- Source code

```
CREATE DATABASE StudentDb;
go
USE StudentDb;

CREATE  TABLE StudentMarks (
    StudentID INT PRIMARY KEY ,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Subject VARCHAR(50) NOT NULL,
    Marks INT NOT NULL
)
INSERT INTO StudentMarks (StudentID,FirstName, LastName, Subject, Marks) VALUES
(1,'Rathan', 'Kumar', 'Math', 70) ,
(2,'vikram', 'kohli', 'social', 85) ,
(3,'kishan', 'Kumar', 'Biology', 98) ,
(4,'nani', 'rao', 'Hindi', 92),
(5,'Nithn', 'Sree', 'Telugu', 68)


SELECT AVG(Marks)AS AverageMarks,
MAX(Marks)  AS MaximumMarks ,
MIN(Marks)  AS MinimumMarks
FROM StudentMarks

   SELECT * FROM StudentMarks;
```

- Code in Visual Studio

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using System.Web.Http.Description;
using Addstudentmarks.Models;

namespace Addstudentmarks.Controllers
{
    public class StudentMarksController : ApiController
    {
        private StudentDbEntities db = new StudentDbEntities();

        // GET: api/StudentMarks
        public IQueryable<StudentMark> GetStudentMarks()
        {
            return db.StudentMarks;
        }

        // GET: api/StudentMarks/5
        [ResponseType(typeof(StudentMark))]
        public IHttpActionResult GetStudentMark(int id)
        {
            StudentMark studentMark = db.StudentMarks.Find(id);
            if (studentMark == null)
            {
                return NotFound();
            }

            return Ok(studentMark);
        }

        // PUT: api/StudentMarks/5
        [ResponseType(typeof(void))]
        public IHttpActionResult PutStudentMark(int id, StudentMark studentMark)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            if (id != studentMark.StudentID)
            {
                return BadRequest();
            }

            db.Entry(studentMark).State = EntityState.Modified;

            try
            {
                db.SaveChanges();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!StudentMarkExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return StatusCode(HttpStatusCode.NoContent);
        }

        // POST: api/StudentMarks
        [ResponseType(typeof(StudentMark))]
        public IHttpActionResult PostStudentMark(StudentMark studentMark)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            db.StudentMarks.Add(studentMark);

            try
            {
                db.SaveChanges();
            }
            catch (DbUpdateException)
            {
                if (StudentMarkExists(studentMark.StudentID))
                {
                    return Conflict();
                }
                else
                {
                    throw;
                }
            }

            return CreatedAtRoute("DefaultApi", new { id = studentMark.StudentID
}, studentMark);
        }

        // DELETE: api/StudentMarks/5
        [ResponseType(typeof(StudentMark))]
        public IHttpActionResult DeleteStudentMark(int id)
        {
            StudentMark studentMark = db.StudentMarks.Find(id);
            if (studentMark == null)
            {
                return NotFound();
            }

            db.StudentMarks.Remove(studentMark);
            db.SaveChanges();

            return Ok(studentMark);
        }

        protected override void Dispose(bool disposing)
```

```
    {
        if (disposing)
        {
            db.Dispose();
        }
        base.Dispose(disposing);
    }

    private bool StudentMarkExists(int id)
    {
        return db.StudentMarks.Count(e => e.StudentID == id) > 0;
    }
}
  }
```