Department Model Class

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Feb12_MainProject1.Model
{
    [Table("DeptMaster")]
    public class DeptMaster
    {
        [Key]
        public int DeptCode { get; set; }

        public string DeptName { get; set; }
        public string PropertyName { get; set; }
        public virtual ICollection<Employee> Employee { get; set; }

    }
}
```

Employee Model Class

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Mainprojec1.Model
{
    [Table("Employee")]
    public class Employee
    {
        [Key]
        public int EmpCode { get; set; }
        public DateTime DateOfBirth { get; set; }
        //[StringLength(100)]
        public string EmpName { get; set; }
        public string Email { get; set; }
        public int DeptCode { get; set; }
        public virtualDeptMaster DeptMaster{ get; set; }

    }
}
```

Employees Controller Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Mainprojec1.Data;
using Mainprojec1.Model;

namespace Mainprojec1.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class EmployeesController : ControllerBase
    {
        private readonly EmployeeDbContext _context;

        public EmployeesController(EmployeeDbContext context)
        {
            _context = context;
        }

        // GET: api/Employees
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Employee>>> GetEmployee()
        {
          if (_context.Employee == null)
          {
              return NotFound();
          }
            return await _context.Employee.ToListAsync();
        }

        // GET: api/Employees/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Employee>> GetEmployee(int id)
        {
          if (_context.Employee == null)
          {
              return NotFound();
          }
            var employee = await _context.Employee.FindAsync(id);

            if (employee == null)
            {
                return NotFound();
            }

            return employee;
        }

        // PUT: api/Employees/5
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPut("{id}")]
        public async Task<IActionResult> PutEmployee(int id, Employee employee)
        {
            if (id != employee.EmpCode)
            {
                return BadRequest();
            }

            _context.Entry(employee).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!EmployeeExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return NoContent();
        }

        // POST: api/Employees
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPost]
        public async Task<ActionResult<Employee>> PostEmployee(Employee employee)
        {
          if (_context.Employee == null)
          {
              return Problem("Entity set 'EmployeeDbContext.Employee'  is null.");
          }
            _context.Employee.Add(employee);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetEmployee", new { id = employee.EmpCode },
employee);
        }

        // DELETE: api/Employees/5
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteEmployee(int id)
        {
            if (_context.Employee == null)
            {
                return NotFound();
            }
            var employee = await _context.Employee.FindAsync(id);
            if (employee == null)
            {
                return NotFound();
            }

            _context.Employee.Remove(employee);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool EmployeeExists(int id)
        {
            return (_context.Employee?.Any(e => e.EmpCode ==
id)).GetValueOrDefault();
        }
    }
}
```

Appsetting.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "EmployeeDbContext": "Server=DESKTOP-
RQ11795\\SQLEXPRESS;Database=Mainprojec1.Data;Trusted_Connection=True;MultipleActive
ResultSets=true"
  }
}
```

After Adding Migration Code

```
using System;
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Mainprojec1.Migrations
{
    public partial class Employeemig : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Employee",
                columns: table => new
                {
                    EmpCode = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    DateOfBirth = table.Column<DateTime>(type: "datetime2",
nullable: false),
                    EmpName = table.Column<string>(type: "nvarchar(max)", nullable:
false),
                    Email = table.Column<string>(type: "nvarchar(max)", nullable:
false),
                    DeptCode = table.Column<int>(type: "int", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Employee", x => x.EmpCode);
                });
        }
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Employee");
        }
    }
}
```