



LINQ and Entity Framework Core Lab Book



Document Revision History

Date	Revision No.	Author	Summary of Changes
6-Jul-2020	1.0	RamKumar	Implementing EF Core



Table of Contents

Lab 1.	LINQ Basics	5
Lab 2.	Using LINQ to DataSet / DataTable.....	7
Lab 3.	Implementing DB First Approach in EF Core	8
Lab 4.	Implementing Code First Approach in EF Core	10
Lab 5.	Basic Query Operations using LINQ to Entities	11



Getting Started

Overview

This Lab book is a guided tour for LINQ and Entity Framework Core. It comprises solved examples and 'To Do' assignments. Follow the steps provided in the solved examples and work out the 'To Do' assignments given.

Setup Checklist for LINQ and Entity Framework Core

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

Processor, HDD & RAM

- Processor - Minimum: Intel Core 3 or Higher
- Processor speed: Minimum: 1.4 GHz
- Recommended: 2.0 GHz or faster
- RAM - Minimum: 2 GB, Recommended: 4 GB or more
- HDD – 150 GB

Operating System

- Windows 7 Professional 64 bit
- Windows 10 Professional 64 bit

Please ensure that the following is done:

- Visual Studio 2017 or above

Instructions

- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory EFCore_Assgn. For each lab exercise create a directory as lab <lab number>.
- You may also look up the on-line help provided in the MSDN library.

Learning More (Bibliography if applicable)

- <http://msdn.microsoft.com>
- <http://www.asp.net/entity-framework>
- <https://msdn.microsoft.com/en-in/data/ef.aspx>
- Entity Framework 6 Recipes by Apress publication

**Lab 1. LINQ Basics**

Goals	Understand the process of Implementing LINQ to a Collection
	Learn to use LINQ
	Learn to use LINQ Operators
Time	60 minutes

1. Create .NET Core console application and add class named Employee with following field.

Employee Class**EmployeeID (Integer)****FirstName (String)****LastName (String)****Title (String)****DOB (Date)****DOJ (Date)****City (String)**

2. Create a Generic List Collection empList and populate it with the following records.

EmployeeID	FirstName	LastName	Title	DOB	DOJ	City
1001	Malcolm	Daruwalla	Manager	16/11/1984	8/6/2011	Mumbai
1002	Asdin	Dhalla	AsstManager	20/08/1984	7/7/2012	Mumbai
1003	Madhavi	Oza	Consultant	14/11/1987	12/4/2015	Pune
1004	Saba	Shaikh	SE	3/6/1990	2/2/2016	Pune
1005	Nazia	Shaikh	SE	8/3/1991	2/2/2016	Mumbai
1006	Amit	Pathak	Consultant	7/11/1989	8/8/2014	Chennai
1007	Vijay	Natrajan	Consultant	2/12/1989	1/6/2015	Mumbai
1008	Rahul	Dubey	Associate	11/11/1993	6/11/2014	Chennai
1009	Suresh	Mistry	Associate	12/8/1992	3/12/2014	Chennai
1010	Sumit	Shah	Manager	12/4/1991	2/1/2016	Pune

3. Now once the collection created write down and execute the LINQ queries for collection as follows
 - a. Display detail of all the employee
 - b. Display details of all the employee whose location is not Mumbai
 - c. Display details of all the employee whose title is AsstManager
 - d. Display details of all the employee whose Last Name start with S
 - e. Display a list of all the employee who have joined before 1/1/2015



LINQ AND ENTITY FRAMEWORK CORE LAB BOOK

- f. Display a list of all the employee whose date of birth is after 1/1/1990
- g. Display a list of all the employee whose designation is Consultant and Associate
- h. Display total number of employees
- i. Display total number of employees belonging to "Chennai"
- j. Display highest employee id from the list
- k. Display total number of employee who have joined after 1/1/2015
- l. Display total number of employee whose designation is not "Associate"
- m. Display total number of employee based on City
- n. Display total number of employee based on city and title
- o. Display total number of employee who is youngest in the list

**Lab 2. Using LINQ to DataSet / DataTable**

Goals	Implementing LINQ to DataSet or LINQ to Datatable
Time	45 minutes

Create table & insert records based on following SQL Script provided.

```
create table Employee(  
    EmployeeID int,  
    FirstName varchar(20),  
    LastName varchar(20),  
    Title varchar(20),  
    DOB datetime,  
    DOJ datetime,  
    City varchar(20)  
);
```

insert into Employee values

```
(1001, 'Malcolm', 'Daruwalla', 'Manager', '11/16/1984', '6/8/2011', 'Mumbai'),  
(1002, 'Asdin', 'Dhalla', 'AsstManager', '08/20/1984', '7/7/2012', 'Mumbai'),  
(1003, 'Madhavi', 'Oza', 'Consultant', '11/14/1987', '4/12/2015', 'Pune'),  
(1004, 'Saba', 'Shaikh', 'SE', '6/3/1990', '2/2/2016', 'Pune'),  
(1005, 'Nazia', 'Shaikh', 'SE', '3/8/1991', '2/2/2016', 'Mumbai'),  
(1006, 'Suresh', 'Pathak', 'Consultant', '11/7/1989', '8/8/2014', 'Chennai'),  
(1007, 'Vijay', 'Natrajan', 'Consultant', '12/2/1989', '6/1/2015', 'Mumbai'),  
(1008, 'Rahul', 'Dubey', 'Associate', '11/11/1993', '11/6/2014', 'Chennai'),  
(1009, 'Amit', 'Mistry', 'Associate', '8/12/1992', '12/3/2014', 'Chennai'),  
(1010, 'Sumit', 'Shah', 'Manager', '4/12/1991', '1/2/2016', 'Pune');
```

select * from Employee

1. Write LINQ to DataSet Queries for following requirements
 - a. Sort & Display Employee by their Joining Year in ascending order & then by their First-Name in descending order.
 - b. Get Average Age in Years (Eg-32.87) of Manager Employees
 - c. Group Employees by their City in such way that younger Employees will be displayed on top

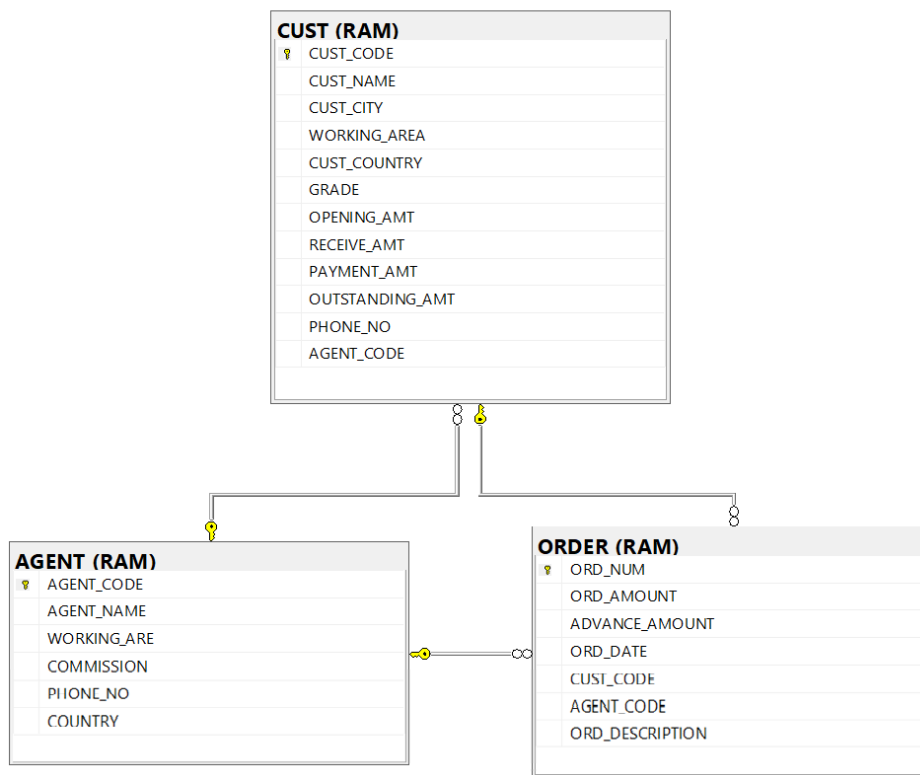
**Lab 3. Implementing DB First Approach in EF Core**



Goals	Understand the process of Creating Entity Data Model using Database First Approach Learn to use of DB First Approach to access tables from DB in EF Core
Time	90 minutes

Database First Approach: -

Entity Framework Core's Database First approach allows developers to build software applications from their existing databases. You connect to an existing database and Visual Studio and EF Core build a data object model and the complete application for you with very little code.

Create tables based on below DB diagram (with their Schema). Use the DDL & DML Script provided below the diagram



DDL Script	DML Script
 DDLScript.txt	 DMLScript.txt



Create .NET Core Console Application which should establish connection to the DB using “DB First Approach”. In the Console Application, above table details should be stored in “Model” folder.

In the Program.cs file, create static method called “ShowAllAgent()” which should display all Agent details and “ShowAllOrder()” method will display all order details using project’s EF Core DB First Approach.

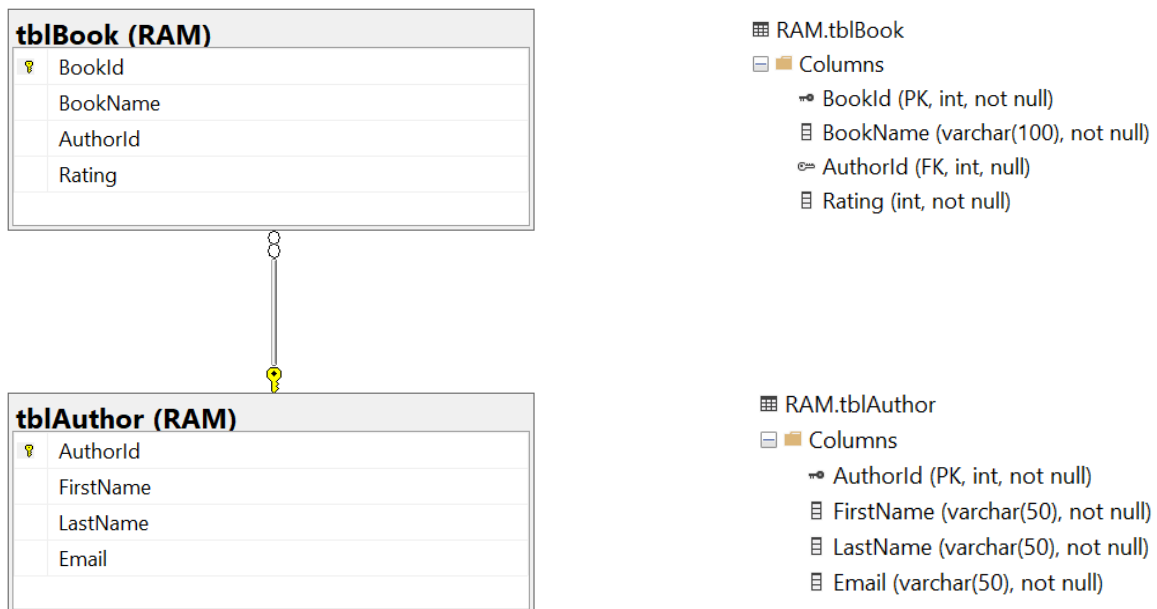
Invoke above methods in the main and ensure you are getting all the Agent and Order related details in the Console.

**Lab 4. Implementing Code First Approach in EF Core**

Goals	Learn to use EF Core Code First Approach to create & access tables from Database
Time	120 minutes

Code First Approach:-

Code First is a technique which helps us to create a database, migrate and maintain the database and its tables from the code. From the code, means that you directly maintain the database and its corresponding tables from the .NET Code. It is helpful when you don't have a database ready and you want to start working with new fresh project and want to create a database and maintain the database directly from your code.



Create .NET Core Console Application which should establish connection to the DB using “Code First Approach”. Based on above diagram, create the necessary entity class in the Model folder. Add the proper attributes and navigational properties in the file. Apply Migration, so that Entity class implementation should get reflected in the DB.

In the Program.cs file, write appropriate program logic that should display number of records present in the tblAuthor table with proper Exception handling Mechanism.

Lab 5. Basic Query Operations using LINQ to Entities

Goals	Learn to use of LINQ to Entities Learn to Manipulate Data
Time	120 minutes

Question:-

Create .NET Core Console Application which establish connection to SQL Server using “DB First Approach” for the following tables.

Server: **NDAMSSQL\SQLILEARN**

Database: **Training**

Tables:

[dbo].Staff_Master

[dbo].Student_Master

(Records already inserted in both the tables)

STAFF MASTER	
Staff_code	
Staff_Name	
Design_code	
Dept_code	
HireDate	
Staff_dob	
Staff_address	
Mgr_code	
Staff_sal	
MGR_NAME	

dbo.STAFF_MASTER
Columns
Staff_code (PK, int, not null)
Staff_Name (varchar(50), not null)
Design_code (FK, int, null)
Dept_code (FK, int, null)
HireDate (datetime, null)
Staff_dob (datetime, null)
Staff_address (varchar(240), null)
Mgr_code (int, null)
Staff_sal (decimal(10,2), null)
MGR_NAME (varchar(255), null)

Student master	
Stud_Code	
Stud_Name	
Dept_Code	
Stud_Dob	
Address	
doj	

dbo.Student_master
Columns
Stud_Code (PK, numeric(6,0), not null)
Stud_Name (varchar(30), not null)
Dept_Code (FK, numeric(2,0), null)
Stud_Dob (datetime, null)
Address (varchar(30), null)
doj (int, null)

Now we have to write LINQ query against the model for reading the data.

Write Linq queries in Program.cs file for the following.

- 1) To display staff details write the following query

```
static void Main(string[] args)
{
    trainingEntities context = new trainingEntities();

    var query = from staff in context.Staff_Master
                select staff;

    //Displaying details from Staff_Master Entity
    foreach (Staff_Master s in query)
    {
        Console.WriteLine("Staff Code= {0},Name = {1},HireDate = {2}",
                           s.Staff_Code,s.Staff_Name,s.Hiredate);
    }
}
```

- 2) To display list of employee whose salary is more than 30000

```
static void Main(string[] args)
{
    trainingEntities context = new trainingEntities();

    var query = from staff in context.Staff_Master
                where staff.Salary >30000
                select staff;

    foreach (Staff_Master s in query)
    {
        Console.WriteLine("Staff Code= {0},Name = {1},Salary = {2}",
                           s.Staff_Code,s.Staff_Name,s.Salary);
    }
}
```

Perform the following query by yourself

- 3) Display the list of student(s) where city is not null
- 4) Display the list of student(s) which includes Student name, department and date of birth
- 5) Display count of total student belonging to Bangalore
- 6) Display list of employees whose salary is more than the average salary of the employee.



Data Manipulation:-

Now we will perform CUD operation on the model that we have created.

- 7) Use the Student_Master Entity to ADD a record to the student master table. Create a static method called AddStudentData() to implement the logic
- 8) Update the Student_master Address to “Bangalore” for the StudentCode “1020”. Create a static method called UpdateStudentData() to implement the logic.
- 9) Whatever record you added using AddStudentData(), delete that record from the student_master table. Create a static method called DeleteStudentData() to implement the logic
- 10) Retrieve all staff related details like Staffcode, Staffname, Deptcode and Staffaddress from Staff_master table using RawSql() and display output in the console application
- 11) Create a stored procedure which accepts Student_master related details as parameter and add those details in the table. Invoke the SP in the console application (.NET Core) and ensure record got saved in the database.