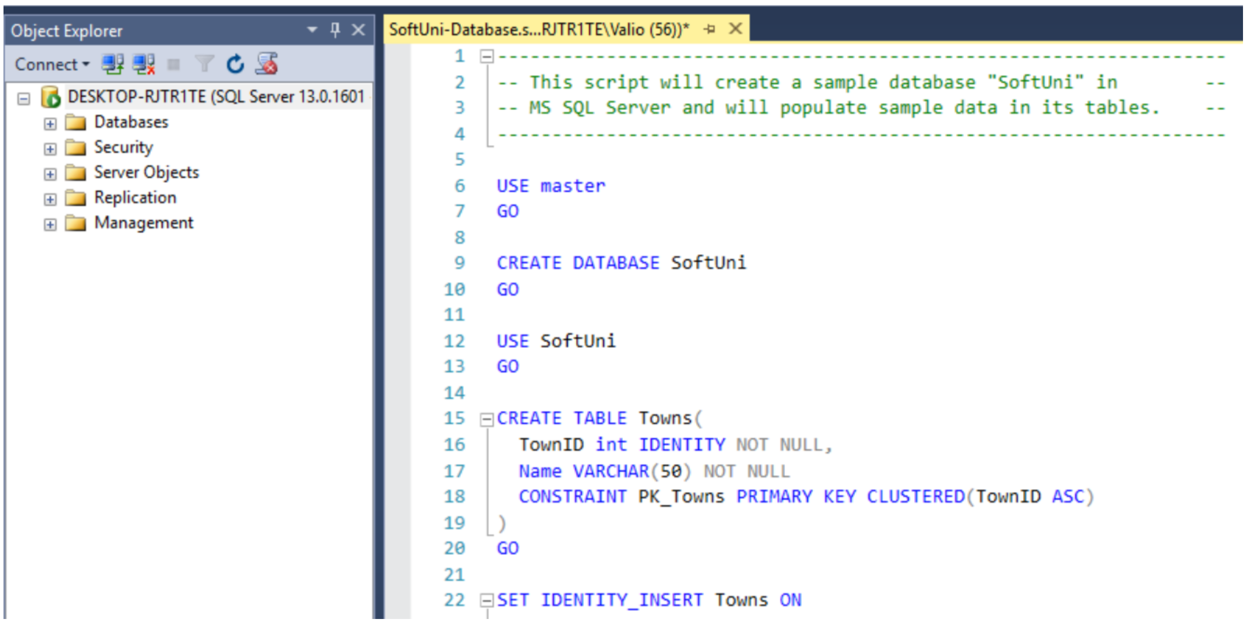
Exercises: Introduction to Entity Framework

This document defines the **exercise assignments** for the ["Databases Advanced – Entity Framework" course @ Software University.](#) Please submit your solutions in [Judge.](#)

1. Import the SoftUni Database

Import the SoftUni database into SQL Management Studio by **executing** the provided **.sql** script.

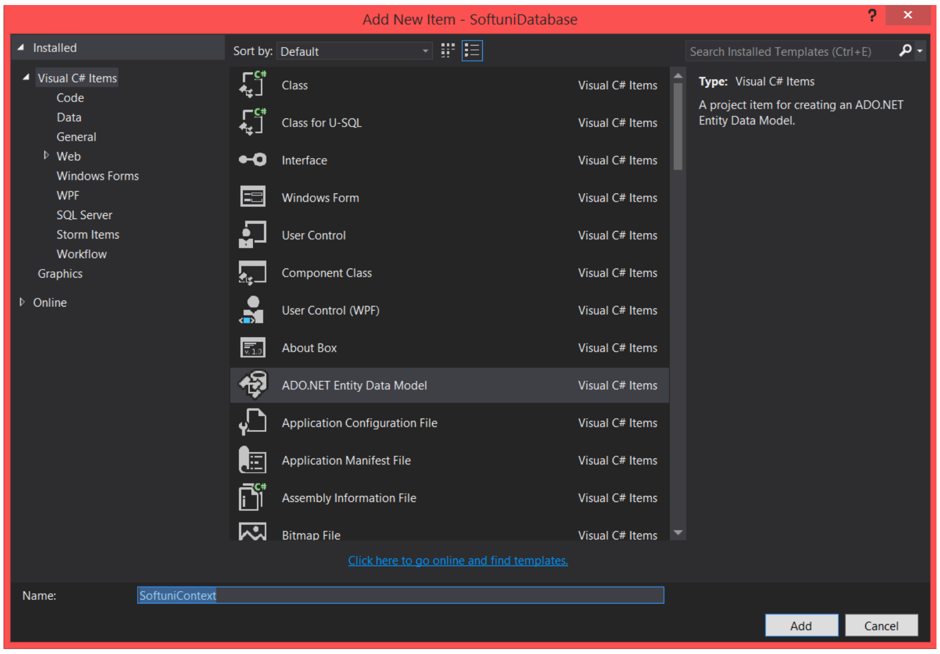

Object Explorer
Connect
DESKTOP-RJTR1TE (SQL Server 13.0.1601
Databases
Security
Server Objects
Replication
Management
SoftUni-Database.s...RJTR1TE\Valio (56))*

```
--------------------------------------------------------------------
-- This script will create a sample database "SoftUni" in         --
-- MS SQL Server and will populate sample data in its tables.     --
--------------------------------------------------------------------

USE master
GO

CREATE DATABASE SoftUni
GO

USE SoftUni
GO

CREATE TABLE Towns(
  TownID int IDENTITY NOT NULL,
  Name VARCHAR(50) NOT NULL
  CONSTRAINT PK_Towns PRIMARY KEY CLUSTERED(TownID ASC)
)
GO

SET IDENTITY_INSERT Towns ON
```

2. Database First

Model the existing database by using Database First.

First create a new empty **ConsoleApplication** and after it is created right click on the project and add a new item. In the newly appeared window select **ADO.NET Entity Data Model** and call it **SoftuniContext** as shown below:


Add New Item - SoftuniDatabase
Installed
Visual C# Items
Code
Data
General
Web
Windows Forms
WPF
SQL Server
Storm Items
Workflow
Graphics
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
Class — Visual C# Items
Class for U-SQL — Visual C# Items
Interface — Visual C# Items
Windows Form — Visual C# Items
User Control — Visual C# Items
Component Class — Visual C# Items
User Control (WPF) — Visual C# Items
About Box — Visual C# Items
ADO.NET Entity Data Model — Visual C# Items
Application Configuration File — Visual C# Items
Application Manifest File — Visual C# Items
Assembly Information File — Visual C# Items
Bitmap File — Visual C# Items
Click here to go online and find templates.
Add
Cancel

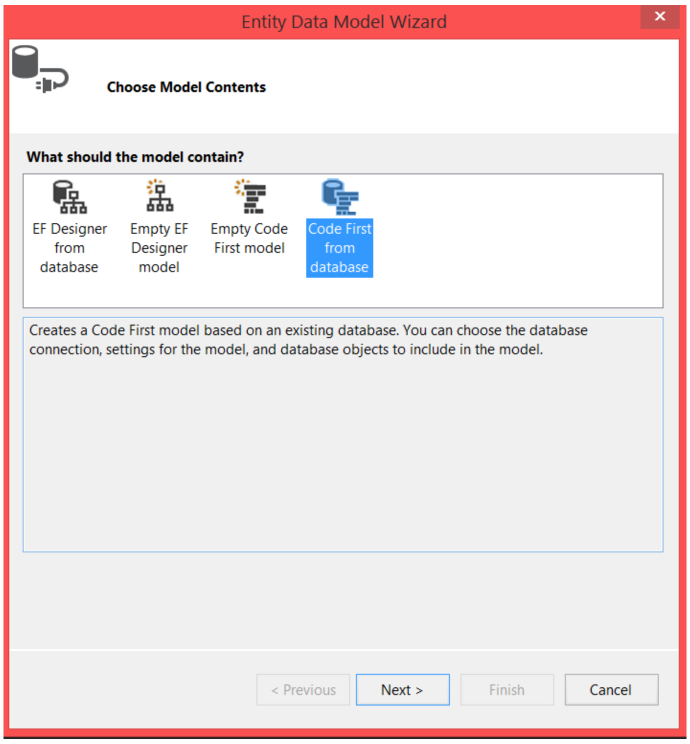
Type: Visual C# Items

A project item for creating an ADO.NET Entity Data Model.

Name: SoftuniContext

© Software University Foundation ([softuni.org](#)). This work is licensed under the [CC-BY-NC-SA](#) license.

Follow us:

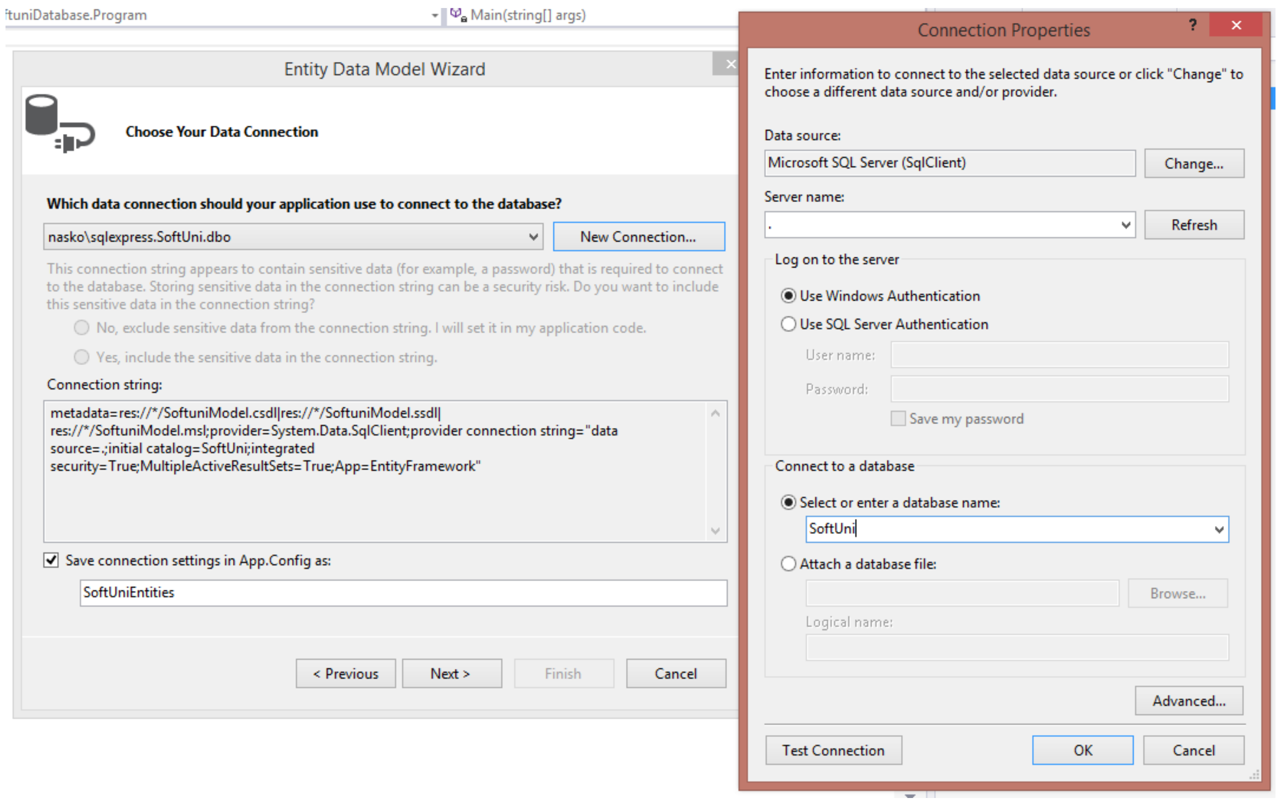


Choose Model Contents

What should the model contain?

Creates a Code First model based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model.

Choose your **server name** (SQL management studio connection) and the **database** you wish to model.



Choose Your Data Connection

Which data connection should your application use to connect to the database?

nasko\sqlexpress.SoftUni.dbo

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

No, exclude sensitive data from the connection string. I will set it in my application code.

Yes, include the sensitive data in the connection string.

Connection string:

```
metadata=res://*/SoftuniModel.csdl|res://*/SoftuniModel.ssdl|
res://*/SoftuniModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=.;initial catalog=SoftUni;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

[✓] Save connection settings in App.Config as: SoftUniEntities

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: .

Log on to the server

[✓ Use Windows Authentication]

Use SQL Server Authentication

User name:

Password:

Save my password

Connect to a database

[✓ Select or enter a database name:] SoftUni

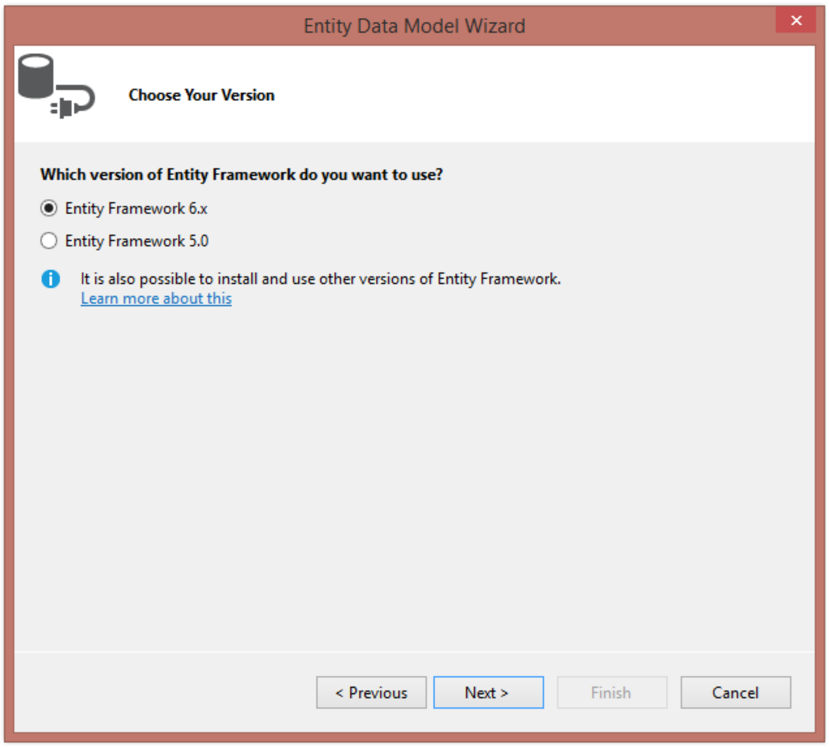
Attach a database file:

Logical name:

Select the latest stable Entitiy Framework version.

© Software University Foundation (softuni.org). This work is licensed under the CC-BY-NC-SA license.

Follow us:


Entity Data Model Wizard

Choose Your Version

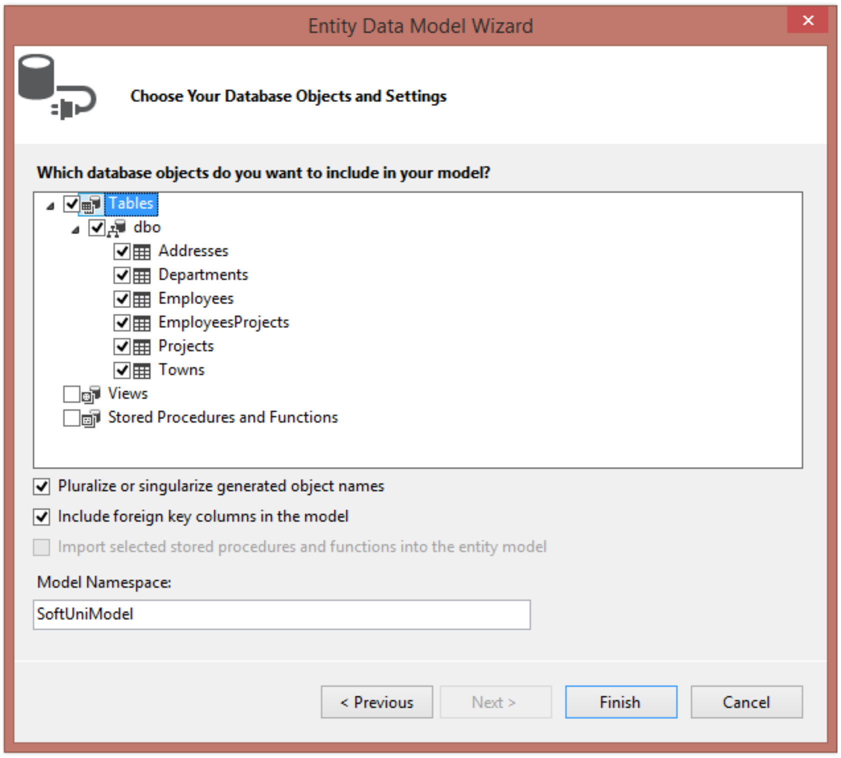
Which version of Entity Framework do you want to use?

[✓ Entity Framework 6.x]

Entity Framework 5.0

It is also possible to install and use other versions of Entity Framework.
Learn more about this

Select the tables the desired tables from the target database. Exclude any views and stored procedures.


Entity Data Model Wizard

Choose Your Database Objects and Settings

Which database objects do you want to include in your model?

- [✓] Tables
  - [✓] dbo
    - [✓] Addresses
    - [✓] Departments
    - [✓] Employees
    - [✓] EmployeesProjects
    - [✓] Projects
    - [✓] Towns
- [ ] Views
- [ ] Stored Procedures and Functions

[✓] Pluralize or singularize generated object names

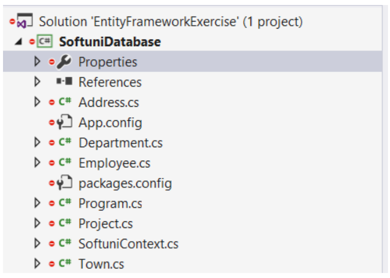
[✓] Include foreign key columns in the model

[ ] Import selected stored procedures and functions into the entity model

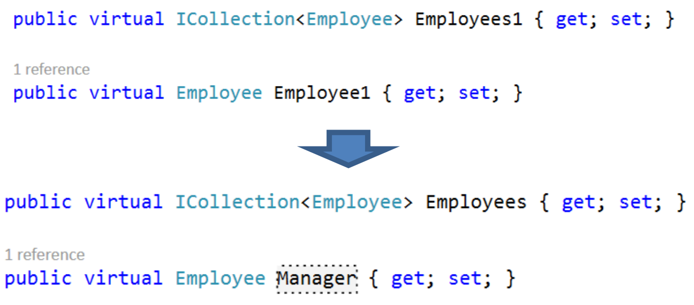
Model Namespace:

SoftUniModel

Click finish. The result should be all the tables that are in your table, generated as classes:


```
Solution 'EntityFrameworkExercise' (1 project)
  SoftuniDatabase
    Properties
    References
    Address.cs
    App.config
    Department.cs
    Employee.cs
    packages.config
    Program.cs
    Project.cs
    SoftuniContext.cs
    Town.cs
```

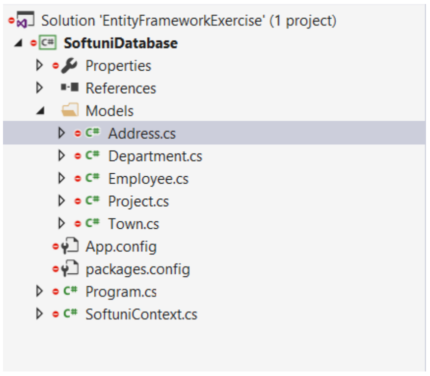
Entity Framework has successfully **mapped the database schema to C# classes**. However, it isn't quite good with names (See the **Employee** class) - e.g. **Employees1**, **Employee1**. Edit the properties and give them more **proper names**.



```
public virtual ICollection<Employee> Employees1 { get; set; }

1 reference
public virtual Employee Employee1 { get; set; }
```

```
public virtual ICollection<Employee> Employees { get; set; }

1 reference
public virtual Employee Manager { get; set; }
```

Finally, before we begin working with the data, please create a new folder in the project called Models and insert your models in it. (Don't forget to change the namespace, once the models are moved).



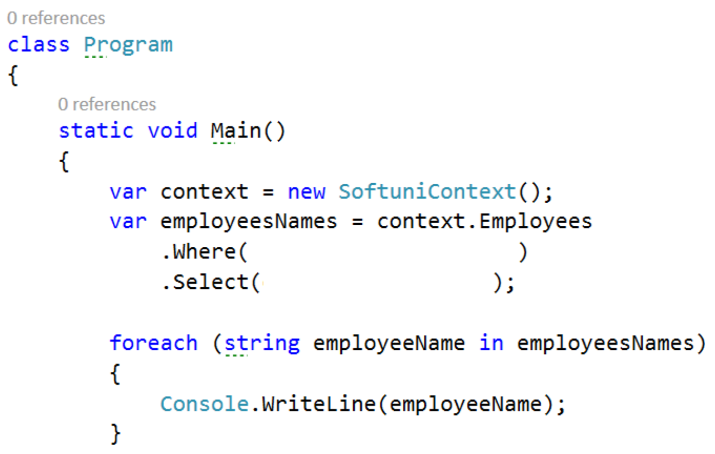
```
Solution 'EntityFrameworkExercise' (1 project)
  SoftuniDatabase
    Properties
    References
    Models
      Address.cs
      Department.cs
      Employee.cs
      Project.cs
      Town.cs
    App.config
    packages.config
    Program.cs
    SoftuniContext.cs
```

3. Employees full information

Let's start writing C# queries! Your task is to extract **all employees** print their **first**, **last** and **middle** name, their **job title** and **salary** separated with a space. (**Test in judge)**

4. Employees with Salary Over 50 000

Your task is to extract **all employees** with **salary** over **50000**. Create a new context and use lambdas to build your query. Make sure the query returns **only the first names** of those employees.

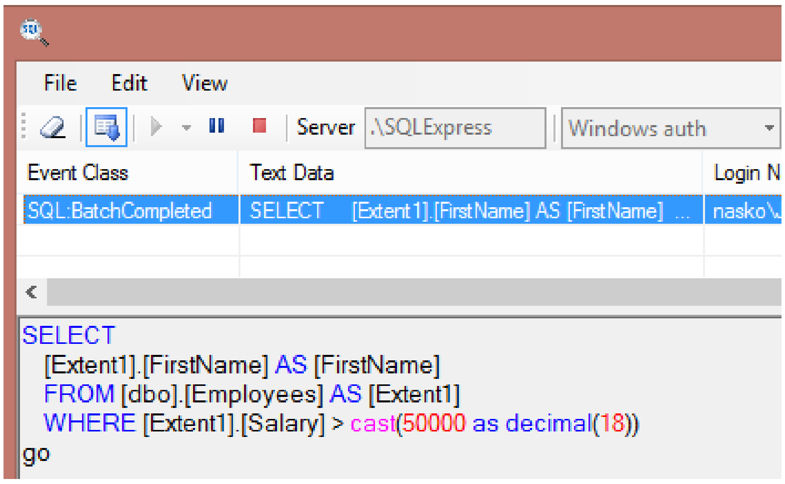


```
0 references
class Program
{
    0 references
    static void Main()
    {
        var context = new SoftuniContext();
        var employeesNames = context.Employees
            .Where(                     )
            .Select(                  );

        foreach (string employeeName in employeesNames)
        {
            Console.WriteLine(employeeName);
        }
```

Use Express Profiler and check if the made query by Entity Framework is correct (there is only one query, but there may be more that are performed by EF for checks).

© Software University Foundation (softuni.org). This work is licensed under the CC-BY-NC-SA license.

Follow us:



Event Class	Text Data	Login N
SQL:BatchCompleted	SELECT [Extent1].[FirstName] AS [FirstName] ...	nasko\

```
SELECT
  [Extent1].[FirstName] AS [FirstName]
  FROM [dbo].[Employees] AS [Extent1]
  WHERE [Extent1].[Salary] > cast(50000 as decimal(18))
go
```

Test in Judge

5. Employees from Seattle

Extract all employees from the **Research and Development** department. Order them by **salary** (in ascending order), then by **first name** (in descending order). Print only their **first name**, **last name**, **department name** and **salary** in the format shown below:

```
var employees =
    context.Employees.Where(employee =>
                                        [redacted])
    [redacted]
    .[redacted]
foreach (var employee in employees)
{
    Console.WriteLine($"{employee.FirstName} {employee.LastName} " +
                      $"from {employee.DepartmentName} - ${employee.Salary:F2}");
}
```

Use Express Profiler and check if the made query by Entity Framework is correct (there is only one query).

```
SELECT
    [Extent1].[EmployeeID] AS [EmployeeID],
    [Extent1].[FirstName] AS [FirstName],
    [Extent1].[LastName] AS [LastName],
    [Extent1].[MiddleName] AS [MiddleName],
    [Extent1].[JobTitle] AS [JobTitle], |
    [Extent1].[DepartmentID] AS [DepartmentID],
    [Extent1].[ManagerID] AS [ManagerID],
    [Extent1].[HireDate] AS [HireDate],
    [Extent1].[Salary] AS [Salary],
    [Extent1].[AddressID] AS [AddressID]
    FROM  [dbo].[Employees] AS [Extent1]
    INNER JOIN [dbo].[Departments] AS [Extent2] ON [Extent1].[DepartmentID] = [Extent2].
[DepartmentID]
    WHERE 'Research and Development' = [Extent2].[Name]
    ORDER BY [Extent1].[Salary] ASC, [Extent1].[FirstName] DESC
go
```

Test in judge

6. Adding a New Address and Updating Employee

Create a new address with **text** "**Vitoshka 15**" and **TownId 4**. Set that address to the employee with last name "Nakov".

```
var address = new Address()
{
    AddressText = "Vitoshka 15",
    TownID = 4
};

Employee employee = null;
//TODO: Get employee with last name 'Nakov'
//TODO: Set address to new address

context.SaveChanges();
```

The above code should successfully **insert a new address** in the database and **set it as Nakov's new address**.

Then order by descending all the employees by their Address' Id, take 10 rows and from them, take the AddressText. Print the results each on a new line:

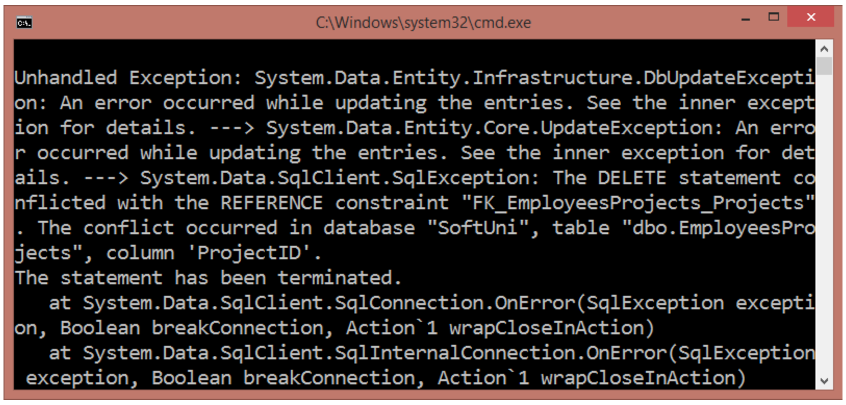
Test in judge

7. Delete Project by Id

Let's delete a project by id(2). Sounds simple:

```
var project = context.Projects.Find(2);
context.Projects.Remove(project);

context.SaveChanges();
```

However, …



```
Unhandled Exception: System.Data.Entity.Infrastructure.DbUpdateExcepti
on: An error occurred while updating the entries. See the inner except
ion for details. ---> System.Data.Entity.Core.UpdateException: An erro
r occurred while updating the entries. See the inner exception for det
ails. ---> System.Data.SqlClient.SqlException: The DELETE statement co
nflicted with the REFERENCE constraint "FK_EmployeesProjects_Projects"
. The conflict occurred in database "SoftUni", table "dbo.EmployeesPro
jects", column 'ProjectID'.
The statement has been terminated.
   at System.Data.SqlClient.SqlConnection.OnError(SqlException excepti
on, Boolean breakConnection, Action`1 wrapCloseInAction)
   at System.Data.SqlClient.SqlInternalConnection.OnError(SqlException
 exception, Boolean breakConnection, Action`1 wrapCloseInAction)
```

The project is **referenced** by the junction (many-to-many) table **EmployeesProjects**. Therefore we cannot safely delete it. First, we need to remove any references to that row in the **Projects** table.

This is done by removing the project from all employees who reference it. **Tip**: Get those employees from the **Employee** navigation property.

```
var project = context.Projects.Find(2);
//TODO: Delete project from employees who reference it

context.Projects.Remove(project);

context.SaveChanges();
```

Then take 10 projects, select their names and print them on the console each on a new line.

Test in judge.

8. Find employees in period

Find the first **30** employees who have **projects** started in the time period **2001 - 2003** (inclusive). Print each employee's **first name**, **last name** and **manager's first name** and each of their projects' **name**, **start date**, **end date**. Here is the format:

"first Name lastName managerFirstName"

"—projectName projectStart projectEnd"

Test in judge

9. Addresses by town name

Find all addresses, **ordered** by the **number of employees** who live there (descending), then by **town name** (ascending). Take only the **first 10 addresses**. For each address print it in the format

"addressText, townName – numberOfEmployees employees"

Test in judge

10. Employee with id 147 sorted by project names

Get an **employee with id 147.** Print only his/her **first name**, **last name**, **job title** and **projects** (print only their names). The projects should be **ordered by name** (ascending). Format of the output:

"first Name lastName job Title"

"--projectName"

Test in judge

11. Departments with more than 5 employees

Find **all departments** with more than **5 employees**. Order them by **employee count** (ascending).Print the **department name**, **manager's first name** and first name, last name and job title of every **employee**. Format of the output:

"department.Name manager.FirstName"

"employee1.FirstName employee1.LastName employee1.JobTitle"

…

Test in judge

12. *Native SQL Query

Find all **employees** who have projects with **start date** in the year **2002**. Select only their **first name**. Solve this task by using both **LINQ query** and **native SQL query** through the context.

Measure the difference in performance in both cases with a **Stopwatch**.

© Software University Foundation (softuni.org). This work is licensed under the CC-BY-NC-SA license.

Follow us:

```
var timer = new Stopwatch();
timer.Start();
PrintNamesWithNativeQuery(context);
timer.Stop();
Console.WriteLine($"Native: {timer.Elapsed}");

timer.Restart();
PrintNameWithLinq(context);
timer.Stop();
Console.WriteLine($"Linq: {timer.Elapsed}");
```

Tip: Use the **`context.Database.SqlQuery<T>()`** method for executing native SQL queries.

13. * Play with the SQL Server Profiler

Your task is to use the SQL Server ExpressProfiler to view all your queries from the previous homework task.

14. * Explore the Full Source Code of Entity Framework

Your task is to download (clone the repository) and explore the full source code of Entity Framework. You can find it on http://entityframework.codeplex.com. Do not submit anything for this problem.

15. Find Latest 10 Projects

Write a program that prints information about **last 10 started projects**. Print **their name, description, start and end date** and **sort them by name** lexicographically. Format of the output:

“Name Description StartDate EndDate”

Test in judge

16. Increase Salaries

Write a program that increase salaries of all employees that are in the **Engineering**, **Tool Design**, **Marketing** or **Information Services** department by **12%**. Then **print first name, last name and salary** for those employees whose salary was increased. Format of the output:

“FirstName LastName $Salary”

Test in judge

17. Remove Towns

Write a program that **deletes town** which name is given as an input. Also, **delete all addresses** that are in those towns. Print on the console the number addresses that were deleted. You may also note that there are some Employees in that have addresses which may be a problem for deleting the addresses. So first off start by setting the AddressID of each employee in for the given address to null. After all of them are set to null, you may safely remove all the addresses from the context.Addresses and finally remove the given town. You may test this task locally, so that you can see what happens for more than 1 case of deletion.

Example

Input	Output
`Sofia`	`1 address in Sofia was deleted`
`Seattle`	`44 addresses in Seattle were deleted`

18. Find Employees by First Name starting with ‘SA’

Write a program that finds all employees whose first name starts with ‘SA’. Print their first, last name, their job title and salary in the format given in the example below.

```
FirstName LastName – JobTitle - ($Salary)
```

Test in judge

19. First Letter

Use the **Gringotts database**. Write a program that prints all unique wizard first letters of their **first names** only if they have deposit of type Troll Chest. Order them alphabetically. Use DISTINCT for uniqueness.

Example

Output
A
…

Test in judge

© Software University Foundation (softuni.org). This work is licensed under the CC-BY-NC-SA license.

Follow us: