

# ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3

Использование библиотеки LINQ для выполнения запросов в базу данных MS SQL.

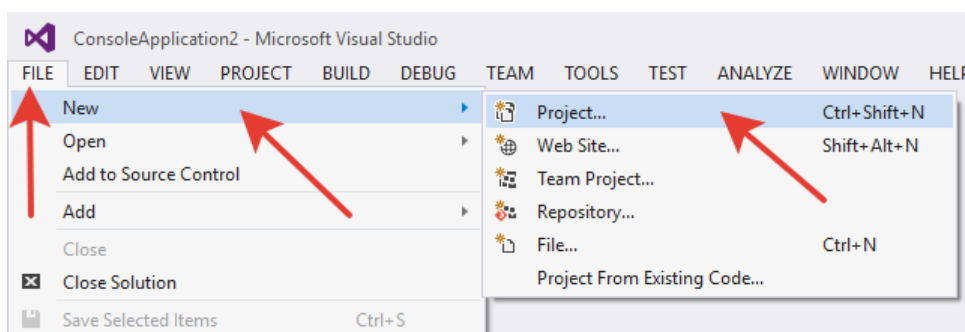
## БИБЛИОТЕКА C# LINQ

В лабораторной работе №2 мы рассмотрели, как можно организовать работу с базой данных при помощи стандартных компонент и текстовых запросов. К сожалению, такой подход не всегда удобен, у программиста часто возникает потребность работы с базой данных без использования табличного представления. К счастью, в 2008 году в язык C# была добавлена библиотека запросов **LINQ**. Библиотека **LINQ** скрывает весь сложный механизм работы с базой данных, оставляя только классическую механику объектно-ориентированных языков.

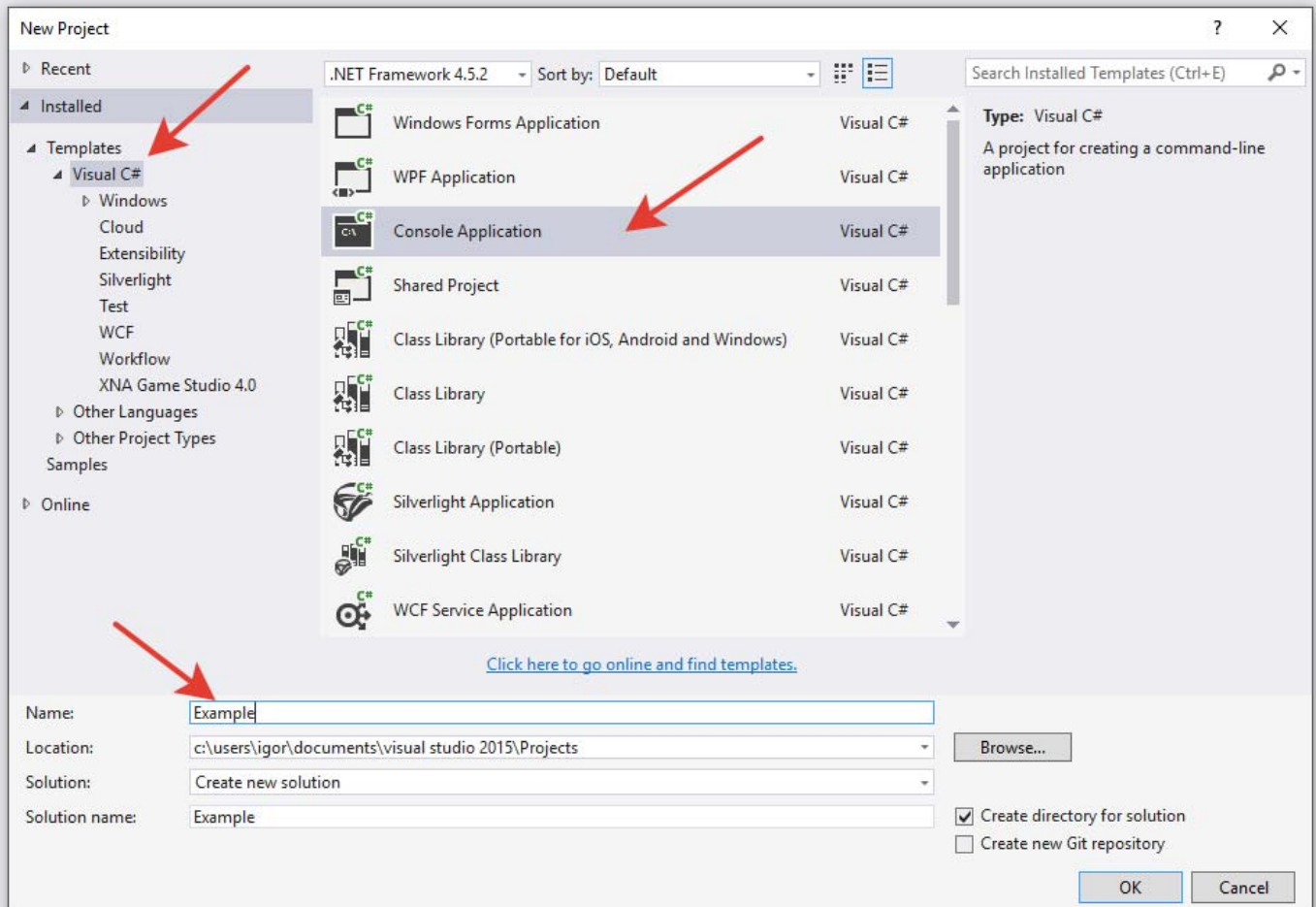
Для выполнения лабораторной работы №3 нам потребуется база данных **aero**, используемая в лабораторных работах №1 и №2.

## ИСПОЛЬЗОВАНИЕ LINQ

Откройте **Visual Studio** и создайте новый проект C# **Console Application**.



Консольное приложение выбрано исключительно для упрощения понимания работы с библиотекой **LINQ**.



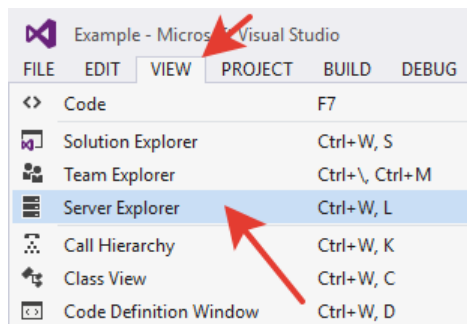
Введите название проекта, например **Example** и нажмите кнопку **OK**.

Конструктор сгенерирует нам классическое простейшее консольное приложение, состоящее из одного файла с исходным кодом – **Program.cs**:

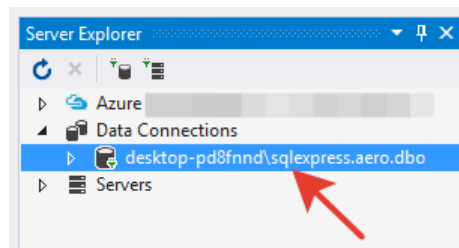
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Example {
    class Program {
        static void Main(string[] args) {
        }
    }
}
```

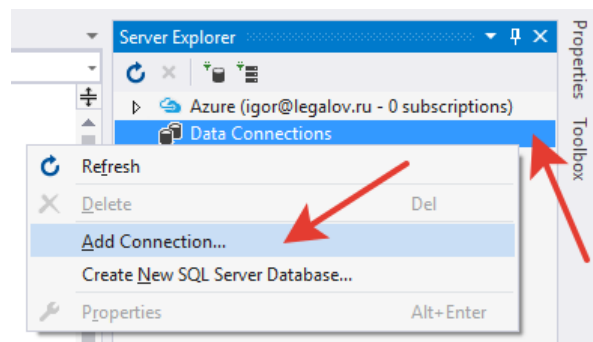
Откройте панель обозревателя серверов (Server Explorer). Для этого выберите пункт меню **Вид – Обозреватель серверов (View – Server Explorer)**:



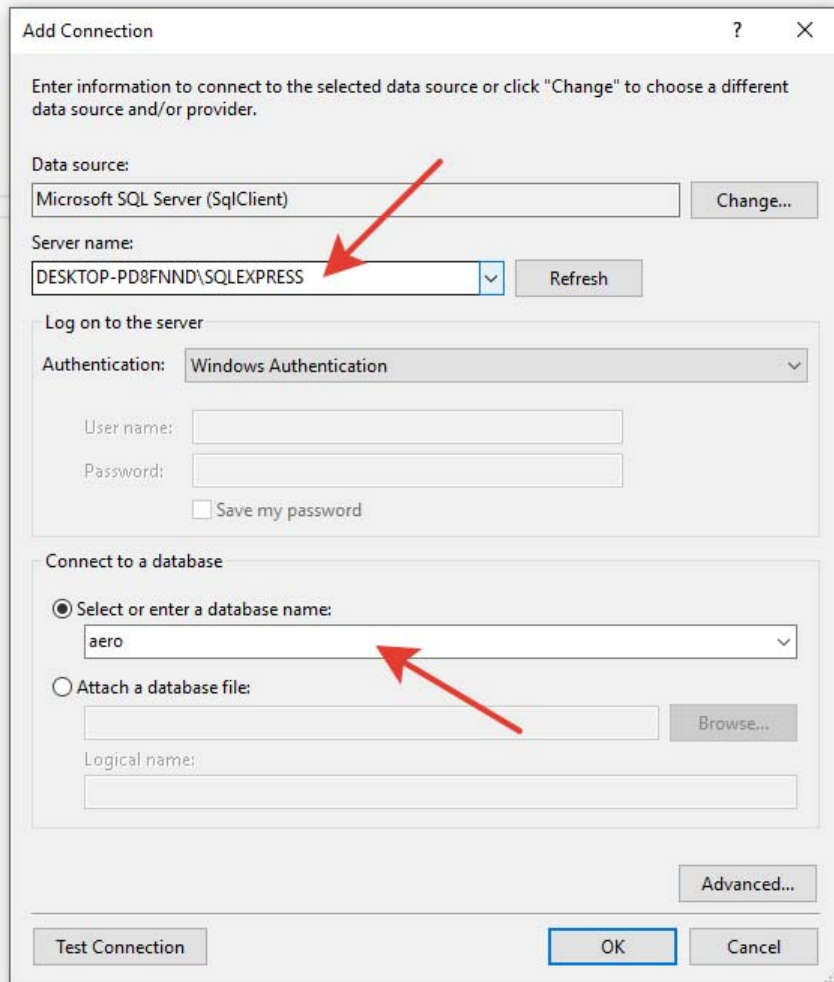
В панели обозревателя серверов найдите узел **Подключения данных (Data Connections)**. Раскройте этот узел, если внутри имеются элементы, то выберите их и удалите, нажав клавишу **Delete**.



Нажмите правой кнопкой мыши на узле **Подключения данных (Data Connections)** и выберите пункт **Добавить подключение:**

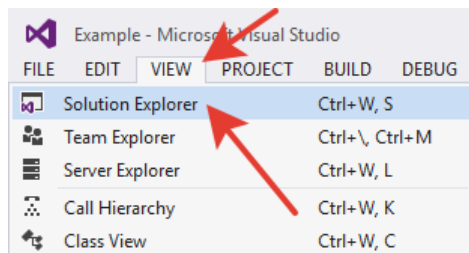


В открывшемся диалоговом окне введите название вашего сервера (см. лабораторные работы №1 и №2), в качестве используемой базы данных выберите **aero**:

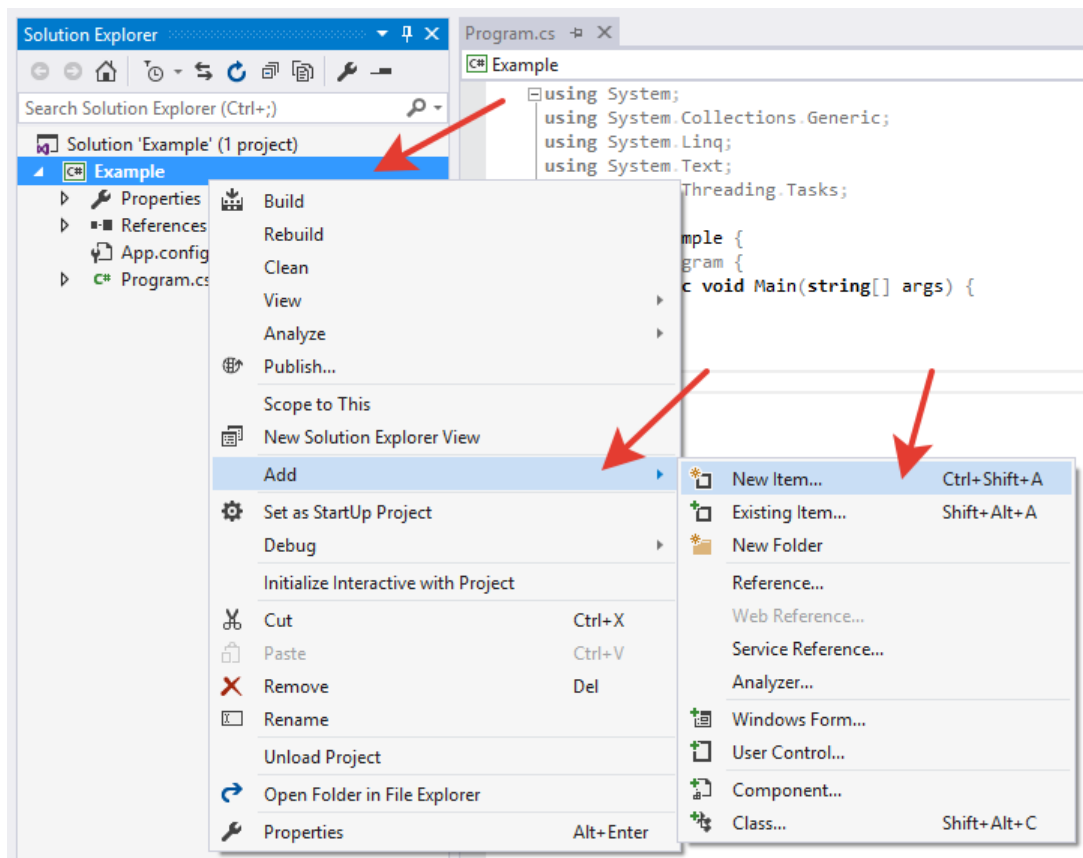


Нажмите кнопку **ОК**.

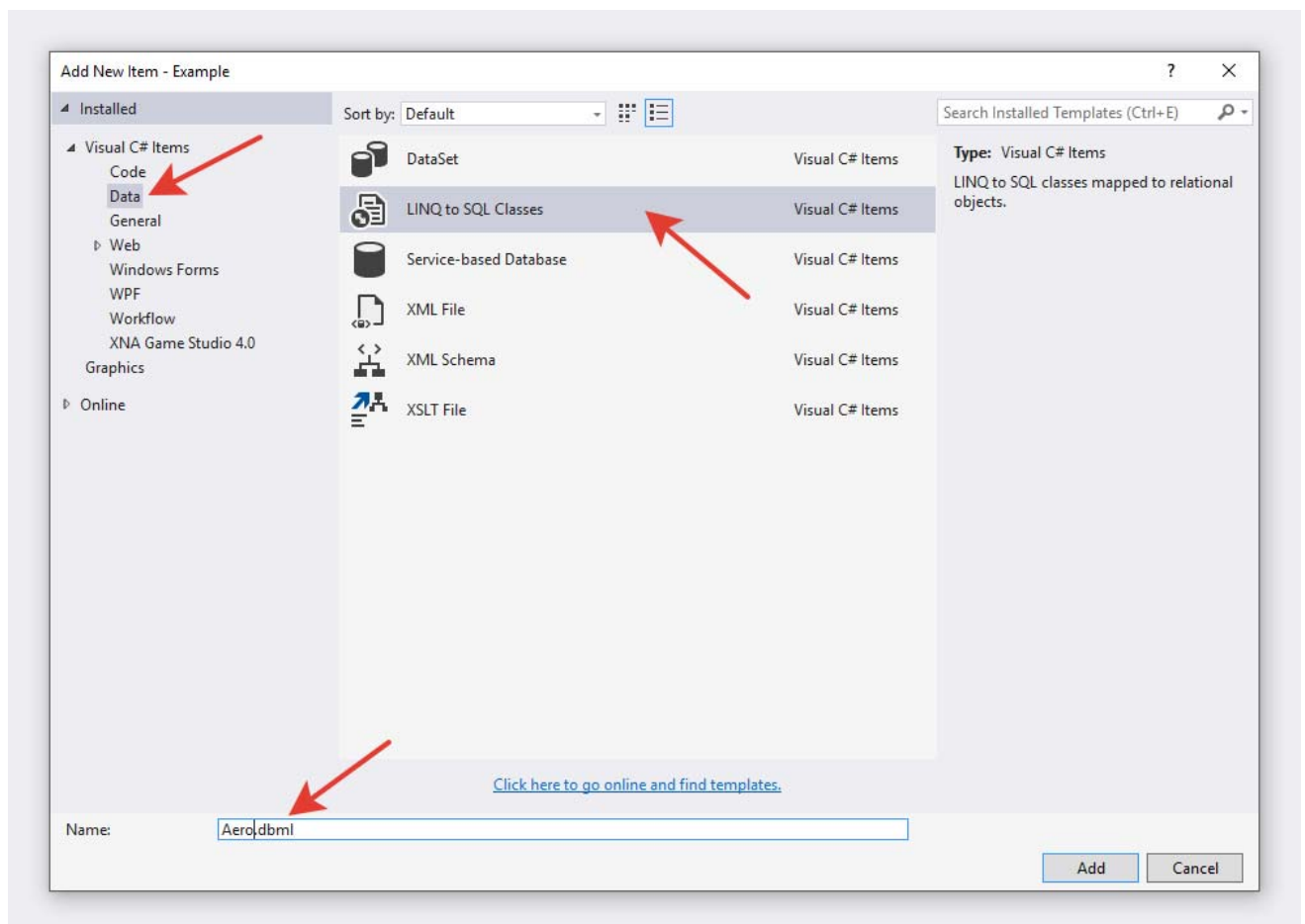
Откройте панель обозревателя решений. Для этого выберите пункт меню **Вид – Обозреватель решений (View – Solution Explorer)**:



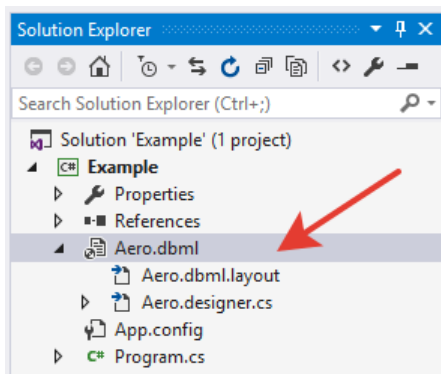
На панели обозревателя решений найдите узел проекта, он называется **Example**, и щелкните на нем правой кнопкой мыши. В выпадающем меню выберите пункты **Добавить – Новый элемент (Add – New Item)**.



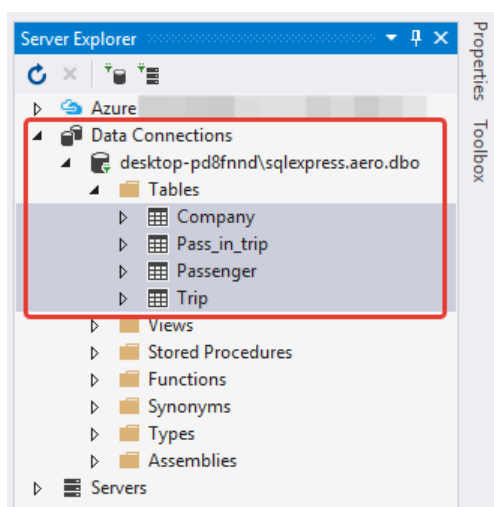
В открывшемся диалоговом окне найдите пункт **LINQ to SQL Classes** в разделе **Данные (Data)**. В качестве имени укажите **Aero**:



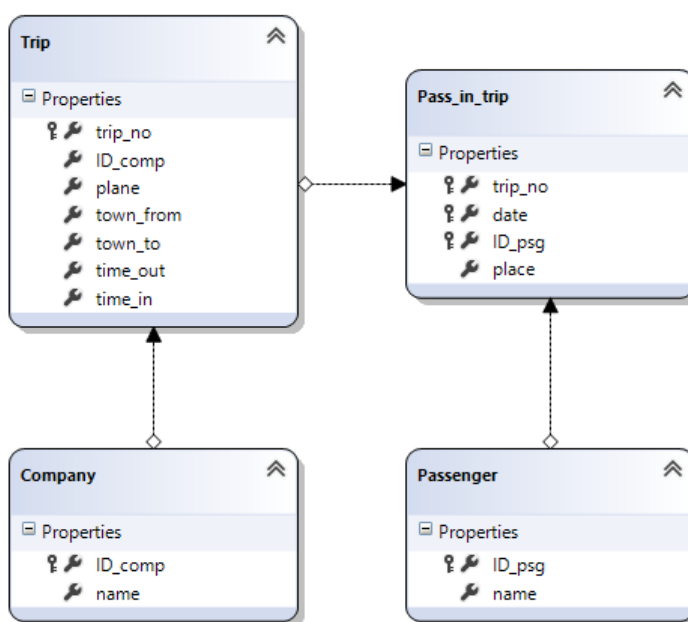
Откроется окно **конструктора отношений**. Также, в панели обозревателя решений, появится новый узел - **Aero.dbml**. Если вы случайно закроете конструктор отношений, то всегда сможете снова открыть его дважды щелкнув на узле **Aero.dbml**:



Откройте панель обозревателя серверов. Внутри обозревателя серверов раскройте последовательно узлы **Подключения данных(Data Connections)** – **<узел с названием вашего сервера>** - **Таблицы (Tables)**. Выделите все имеющиеся таблицы и перетащите их в левую половину **конструктора отношений**.



В конструкторе отношений должна появиться схема таблиц и связей между ними:



Одновременно с этим среда **Visual Studio** сгенерирует файл **Aero.designer.cs** с набором классов. Структура классов будет соответствовать таблицам из базы данных **Aero**. Класс, представляющий саму базу данных, будет называться **AeroDataContext**.

Приступим к написанию кода. Посредством панели обозревателя решений откройте файл **Program.cs**.

---

## ПРОГРАММИРОВАНИЕ С LINQ

---

### ЧТЕНИЕ ДАННЫХ

Первый наш запрос будет читать из базы данных содержимое таблицы **Passenger** и выводить его в консоль.

Найдите функцию **Main**.

Для начала работы с базой данных нам необходимо создать объект класса **AeroDataContext**. Для этого достаточно одной строчки кода:

```
var db = new AeroDataContext();
```

Теперь выполним запрос на выборку данных. Для этого мы будем использовать новые ключевые слова языка C# - **from**, **where** и **select**. Добавьте следующий код:

```
var query = from c in db.Passengers
            select c;
```

Этот код очень похож по своему синтаксису на запрос SQL, и, по сути, им и является.

В базе данных **bd** (наша **aero**) выбирается содержимое таблицы **Passenger** и сохраняется в список **Passengers**. Список **Passengers** был автоматически создан конструктором отношений. Для каждого элемента **c** списка **Passengers** выбирается все его содержимое без каких-либо ограничений или преобразований.

Данный запрос будет сохранен для дальнейшего использования в объекте **query**.

Теперь мы можем выполнять различные операции над результатами запроса. Например, давайте распечатаем содержимое таблицы **Passenger**:

```
foreach (var q in query) {
    Console.WriteLine(q.ID_psg + ". " + q.name);
}
Console.ReadKey();
```

Переменная **q** хранит в себе одну строку из результата выполнения запроса **query**. Переменная **q** принадлежит автоматически сгенерированному классу **Passenger**, поэтому мы можем получить отдельные поля таблицы при помощи свойств **ID\_psg** и **name**.

Результат работы программы:



```
file:///c:/users/igor/documents/visui
4. Donald Sutherland
5. Jennifer Lopez
6. Ray Liotta
7. Samuel L. Jackson
8. Nikole Kidman
9. Alan Rickman
10. Kurt Russell
11. Harrison Ford
12. Russell Crowe
13. Steve Martin
14. Michael Caine
15. Angelina Jolie
16. Mel Gibson
17. Michael Douglas
18. John Travolta
19. Sylvester Stallone
20. Tommy Lee Jones
21. Catherine Zeta-Jones
22. Antonio Banderas
23. Kim Basinger
24. Sam Neill
25. Gary Oldman
26. Clint Eastwood
27. Brad Pitt
28. Johnny Depp
29. Pierce Brosnan
30. Sean Connery
31. Bruce Willis
37. Mullah Omar
```

Можно выбрать из таблицы пассажиров только имена:

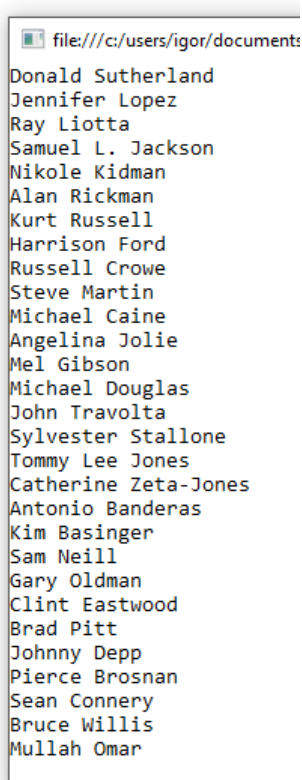
```
var query = from c in db.Passengers
            select c.name;
```

В таком случае тип переменной **q** изменится и станет **string** вместо **Passenger**:

```
foreach (var q in query) {
    Console.WriteLine(q);
}
Console.ReadKey();
```

Результат работы программы:



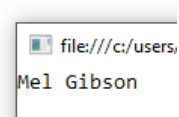
A screenshot of a text file with the path 'file:///c:/users/igor/documents'. The file contains a list of 25 names, one per line, including Donald Sutherland, Jennifer Lopez, Ray Liotta, Samuel L. Jackson, Nikole Kidman, Alan Rickman, Kurt Russell, Harrison Ford, Russell Crowe, Steve Martin, Michael Caine, Angelina Jolie, Mel Gibson, Michael Douglas, John Travolta, Sylvester Stallone, Tommy Lee Jones, Catherine Zeta-Jones, Antonio Banderas, Kim Basinger, Sam Neill, Gary Oldman, Clint Eastwood, Brad Pitt, Johnny Depp, Pierce Brosnan, Sean Connery, Bruce Willis, and Mullah Omar.

```
file:///c:/users/igor/documents
Donald Sutherland
Jennifer Lopez
Ray Liotta
Samuel L. Jackson
Nikole Kidman
Alan Rickman
Kurt Russell
Harrison Ford
Russell Crowe
Steve Martin
Michael Caine
Angelina Jolie
Mel Gibson
Michael Douglas
John Travolta
Sylvester Stallone
Tommy Lee Jones
Catherine Zeta-Jones
Antonio Banderas
Kim Basinger
Sam Neill
Gary Oldman
Clint Eastwood
Brad Pitt
Johnny Depp
Pierce Brosnan
Sean Connery
Bruce Willis
Mullah Omar
```

Можно ограничить диапазон получаемых значений при помощи оператора **where**:

```
var query = from c in db.Passengers
            where c.ID_psg == 16
            select c.name;
```

Результат работы программы:

A screenshot of a text file with the path 'file:///c:/users/igor/documents'. The file contains a single line with the name 'Mel Gibson'.

```
file:///c:/users/igor/documents
Mel Gibson
```

---

## ДОБАВЛЕНИЕ ДАННЫХ

Для добавления новой строки в таблицу **Passenger** нам необходимо создать новый объект класса **Passenger**:

```
var p = new Passenger {
    ID_psg = 38,
    name = "Vasily Pupkin"
};
```

Зарегистрировать его в очереди на обновление при помощи метода **InsertOnSubmit**:

```
db.Passengers.InsertOnSubmit(p);
```

Запустить процесс синхронизации изменений, вызвав метод **SubmitChanges**:

```
db.SubmitChanges();
```

Теперь можно вывести результат:

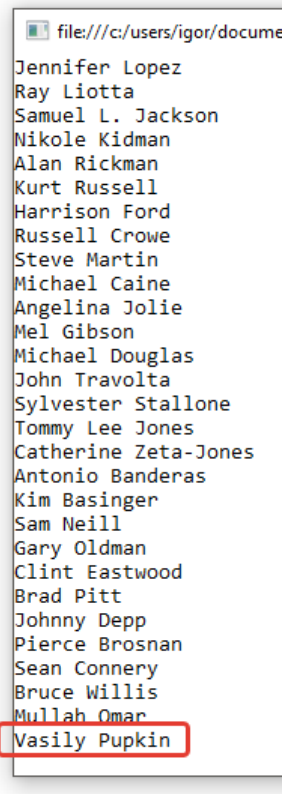
```

var query = from c in db.Passengers
            select c.name;

foreach (var q in query) {
    Console.WriteLine(q);
}
Console.ReadKey();

```

В таблице появилась новая запись:



## ИЗМЕНЕНИЕ ДАННЫХ

Для изменения данных в таблице достаточно изменить нужно поле в результатах запроса **query** и отправить результат обратно на сервер. К сожалению, такой запрос требует выборки всех столбцов из таблицы **Passenger**:

```

var query = from c in db.Passengers
            select c;

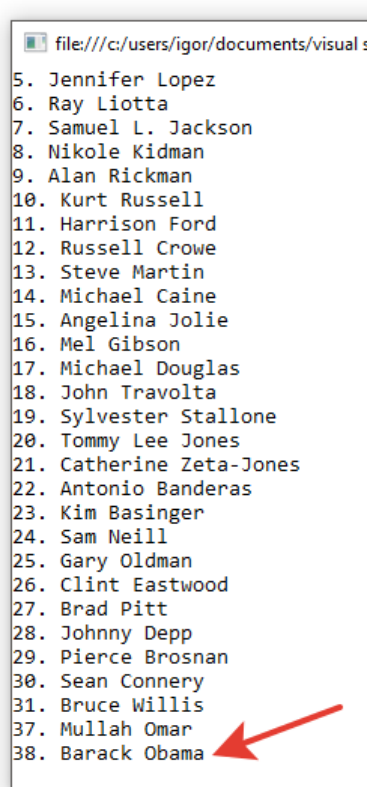
// обходим все строки из таблицы
foreach (var q in query) {
    if (q.ID_psg == 38) {
        q.name = "Barack Obama";
    }
}
// отправляем изменения на сервер
db.SubmitChanges();

// запрос на чтение данных
query = from c in db.Passengers
        select c;

// вывод содержимого таблицы
foreach (var q in query) {
    Console.WriteLine(q.ID_psg + ". " + q.name);
}
Console.ReadKey();

```

Результаты работы программы:



## УДАЛЕНИЕ ДАННЫХ

Для удаления данных из таблицы потребуется:

1. Составить запрос на выборку нужных данных.
2. Вызвать метод **DeleteOnSubmit** у объекта базы данных.
3. Запустить процесс синхронизации изменений при помощи метода **SubmitChanges**.

Пример:

```
var db = new AeroDataContext();

// выбираем из таблицы пассажира с индексом 38
var query = from c in db.Passengers
             where c.ID_psg == 38
             select c;

foreach (var q in query) {
    db.Passengers.DeleteOnSubmit(q); // ставим в очередь на удаление
}
db.SubmitChanges();

query = from c in db.Passengers
        select c;

foreach (var q in query) {
    Console.WriteLine(q.ID_psg + ". " + q.name);
}
Console.ReadKey();
```

После выполнения программы пассажир с индексом равным 38 должен пропасть из списка.

## ЗАДАНИЕ

Напишите программу, позволяющую:

1. Выводить содержимое таблицы **Trip** из базы данных **aero**. Используйте класс **DateTime** для работы с датой и временем.
  - a. Вывести всю таблицу целиком.
  - b. Вывести только строки с нужным идентификатором компании. Идентификатор компании вводится пользователем.
2. Удалять строки из таблицы **Trip** в интервале от A до B. Значения A и B вводятся пользователем.
3. Добавлять новые записи в таблицу **Trip**. Для времени отправления должно использоваться текущее значение системных часов (используйте свойство класса **DateTime.Now**). Время прибытия должно быть задано на час позже времени отправления.

Для работы с базой данных используйте библиотеку **LINQ**. Формат программы на усмотрение учащегося – консольное приложение или формы Windows.