>string</returns>
    public string deleteFriend(int id)
    {
        using (MySqlConnection conn = new MySqlConnection(ConnectionString))
        {
            try
            {
                conn.Open();
```

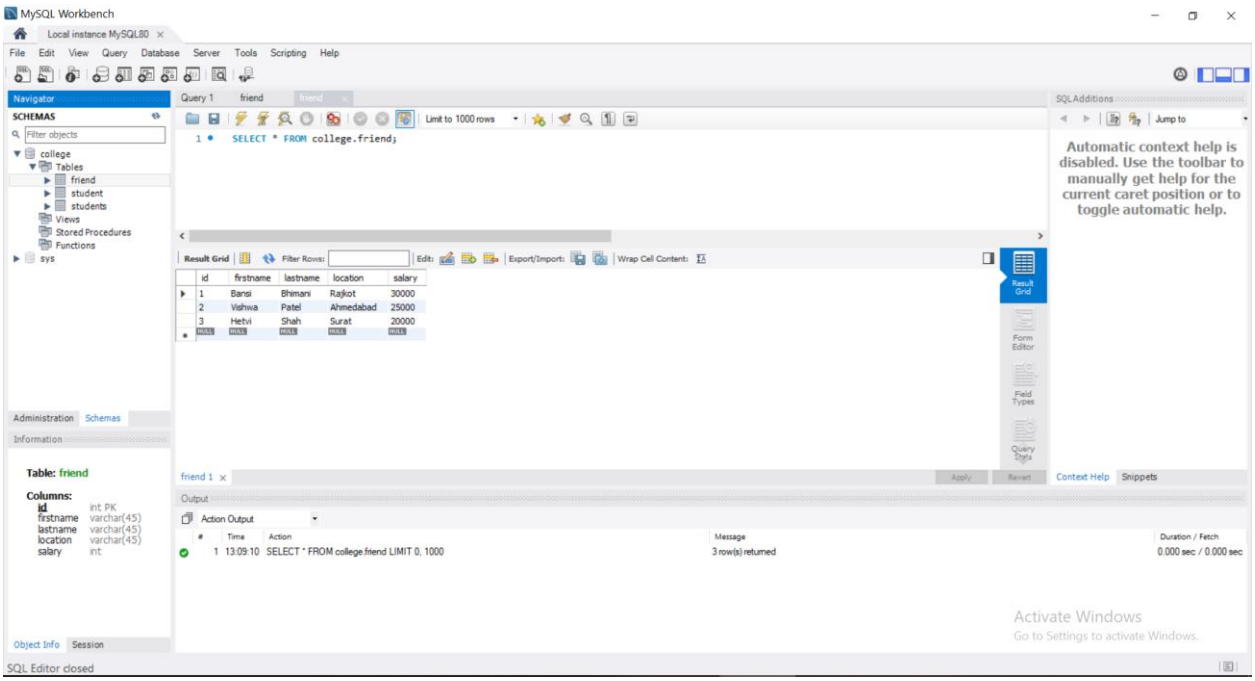
```
            MySqlCommand cmd = new MySqlCommand("delete from friend where id = " + id, conn);

            int effect = cmd.ExecuteNonQuery();
            if (effect > 0)
            {
               return "successfull";
            }
            return "Not successfull query";


         }
         catch (Exception ex)
         {
            return "Cannot open connection with error - " + ex.Message;
         }
         finally
         {
            conn.Close();
         }

      }
   }

   //update data
   public string updateFriend(Friend objFriend)
   {
      List<Friend> friends = new List<Friend>();

      using (MySqlConnection conn = new MySqlConnection(ConnectionString))
      {
         try
         {
            conn.Open();

            MySqlCommand cmd = new MySqlCommand("update countries set firstname = '" + objFriend.firstname +
"',lastname = '" + objFriend.lastname + "',location = '" + objFriend.location + "',salary = '" + objFriend.salary + "';", conn);


            int effect = cmd.ExecuteNonQuery();
            if (effect > 0)
            {
               return "successfull";
            }
            return "Not successfull query";


         }
         catch (Exception ex)
         {
            return "Cannot open connection with error - " + ex.Message;
         }
         finally
         {
            conn.Close();
         }

      }
   }
}
```

```
}
```

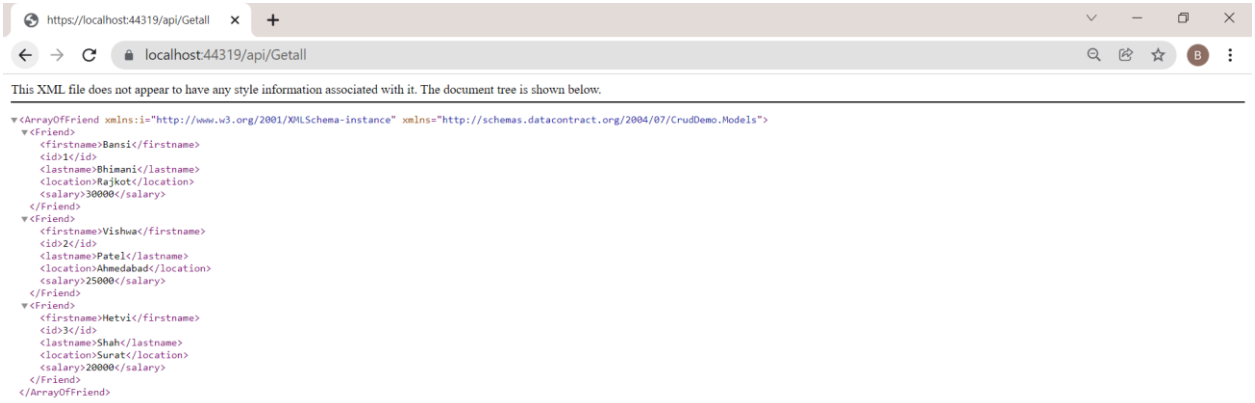
FriendController.cs:

```
using System.Collections.Generic;
using System.Web.Http;
using CrudDemo.Models;
using CrudDemo.BL_Class;

namespace CrudDemo.Controllers
{
   /// <summary>
   /// This is controller class of Friend
   /// </summary>
   public class FriendController : ApiController
   {
      //Object of BL class
      BLfriend objBLfriend = new BLfriend();

      /// <summary>
      /// GetAll method
      /// </summary>
      /// <returns>List<Friend></returns>

      // GET all friends
      [Route("api/Getall")]
      public List<Friend> GetAll()
      {
         return (objBLfriend.selectAll());
      }

      /// <summary>
      /// Get Method
      /// </summary>
      /// <param name="id">This is id of friend</param>
      /// <returns>Friend</returns>
      // GET friends by id
      [Route("api/Get/{id}")]
      public Friend Get(int id)
      {
         return (objBLfriend.getFriendById(id));
      }

      /// <summary>
      /// Add Method
      /// </summary>
      // Add friend
      [HttpPost]
      [Route("api/Add")]
      public void Add([FromBody] Friend objFriend)
      {
         objBLfriend.addFriend(objFriend);
      }

      /// <summary>
      /// Update Method
      /// </summary>
     //update friend data
      [HttpPost]
```

```
        [Route("api/Update")]
        public void Update([FromBody] Friend objFriend)
        {
            objBLfriend.updateFriend(objFriend);
        }

        /// <summary>
        /// Delete Method
        /// </summary>
        /// <param name="id"></param>
        // DELETE friend by id
        [Route("api/Delete/{id}")]
        public void Delete(int id)
        {
            objBLfriend.deleteFriend(id);
        }

}
}
```

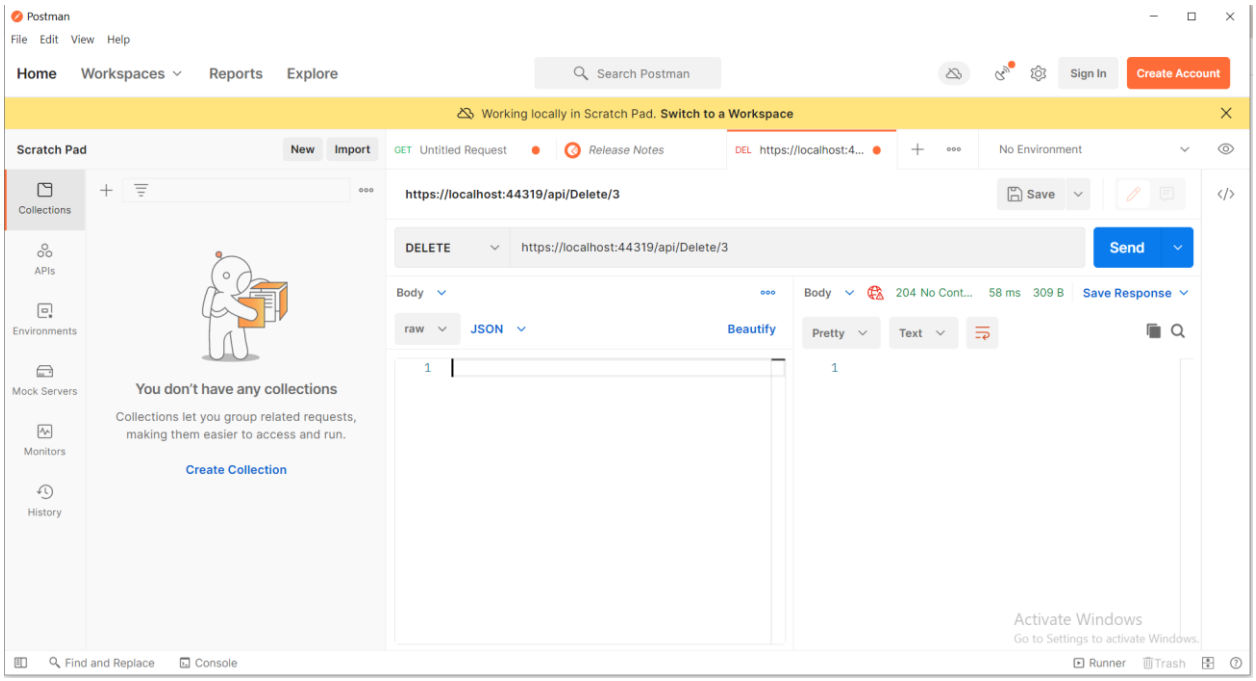


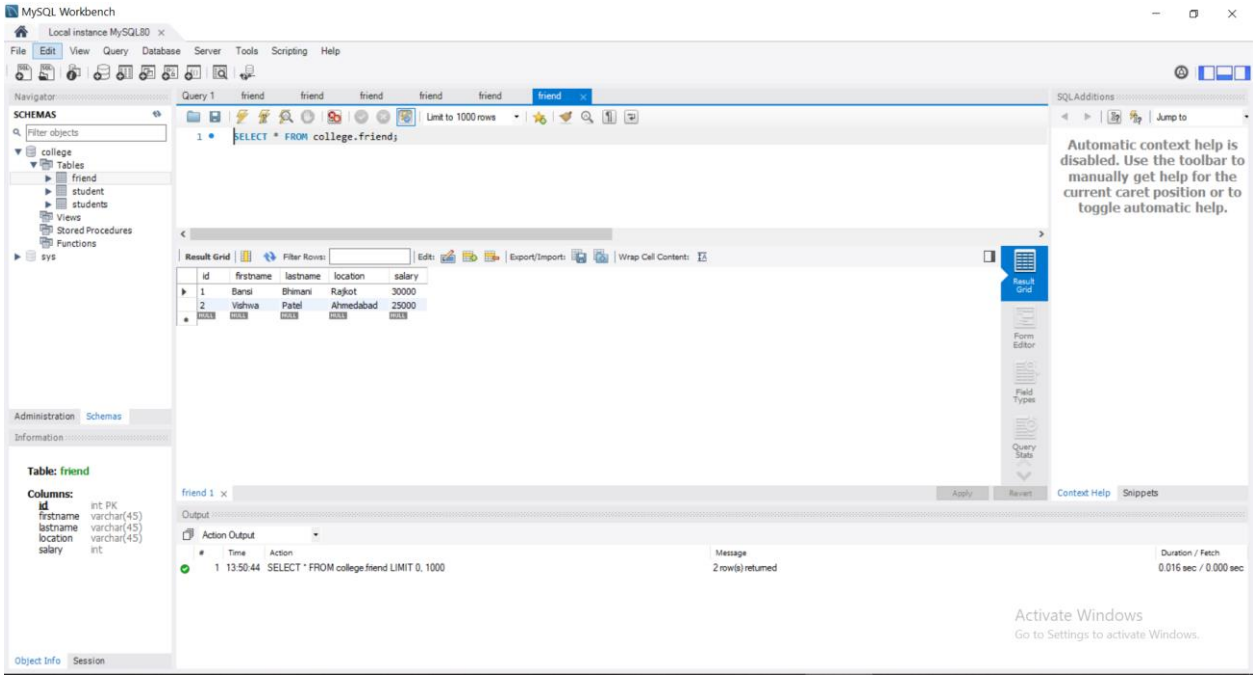
Get:




https://localhost:44319/api/Get/2
localhost:44319/api/Get/2
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<Friend xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/CrudDemo.Models">
<firstname>Vishwa</firstname>
<id>2</id>
<lastname>Patel</lastname>
<location>Ahmedabad</location>
<salary>25000</salary>
</Friend>

Delete:


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
https://localhost:44319/api/Delete/3
DELETE
https://localhost:44319/api/Delete/3
Send
Body
raw JSON
Beautify
204 No Cont... 58 ms 309 B
Save Response
Pretty Text
Activate Windows
Go to Settings to activate Windows.
Find and Replace
Console
Runner
Trash


MySQL Workbench
SCHEMAS
college
Tables
friend
student
students
Views
Stored Procedures
Functions
sys
SELECT * FROM college.friend;
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
id firstname lastname location salary
1 Bansi Bhimani Rajkot 30000
2 Vishwa Patel Ahmedabad 25000
Table: friend
Columns:
id int PK
firstname varchar(45)
lastname varchar(45)
location varchar(45)
salary int
Action Output
1 13:50:44 SELECT * FROM college.friend LIMIT 0, 1000
2 row(s) returned
0.016 sec / 0.000 sec
Activate Windows
Go to Settings to activate Windows.