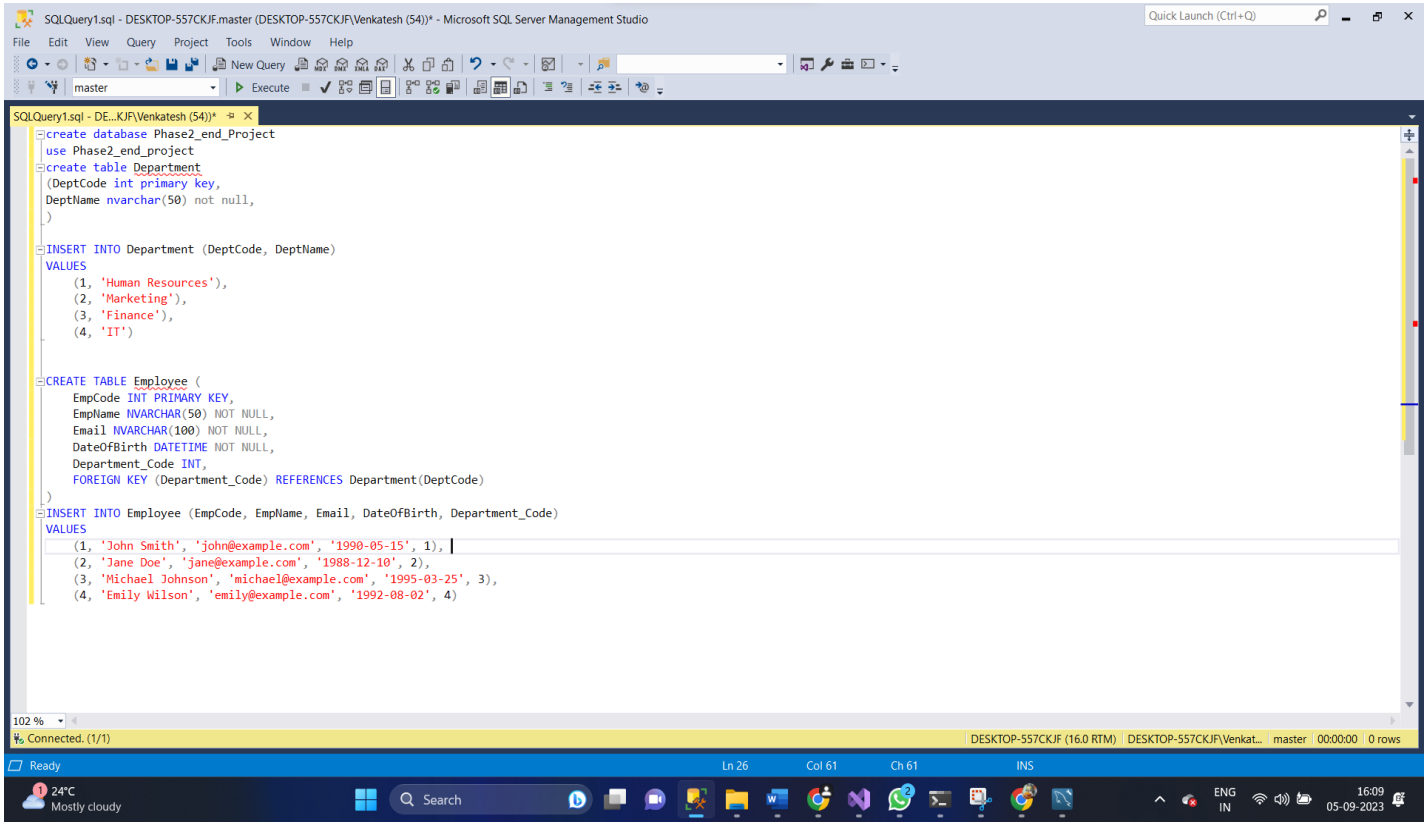
Source Code



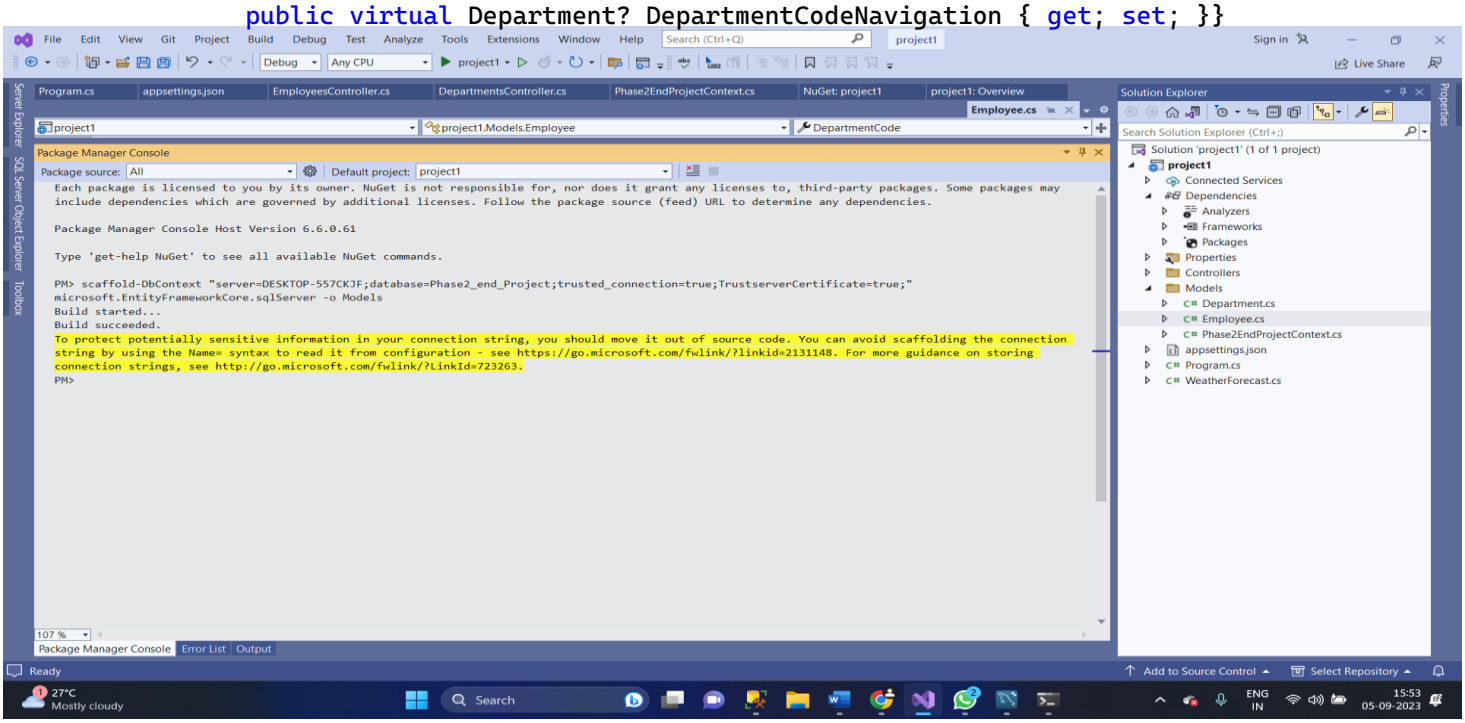
➢ Department Code

```
using System;
using System.Collections.Generic;

namespace project1.Models;

public partial class Department
{
    public int DeptCode { get; set; }

    public string DeptName { get; set; } = null!;

    public virtual ICollection<Employee> Employees { get; set; } = new
List<Employee>();
}
```

➢ Employee Code

```
using System;
using System.Collections.Generic;

namespace project1.Models;
public partial class Employee
{
    public int EmpCode { get; set; }

    public string EmpName { get; set; } = null!;

    public string Email { get; set; } = null!;
```

```
        public DateTime DateOfBirth { get; set; }

        public int? DepartmentCode { get; set; }

        public virtual Department? DepartmentCodeNavigation { get; set; }}
```



➢ DepartmentController Code

```
    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 project1.Models;

namespace project1.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class DepartmentsController : ControllerBase
    {
        private readonly Phase2EndProjectContext _context;

        public DepartmentsController(Phase2EndProjectContext context)
        {
            _context = context;
        }

        // GET: api/Departments
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Department>>> GetDepartments()
        {
          if (_context.Departments == null)
          {
              return NotFound();
          }
```

```
            return await _context.Departments.ToListAsync();
        }

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

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

            return department;
        }

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

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

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

            return NoContent();
        }

        // POST: api/Departments
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPost]
        public async Task<ActionResult<Department>> PostDepartment(Department
department)
        {
          if (_context.Departments == null)
          {
```

```
                return Problem("Entity set 'Phase2EndProjectContext.Departments'
    is null.");
            }
            _context.Departments.Add(department);
            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateException)
            {
                if (DepartmentExists(department.DeptCode))
                {
                    return Conflict();
                }
                else
                {
                    throw;
                }
            }

            return CreatedAtAction("GetDepartment", new { id =
    department.DeptCode }, department);
        }

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

            _context.Departments.Remove(department);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool DepartmentExists(int id)
        {
            return (_context.Departments?.Any(e => e.DeptCode ==
    id)).GetValueOrDefault();
        }
    }
}
```

➢ EmployeeController Code

```
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 project1.Models;
```

```
namespace project1.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class EmployeesController : ControllerBase
    {
        private readonly Phase2EndProjectContext _context;

        public EmployeesController(Phase2EndProjectContext context)
        {
            _context = context;
        }

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

        // GET: api/Employees/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Employee>> GetEmployee(int id)
        {
          if (_context.Employees == null)
          {
              return NotFound();
          }
            var employee = await _context.Employees.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.Employees == null)
          {
              return Problem("Entity set 'Phase2EndProjectContext.Employees'  is
null.");
          }
            _context.Employees.Add(employee);
            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateException)
            {
                if (EmployeeExists(employee.EmpCode))
                {
                    return Conflict();
                }
                else
                {
                    throw;
                }
            }

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

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

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

            return NoContent();
        }
```

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