Laboratorium 3: Entity Framework

Mateusz Surjak

Tydzień A Wt 9:35

1 Wprowadznie

Stworzyłem projekt type Console Application zgodnie z poleceniem.

2 Podpunkt b

Dodaj klasę Product z polami int ProductID, string Name, int UnitsInStock.

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

namespace ISurjakProductEF
{
    class Product
    {
        public int ProductID { get; set; }
        public string Name { get; set; }
        public int UnitsInStock { get; set; }
    }
}
```

3 Podpunkt c i d

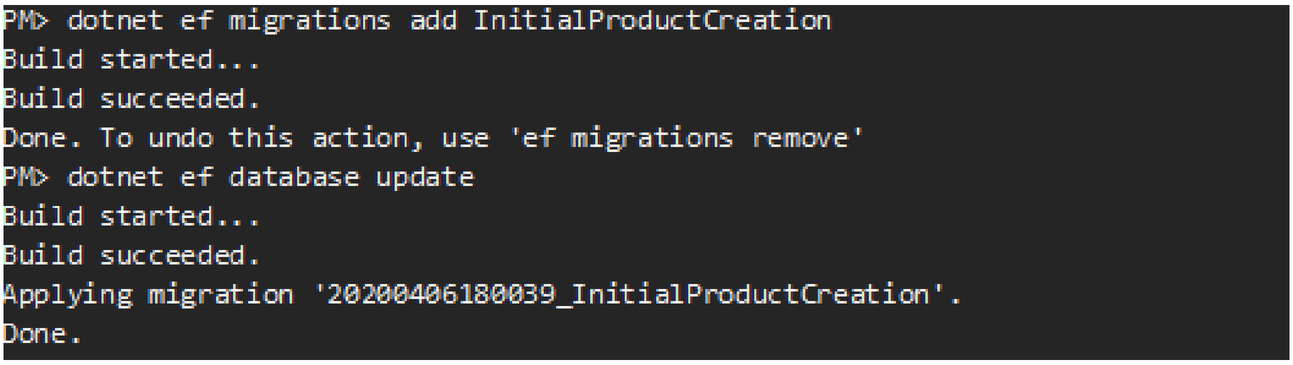
Stwórz klasę ProdContext dziedziczącą po DbContext. Dodaj do klasy kontekstowej zbiór (DbSet) produktów i nazwij go Products.

```
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using System.Text;

namespace ISurjakProductEF
{
    class ProdContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
    }
}
```

4 Podpunkt e

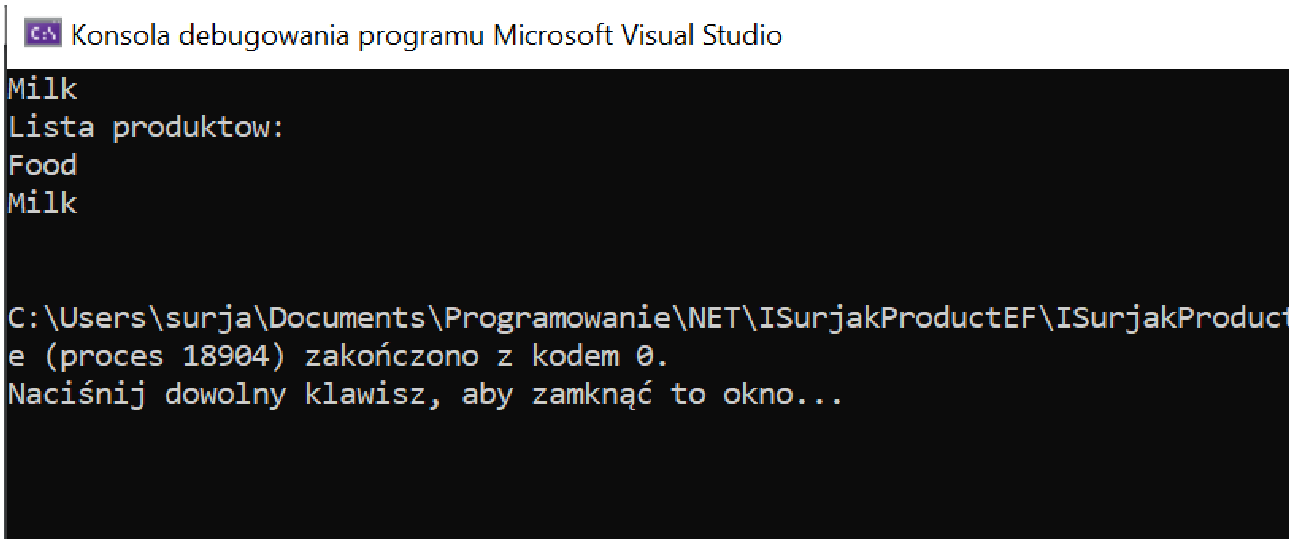
Dodanie niezbędnej konfiguracji do ProdContext.
Podpunkt e wykonałem zgodnie z poleceniami, dostałem oczekiwane błędy w trakcie wykonywania kolejnych kroków. Kod i screeny zamieszczone w tym podpunkcie są z juz działającej aplikacji po wykonaniu wszystkich kroków.

```
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using System.Text;
using Microsoft.EntityFrameworkCore.Sqlite;
namespace ISurjakProductEF
{
    class ProdContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder options)
    => options.UseSqlite("DataSource=Product.db");
    }
}
```



```
PM> dotnet ef migrations add InitialProductCreation
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
PM> dotnet ef database update
Build started...
Build succeeded.
Applying migration '20200406180039_InitialProductCreation'.
Done.
```

```
using System;
using System.Linq;
namespace ISurjakProductEF
{
    class Program
    {
        static void Main(string[] args)
        {
            string name = Console.ReadLine();
            Product product = new Product { Name = name };
            ProdContext prodContext = new ProdContext();
            prodContext.Products.Add(product);
            prodContext.SaveChanges();
            var data = (from p in prodContext.Products select p).ToList();
            Console.WriteLine("Lista produktow: ");
            foreach(var p in data)
            {
                Console.WriteLine(p.Name);
            }
            Console.WriteLine();
        }
    }
}
```

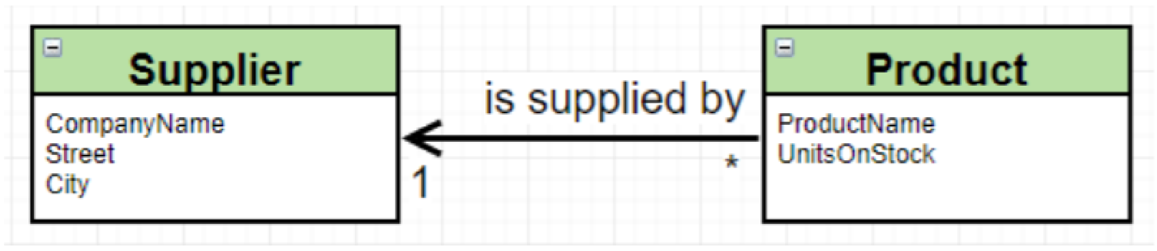


```
Milk
Lista produktow:
Food
Milk


C:\Users\surja\Documents\Programowanie\NET\ISurjakProductEF\ISurjakProduc
e (proces 18904) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Po tych zmianach mogłem już insertować do bazy kolejne produkty.

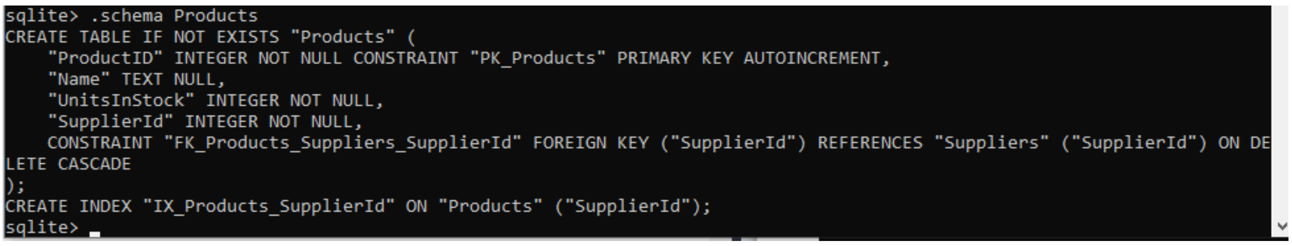
5 Zadanie II

Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej



```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Text;

namespace ISurjakProductEF
{
    class Supplier
    {   [Key]
        public int SupplierId { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ISurjakProductEF
{
    class Product
    {
        public int ProductID { get; set; }
        public string Name { get; set; }
        public int UnitsInStock { get; set; }

        [ForeignKey("Supplier")]
        public int SupplierId { get; set; }
        public Supplier supplier { get; set; }
    }
}
```



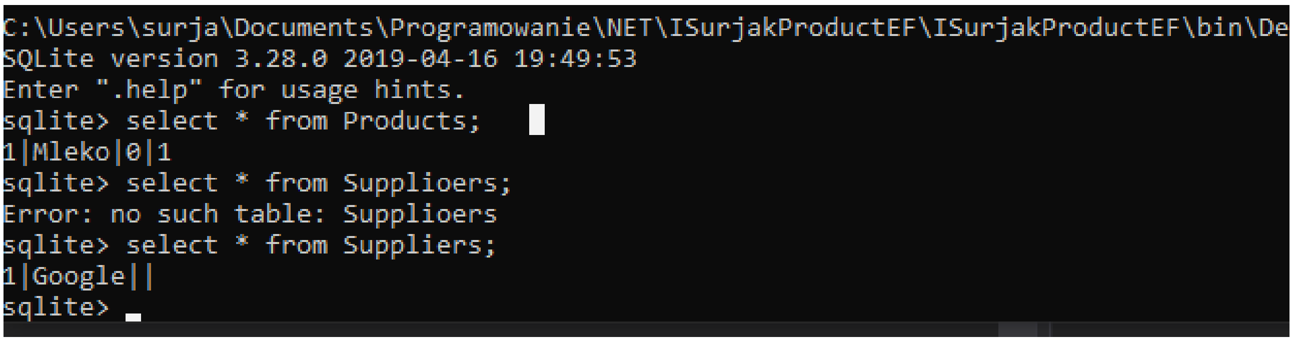
```
sqlite> .schema Products
CREATE TABLE IF NOT EXISTS "Products" (
    "ProductID" INTEGER NOT NULL CONSTRAINT "PK_Products" PRIMARY KEY AUTOINCREMENT,
    "Name" TEXT NULL,
    "UnitsInStock" INTEGER NOT NULL,
    "SupplierId" INTEGER NOT NULL,
    CONSTRAINT "FK_Products_Suppliers_SupplierId" FOREIGN KEY ("SupplierId") REFERENCES "Suppliers" ("SupplierId") ON DE
LETE CASCADE
);
CREATE INDEX "IX_Products_SupplierId" ON "Products" ("SupplierId");
sqlite> _
```

Schemat w bazie danych wyglądaq zgodnie z oczekiwaniami.

W celu możliwości dodania Dostawcó do Bazy danych zmodyfikowałem klasę ProdContext dodając kolejny DbSet.

```
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using System.Text;
using Microsoft.EntityFrameworkCore.Sqlite;
namespace ISurjakProductEF
{
    class ProdContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder options)
    => options.UseSqlite("Data Source=Product.db");

        public DbSet<Supplier> Suppliers { get; set; }
    }
}
```

```
using System;
using System.Linq;
namespace ISurjakProductEF
{
    class Program
    {
        static void Main(string[] args)
        {
            string name = Console.ReadLine();
            Product product = new Product { Name = name };
            ProdContext prodContext = new ProdContext();
            Supplier supplier = new Supplier { CompanyName = "Google"};
            prodContext.Suppliers.Add(supplier);
            prodContext.SaveChanges();

            var sup = (from p in prodContext.Suppliers select p).First();

            product.SupplierId = sup.SupplierId;
            prodContext.Products.Add(product);
            prodContext.SaveChanges();
            var data = (from p in prodContext.Products select p).ToList();
            Console.WriteLine("Lista produktow: ");
            foreach (var p in data)
            {
                Console.WriteLine(p.Name);
            }
            Console.WriteLine();
        }
    }
}
```



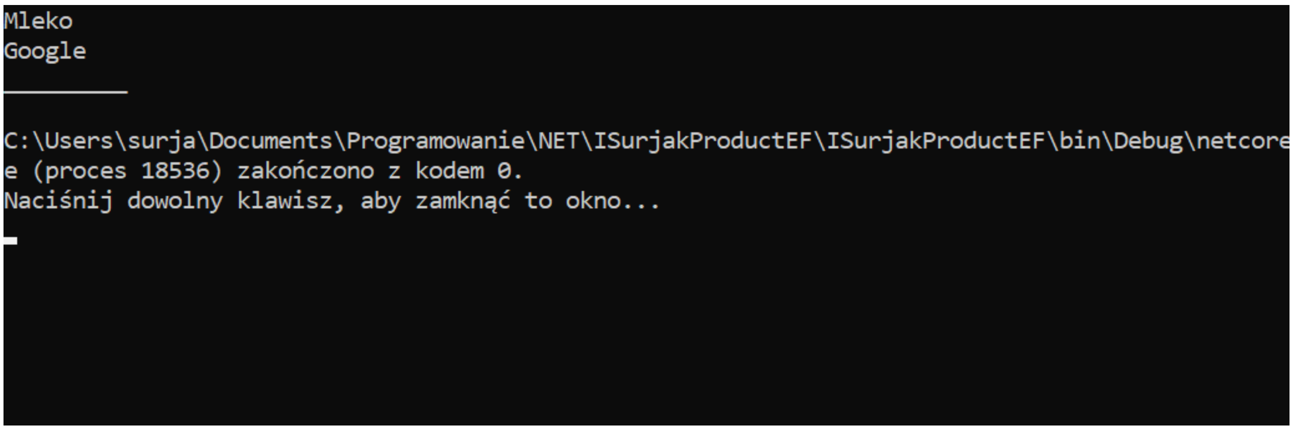
```
C:\Users\surja\Documents\Programowanie\NET\ISurjakProductEF\ISurjakProductEF\bin\De
SQLite version 3.28.0 2019-04-16 19:49:53
Enter ".help" for usage hints.
sqlite> select * from Products;
1|Mleko|0|1
sqlite> select * from Supplioers;
Error: no such table: Supplioers
sqlite> select * from Suppliers;
1|Google||
sqlite>
```

Jak widać baza danych została zamodelowana poprawnie, W tabeli Products mamy klucz obcy który wskazuje na odpowiedniego Dostawcę.

6 Podpunkt g

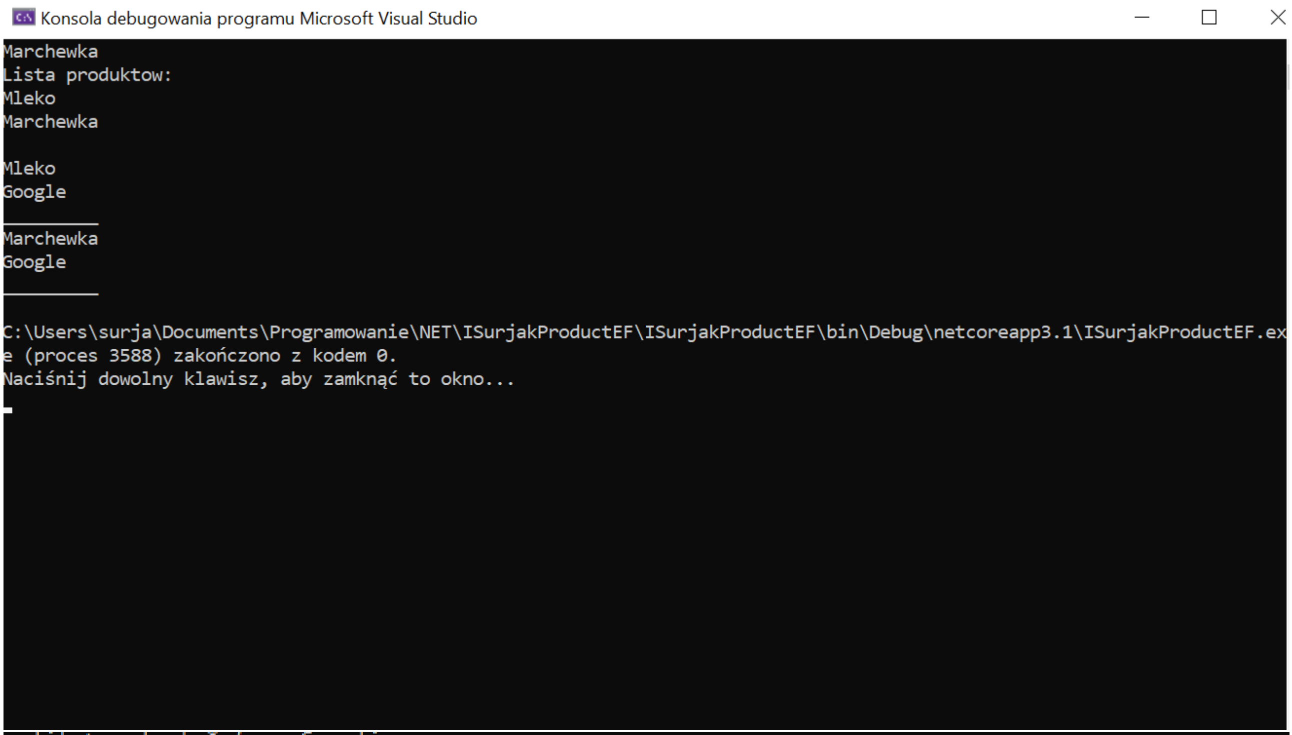
Wyświetl wszystkie produkty wraz z nazwą dostawcy

```
using System;
using System.Linq;
namespace ISurjakProductEF
{
    class Program
    {
        static void Main(string[] args)
        {
            ProdContext prodContext = new ProdContext();

            var data = (from p in prodContext.Products
                        join s in prodContext.Suppliers on p.SupplierId equals s
    .
                        SupplierId select new {prod=p.Name, dost=s.CompanyName })
    .ToList();
            foreach(var d in data)
            {
                Console.WriteLine(d.prod);
                Console.WriteLine(d.dost);
                Console.WriteLine("__________");

            }
        }
    }
}
```



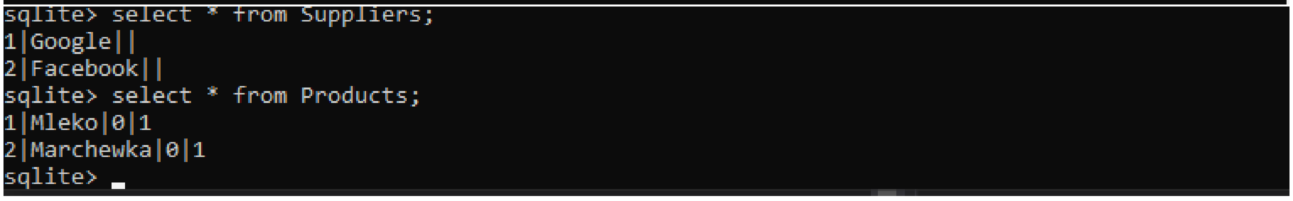
```
Mleko
Google
__________

C:\Users\surja\Documents\Programowanie\NET\ISurjakProductEF\ISurjakProductEF\bin\Debug\netcore
e (proces 18536) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Dodałem więcej danych w celu zobrazowania że aplikacja napewno działa.



Konsola debugowania programu Microsoft Visual Studio

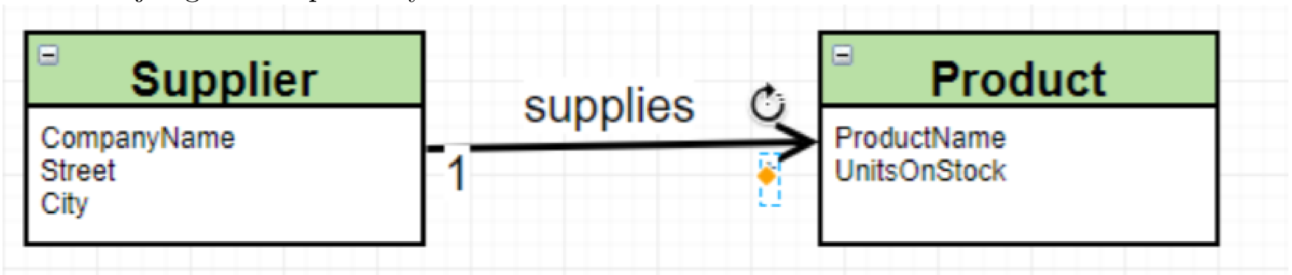
```
Marchewka
Lista produktow:
Mleko
Marchewka

Mleko
Google
_________
Marchewka
Google
_________

C:\Users\surja\Documents\Programowanie\NET\ISurjakProductEF\ISurjakProductEF\bin\Debug\netcoreapp3.1\ISurjakProductEF.ex
e (proces 3588) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```



```
sqlite> select * from Suppliers;
1|Google||
2|Facebook||
sqlite> select * from Products;
1|Mleko|0|1
2|Marchewka|0|1
sqlite>
```

7 Zadanie III

Odwróć relacje zgodnie z poniższym schematem

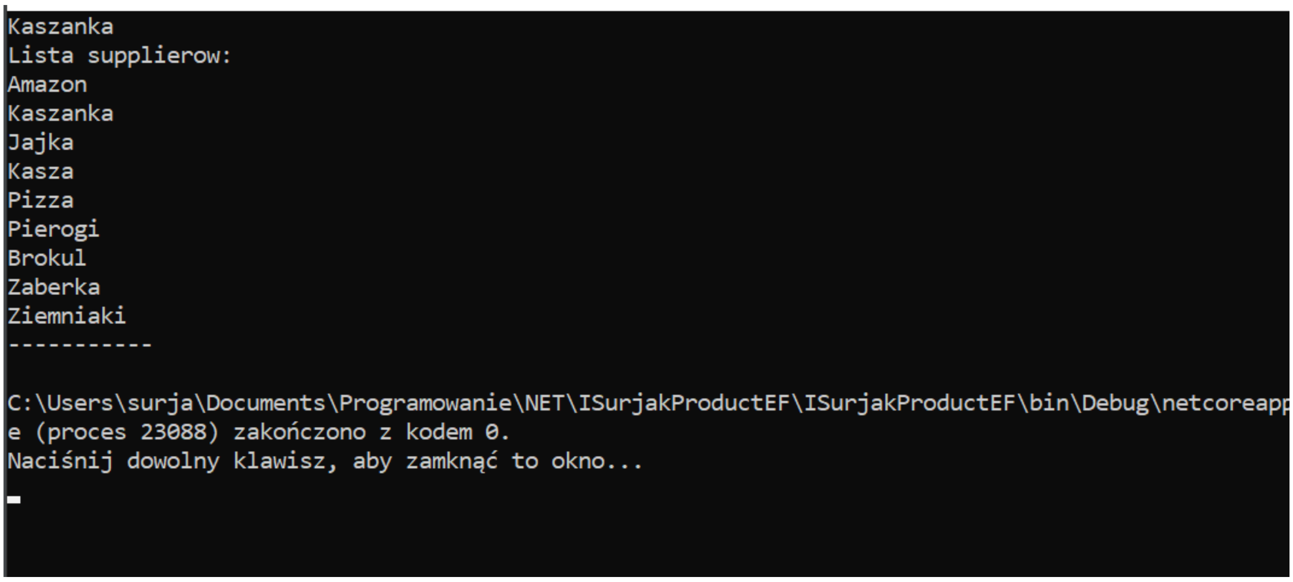


Dodałem do klasy Supplier listę produktów ktore dany dostawca dostarcza. Jednocześnie usunąłem polę Supplier z klasy Product.

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ISurjakProductEF
{
    class Supplier
    {
        public Supplier()
        {
            Products = new Collection<Product>();
        }

        [Key]
        public int SupplierId { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public ICollection<Product> Products { get; set; }

    }
}
```

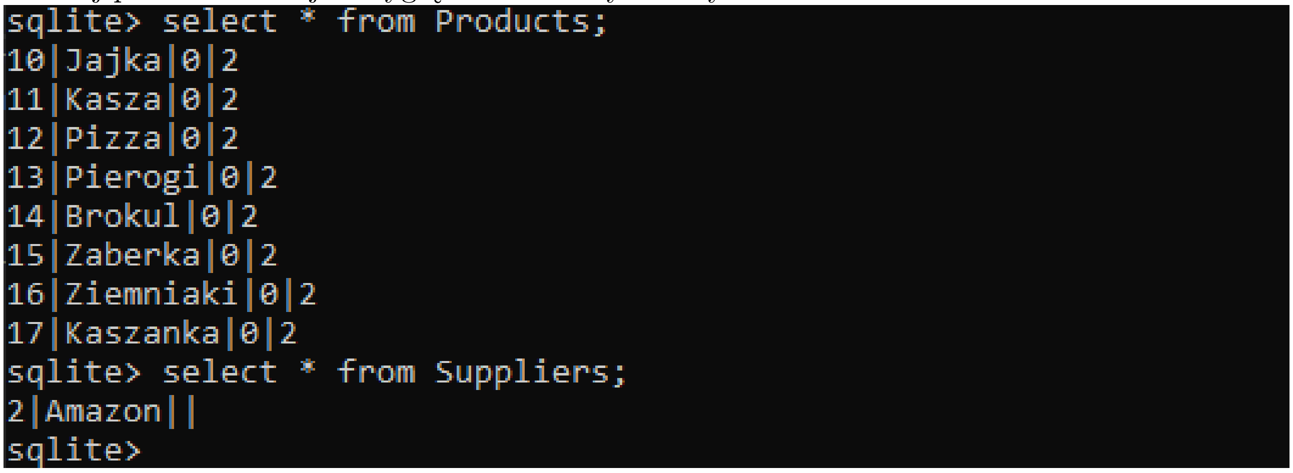
```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ISurjakProductEF
{
    class Product
    {
        public int ProductID { get; set; }
        public string Name { get; set; }
        public int UnitsInStock { get; set; }

    }
}
```

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.ObjectModel;
using System.Linq;
namespace ISurjakProductEF
{
    class Program
    {
        static void Main(string[] args)
        {
            ProdContext prodContext = new ProdContext();
            string name = Console.ReadLine();
            Product product = new Product { Name = name };
            prodContext.Products.Add(product);
            Supplier supplier = prodContext.Suppliers.FirstOrDefault(b => b.CompanyName == "Amazon");
            if (supplier == null)
            {
                supplier = new Supplier { CompanyName = "Amazon" };

                prodContext.Suppliers.Add(supplier);
            }
            supplier.Products.Add(product);
            prodContext.SaveChanges();


            var data1 = prodContext.Suppliers.Include(s => s.Products).ToList();
            Console.WriteLine("Lista supplierow: ");
            foreach (var s in data1)
            {
                Console.WriteLine(s.CompanyName);
                foreach (var p in s.Products)
                {
                    Console.WriteLine(p.Name);
                }
                Console.WriteLine("-------------------");
            }


        }
    }
}
```



```
Kaszanka
Lista supplierow:
Amazon
Kaszanka
Jajka
Kasza
Pizza
Pierogi
Brokul
Zaberka
Ziemniaki
-----------

C:\Users\surja\Documents\Programowanie\NET\ISurjakProductEF\ISurjakProductEF\bin\Debug\netcoreapp
e (proces 23088) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

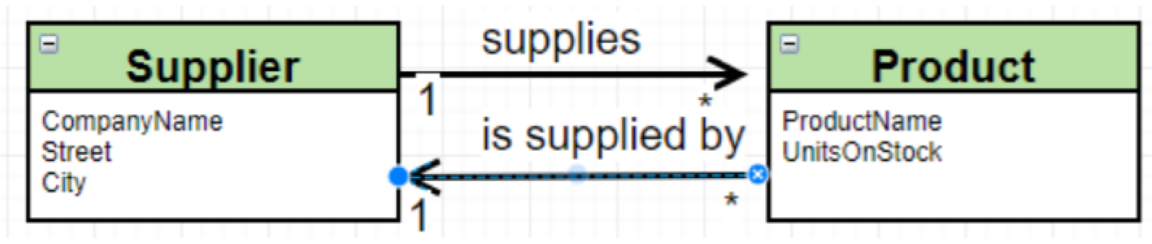
Aplikacja działa zgodnie z oczekiwaniami, Produkty dostawcy "Amazon" wyświetliły się poprawnie.
Poniżej przedstawiam jak wygląda baza danych w tym momencie.



```
sqlite> select * from Products;
10|Jajka|0|2
11|Kasza|0|2
12|Pizza|0|2
13|Pierogi|0|2
14|Brokul|0|2
15|Zaberka|0|2
16|Ziemniaki|0|2
17|Kaszanka|0|2
sqlite> select * from Suppliers;
2|Amazon||
sqlite>
```

8 Zadanie IV

Zamodeluj relacje dwustronną jak poniżej:

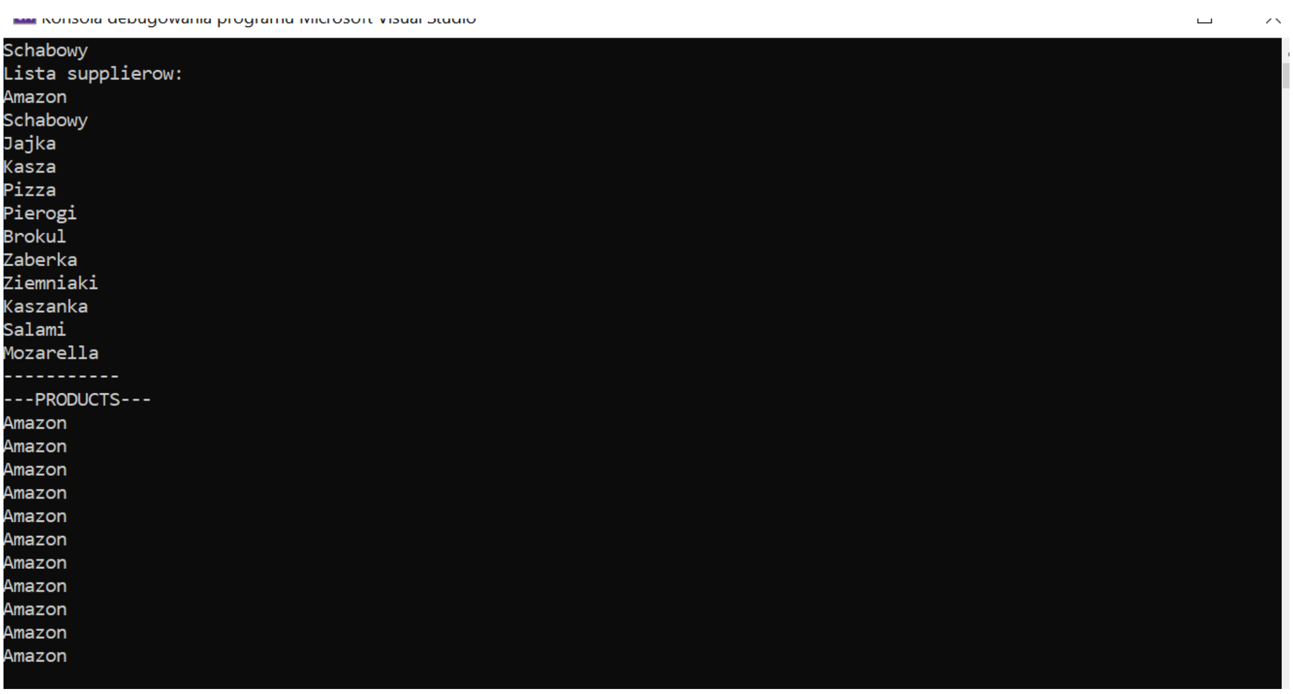


Do klasy Product dodałem Dostawcę aby można było się do niego bezpośrednio odwoływać.

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ISurjakProductEF
{
    class Supplier
    {
        public Supplier()
        {
            Products = new Collection<Product>();
        }

        public int SupplierId { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public virtual ICollection<Product> Products { get; set; }

    }
}
```

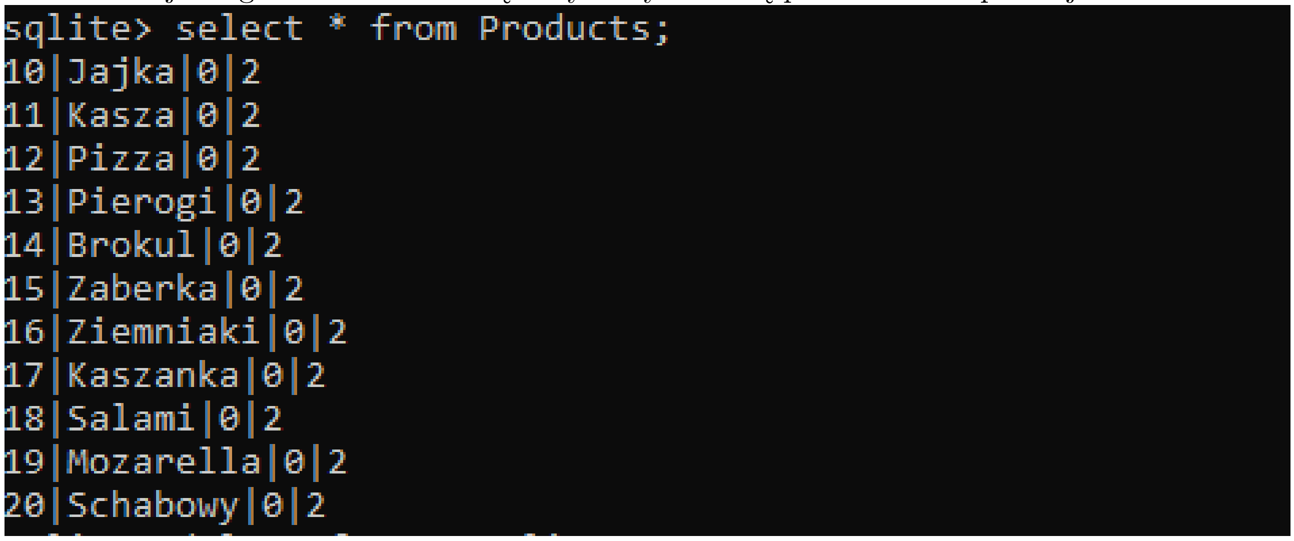
```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ISurjakProductEF
{
    class Product
    {

        public int ProductId { get; set; }
        public string Name { get; set; }
        public int UnitsInStock { get; set; }
        public Supplier Supplier { get; set; }
    }
}
```

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.ObjectModel;
using System.Linq;
namespace ISurjakProductEF
{
    class Program
    {
        static void Main(string[] args)
        {
            ProdContext prodContext = new ProdContext();
            string name = Console.ReadLine();
            Product product = new Product { Name = name };
            prodContext.Products.Add(product);
            Supplier supplier = prodContext.Suppliers.FirstOrDefault(b => b.CompanyName == "Amazon");
            if (supplier == null)
            {
                supplier = new Supplier { CompanyName = "Amazon" };

                prodContext.Suppliers.Add(supplier);
            }
            product.Supplier = supplier;
            supplier.Products.Add(product);
            prodContext.SaveChanges();


            var data1 = prodContext.Suppliers.Include(s => s.Products).ToList();
            Console.WriteLine("Lista supplierow: ");
            foreach (var s in data1)
            {
                Console.WriteLine(s.CompanyName);
                foreach (var p in s.Products)
                {
                    Console.WriteLine(p.Name);
                }
                Console.WriteLine("-------------");
            }
            Console.WriteLine("---PRODUCTS---");
            var data2 = prodContext.Products.Include(p => p.Supplier).ToList();
            foreach(var p in data2)
            {
                Console.WriteLine(p.Supplier.CompanyName);
            }


        }
    }
}
```



```
Schabowy
Lista supplierow:
Amazon
Schabowy
Jajka
Kasza
Pizza
Pierogi
Brokul
Zaberka
Ziemniaki
Kaszanka
Salami
Mozarella
-----------
---PRODUCTS---
Amazon
Amazon
Amazon
Amazon
Amazon
Amazon
Amazon
Amazon
Amazon
Amazon
Amazon
```

Można zauważyć że dostawca produktu wypisuje się dobrze. Dla każdego produktu wypisał się Amazon co jest zgodne z zawartością bazy danych którą przedstawiam poniżej.



```
sqlite> select * from Products;
10|Jajka|0|2
11|Kasza|0|2
12|Pizza|0|2
13|Pierogi|0|2
14|Brokul|0|2
15|Zaberka|0|2
16|Ziemniaki|0|2
17|Kaszanka|0|2
18|Salami|0|2
19|Mozarella|0|2
20|Schabowy|0|2
```

9 Zadanie V

Dodaj klase Category z property int CategoryID, String Name oraz listą produktow

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

namespace ISurjakProductEF
{
    class Category
    {
        public int CategoryId { get; set; }
        public string  Name { get; set; }
        public List<Product> Products { get; set; }
    }
}
```

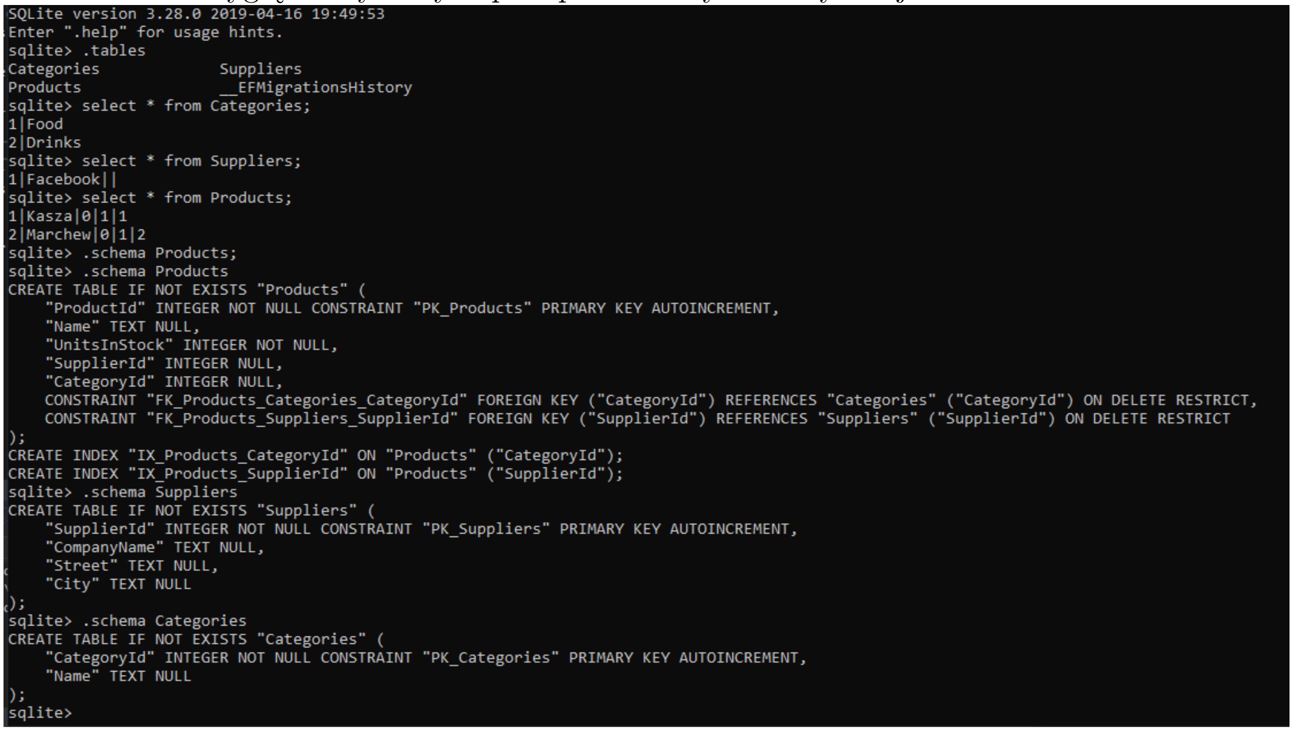
```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ISurjakProductEF
{
    class Product
    {
        public int ProductId { get; set; }
        public string Name { get; set; }
        public int UnitsInStock { get; set; }
        public Supplier Supplier { get; set; }
        public Category Category { get; set; }
    }
}
```

Dodałem Category do Contextu w celu insertu do bazy danych.

```
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using System.Text;
using Microsoft.EntityFrameworkCore.Sqlite;
namespace ISurjakProductEF
{
    class ProdContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder options)
    => options.UseSqlite("Data Source=Product.db");

        public DbSet<Supplier> Suppliers { get; set; }

        public DbSet<Category> Categories { get; set; }
    }
}
```

```
namespace ISurjakProductEF
{
    class Program
    {
        static void Main(string[] args)
        {
            Category category1 = new Category { Name = "Food" };
            Category category2 = new Category { Name = "Drinks" };

            ProdContext prodContext = new ProdContext();

            prodContext.Categories.Add(category1);
            prodContext.Categories.Add(category2);

            string companyName = "Facebook";
            string productName = "Kasza";

            Product product = new Product { Name = productName };
            product.Category = category1;
            Product product1 = new Product { Name = "Marchew" };
            product1.Category = category2;

            prodContext.Products.Add(product);

            Supplier supplier = prodContext.Suppliers.FirstOrDefault(b => b.CompanyName == companyName);
            if (supplier == null)
            {
                supplier = new Supplier { CompanyName = companyName };
                prodContext.Suppliers.Add(supplier);
            }
            product.Supplier = supplier;
            product1.Supplier = supplier;
            supplier.Products.Add(product);
            supplier.Products.Add(product1);
            prodContext.SaveChanges();

            var data1 = prodContext.Suppliers.Include(s => s.Products).ToList();
            Console.WriteLine("Lista supplierow: ");
            foreach (var s in data1)
            {
                Console.WriteLine(s.CompanyName);
                Console.WriteLine("Produkty dla firmy: ");
                foreach (var p in s.Products)
                {
                    Console.WriteLine(p.Name);
                }
                Console.WriteLine("------------");
            }

            Console.WriteLine("----PRODUCTS----");
            var data2 = prodContext.Products.Include(p => p.Supplier).ToList();
            foreach(var p in data2)
            {
                Console.WriteLine(p.Supplier.CompanyName);
            }
            var productsFromCatergory = prodContext.Categories.Include(c => c.Products).Where(c => c.Name == "Warzywa");
            foreach (var c in productsFromCatergory)
            {
                    foreach (var p in c.Products)
                    {
                        Console.WriteLine(p.Name);
                    }
                }

                var categoryFromProd = prodContext.Products.Where(p => p.Name=="Kiwi
    ").Include(c => c.Category).FirstOrDefault();
                Console.WriteLine(categoryFromProd.Category.Name);
            }
        }
}
```

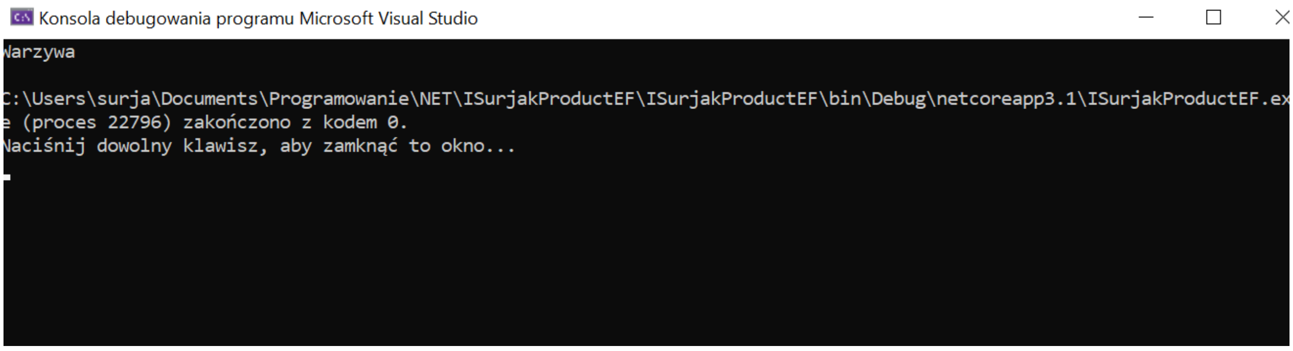
Przedstawiam wygląd bazy danych po wprowadzonych modyfikacjach.



```
SQLite version 3.28.0 2019-04-16 19:49:53
Enter ".help" for usage hints.
sqlite> .tables
Categories             Suppliers
Products               __EFMigrationsHistory
sqlite> select * from Categories;
1|Food
2|Drinks
sqlite> select * from Suppliers;
1|Facebook||
sqlite> select * from Products;
1|Kasza|0|1|1
2|Marchew|0|1|2
sqlite> .schema Products;
sqlite> .schema Products
CREATE TABLE IF NOT EXISTS "Products" (
    "ProductId" INTEGER NOT NULL CONSTRAINT "PK_Products" PRIMARY KEY AUTOINCREMENT,
    "Name" TEXT NULL,
    "UnitsInStock" INTEGER NOT NULL,
    "SupplierId" INTEGER NULL,
    "CategoryId" INTEGER NULL,
    CONSTRAINT "FK_Products_Categories_CategoryId" FOREIGN KEY ("CategoryId") REFERENCES "Categories" ("CategoryId") ON DELETE RESTRICT,
    CONSTRAINT "FK_Products_Suppliers_SupplierId" FOREIGN KEY ("SupplierId") REFERENCES "Suppliers" ("SupplierId") ON DELETE RESTRICT
);
CREATE INDEX "IX_Products_CategoryId" ON "Products" ("CategoryId");
CREATE INDEX "IX_Products_SupplierId" ON "Products" ("SupplierId");
sqlite> .schema Suppliers
CREATE TABLE IF NOT EXISTS "Suppliers" (
    "SupplierId" INTEGER NOT NULL CONSTRAINT "PK_Suppliers" PRIMARY KEY AUTOINCREMENT,
    "CompanyName" TEXT NULL,
    "Street" TEXT NULL,
    "City" TEXT NULL
);
sqlite> .schema Categories
CREATE TABLE IF NOT EXISTS "Categories" (
    "CategoryId" INTEGER NOT NULL CONSTRAINT "PK_Categories" PRIMARY KEY AUTOINCREMENT,
    "Name" TEXT NULL
);
sqlite>
```

9.1 Wydobądź produkty z wybranej kategorii

Kod do tego podpunktu znajduje się mpowyżej w klasie Program

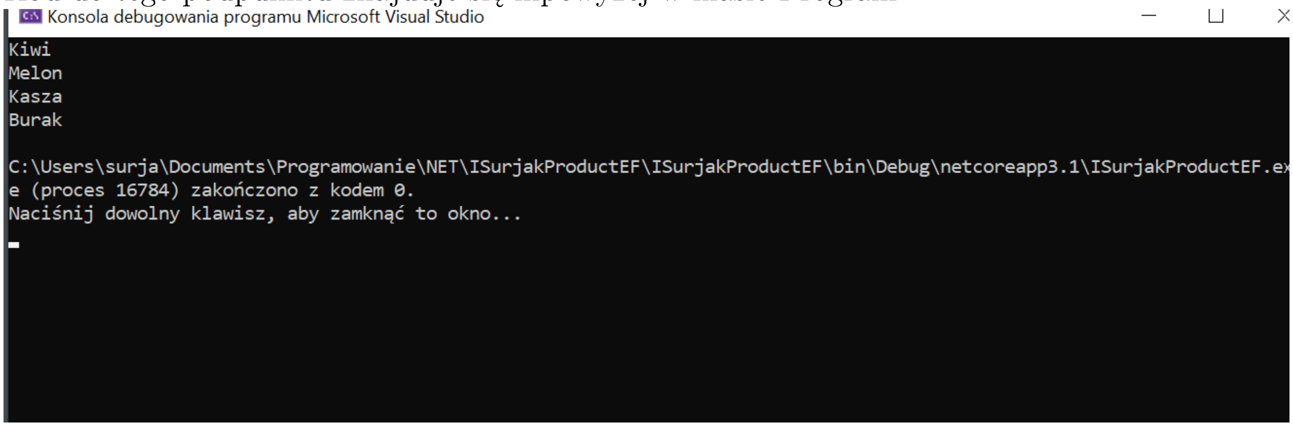


Konsola debugowania programu Microsoft Visual Studio

```
Warzywa

C:\Users\surja\Documents\Programowanie\NET\ISurjakProductEF\ISurjakProductEF\bin\Debug\netcoreapp3.1\ISurjakProductEF.ex
e (proces 22796) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

9.2 Wydobądź kategorię do której należy wybrany produkt

Kod do tego podpunktu znajduje się mpowyżej w klasie Program

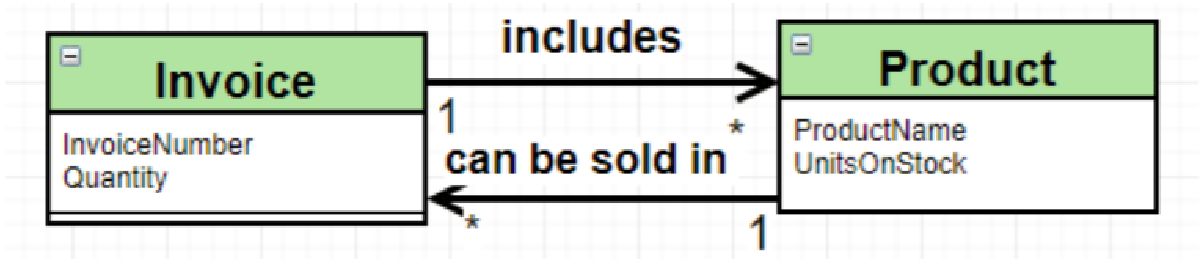


Konsola debugowania programu Microsoft Visual Studio

```
Kiwi
Melon
Kasza
Burak

C:\Users\surja\Documents\Programowanie\NET\ISurjakProductEF\ISurjakProductEF\bin\Debug\netcoreapp3.1\ISurjakProductEF.ex
e (proces 16784) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

10 Zadanie VI

Zamodeluj relacje wiele-do-wielu, jak poniżej:
W celu wykonania tego podpunktu konieczne było stworzenie nowego obiektu InvoiceProduct który przechowuje relacje pomiędzy Invoice a Product. W klasach Product i Invoice stworzyłem kolekcję obiektów InvoiceProducts w celu wyrażenia relacji many-to-many.



```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ISurjakProductEF
{
    class Supplier
    {
        public Supplier()
        {
            Products = new Collection<Product>();
        }
        public int SupplierId { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public virtual ICollection<Product> Products { get; set; }

    }
}
```

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Text;

namespace ISurjakProductEF
{
    class Category
    {
        public Category()
        {
            Products = new Collection<Product>();
        }
        public int CategoryId { get; set; }
        public string  Name { get; set; }
        public ICollection<Product> Products { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Text;

namespace ISurjakProductEF
{
    class InvoiceProduct
    {

        public int ProductId { get; set; }
        public Product Product { get; set; }
        public int InvoiceId { get; set; }
        public Invoice Invoice { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Text;

namespace ISurjakProductEF
{
    class Invoice
    {
        public Invoice()
        {
            InvoiceProducts = new Collection<InvoiceProduct>();
        }
        public int InvoiceId { get; set; }

        public int InvoiceNumber { get; set; }

        public int Quantity { get; set; }

        public ICollection<InvoiceProduct> InvoiceProducts { get; set; }
    }
}
```

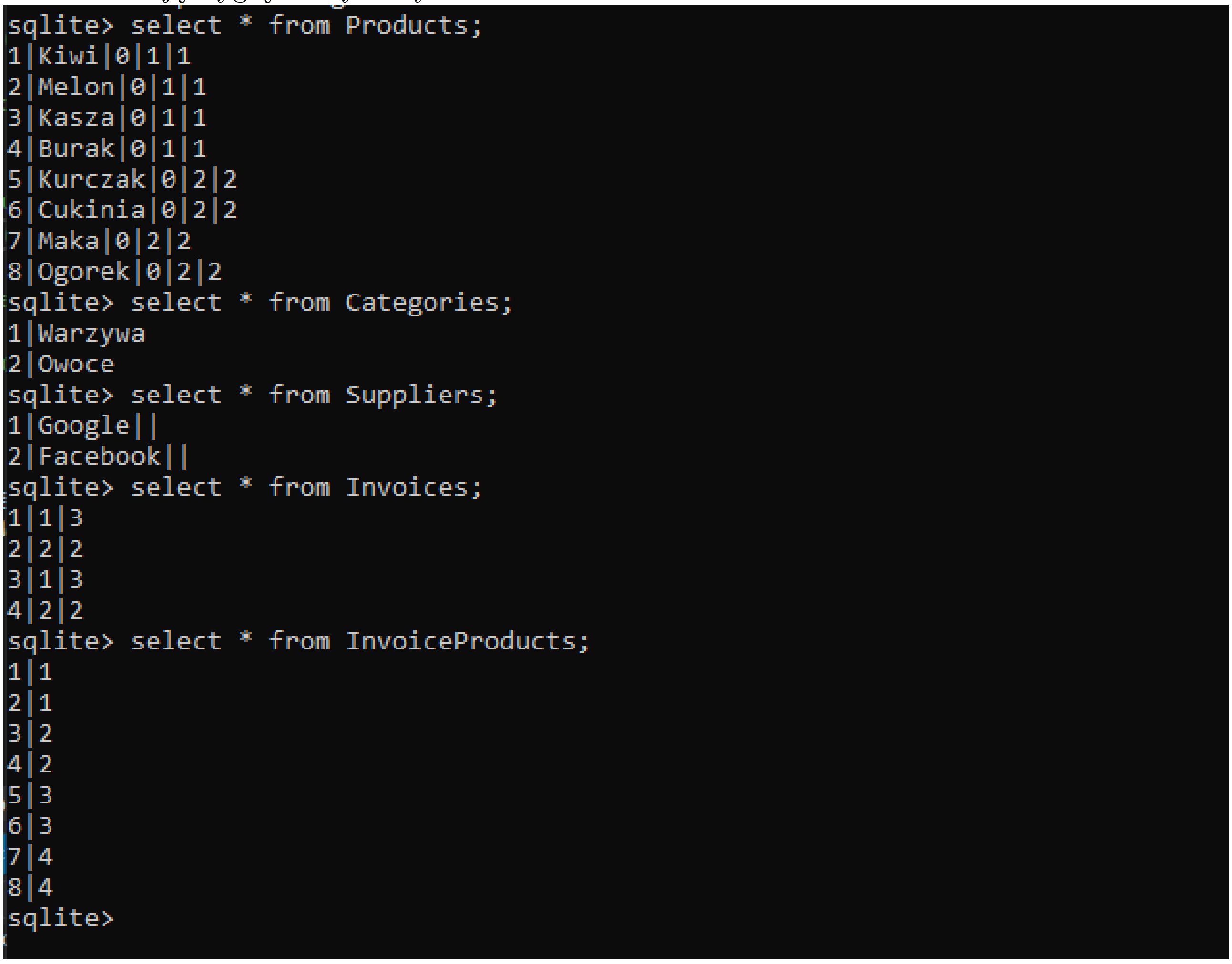
```csharp
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ISurjakProductEF
{
    class Product
    {
        public Product()
        {
            InvoiceProducts = new Collection<InvoiceProduct>();
        }

        public int ProductId { get; set; }
        public string Name { get; set; }
        public int UnitsInStock { get; set; }
        public Supplier Supplier { get; set; }

        public Category Category { get; set; }
        public ICollection<InvoiceProduct> InvoiceProducts{ get; set; }
    }
}
```

Oprócz tego dodałem brakujące kolekcje do Contextu.

```csharp
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using System.Text;
using Microsoft.EntityFrameworkCore.Sqlite;
namespace ISurjakProductEF
{
    class ProdContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder options)
    => options.UseSqlite("Data Source=Product.db");

        public DbSet<Supplier> Suppliers { get; set; }

        public DbSet<Category> Categories { get; set; }

        public DbSet<Invoice> Invoices { get; set; }

        public DbSet<InvoiceProduct> invoiceProducts { get; set; }

        protected override void
         OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<InvoiceProduct>()
                .HasKey(x => new { x.ProductId, x.InvoiceId});
        }


    }
}
```

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.ObjectModel;
using System.Linq;
namespace ISurjakProductEF
{
    class Program
    {
        static void Main(string[] args)
        {

            ProdContext prodContext = new ProdContext();
            Product product1 = new Product { Name = "Kurczak" };
            Product product2 = new Product { Name = "Cukinia" };
            Product product3 = new Product { Name = "Maka" };
            Product product4 = new Product { Name = "Ogorek" };
            Category category = new Category { Name = "Owoce" };
            category.Products.Add(product1);
            category.Products.Add(product2);
            category.Products.Add(product3);
            category.Products.Add(product4);
            product1.Category = category;
            product2.Category = category;
            product3.Category = category;
            product4.Category = category;
            Supplier supplier = new Supplier { CompanyName = "Facebook" };
            supplier.Products.Add(product1);
            supplier.Products.Add(product2);
            supplier.Products.Add(product3);
            supplier.Products.Add(product4);
            product1.Supplier = supplier;
            product2.Supplier = supplier;
            product3.Supplier = supplier;
            product4.Supplier = supplier;
            prodContext.Categories.Add(category);
            prodContext.Products.Add(product1);
            prodContext.Products.Add(product2);
            prodContext.Products.Add(product3);
            prodContext.Products.Add(product4);
            prodContext.Suppliers.Add(supplier);

            Invoice invoice1 = new Invoice { InvoiceNumber = 1, Quantity = 3 };
            Invoice invoice2 = new Invoice { InvoiceNumber = 2, Quantity = 2 };
            prodContext.Invoices.Add(invoice1);
            prodContext.Invoices.Add(invoice2);

            InvoiceProduct invoiceProduct1 = new InvoiceProduct { Invoice =
    invoice1, Product = product1 };
            invoice1.InvoiceProducts.Add(invoiceProduct1);
            InvoiceProduct invoiceProduct2 = new InvoiceProduct { Invoice =
    invoice1, Product = product2 };
            invoice1.InvoiceProducts.Add(invoiceProduct2);
            InvoiceProduct invoiceProduct3 = new InvoiceProduct { Invoice =
    invoice2, Product = product3 };
            invoice2.InvoiceProducts.Add(invoiceProduct3);
            InvoiceProduct invoiceProduct4 = new InvoiceProduct { Invoice =
    invoice2, Product = product4 };
            invoice2.InvoiceProducts.Add(invoiceProduct4);
            prodContext.invoiceProducts.Add(invoiceProduct1);
            prodContext.invoiceProducts.Add(invoiceProduct2);
            prodContext.invoiceProducts.Add(invoiceProduct3);
            prodContext.invoiceProducts.Add(invoiceProduct4);

            product1.InvoiceProducts.Add(invoiceProduct1);
            product2.InvoiceProducts.Add(invoiceProduct2);
            product3.InvoiceProducts.Add(invoiceProduct3);
            product4.InvoiceProducts.Add(invoiceProduct4);

            prodContext.SaveChanges();

            var products = prodContext.invoiceProducts.Include(d => d.Product).Where(d => d.InvoiceId == 3).Select(d => d.Product.Name).ToList();
            var invoices = prodContext.invoiceProducts.Include(d => d.Invoice).Where(p => p.ProductId == 1).Select(d => d.Invoice.InvoiceNumber);

            foreach (var p in products)
            {
                Console.WriteLine(p);
            }

            foreach (var p in invoices)
            {
                Console.WriteLine(p);
            }

            var productsFromCatergory = prodContext.Categories.Include(c => c.Products).Where(c => c.Name == "Warzywa");
            foreach (var c in productsFromCatergory)
            {
                foreach (var p in c.Products)
                {
                    Console.WriteLine(p.Name);
                }
            }

            var categoryFromProd = prodContext.Products.Where(p => p.Name=="Kiwi").Include(c => c.Category).FirstOrDefault();
            Console.WriteLine(categoryFromProd.Category.Name);
        }
    }
}
```

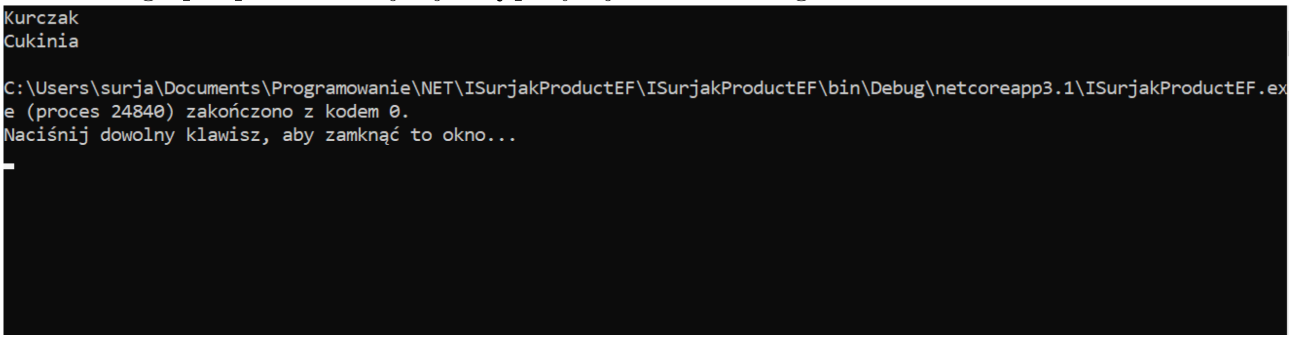
Prezentuję wygląd bazy danych



```
sqlite> select * from Products;
1|Kiwi|0|1|1
2|Melon|0|1|1
3|Kasza|0|1|1
4|Burak|0|1|1
5|Kurczak|0|2|2
6|Cukinia|0|2|2
7|Maka|0|2|2
8|Ogorek|0|2|2
sqlite> select * from Categories;
1|Warzywa
2|Owoce
sqlite> select * from Suppliers;
1|Google||
2|Facebook||
sqlite> select * from Invoices;
1|1|3
2|2|2
3|1|3
4|2|2
sqlite> select * from InvoiceProducts;
1|1
2|1
3|2
4|2
5|3
6|3
7|4
8|4
sqlite>
```

10.1 Pokaż produkty sprzedane w ramach wybranej faktury/transakcji

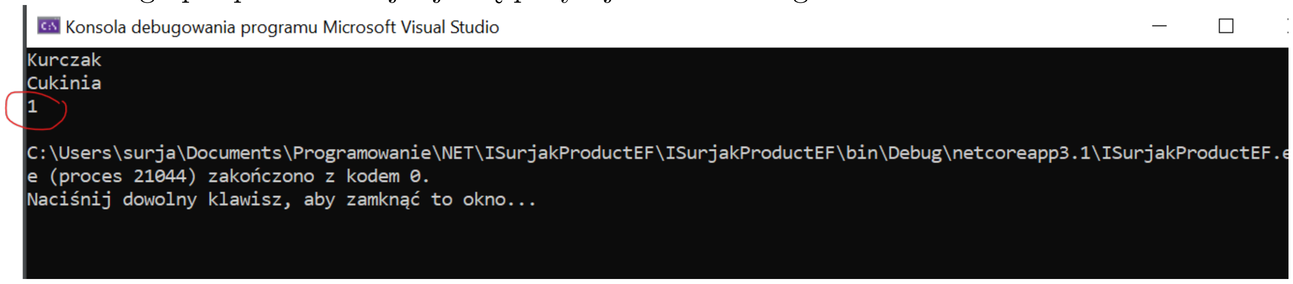
Kod do tego podpunktu znajduje się pozyżej w klasie Program w linii 67



```
Kurczak
Cukinia

C:\Users\surja\Documents\Programowanie\NET\ISurjakProductEF\ISurjakProductEF\bin\Debug\netcoreapp3.1\ISurjakProductEF.ex
e (proces 24840) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

10.2 Pokaż faktury w ramach których był sprzedany wybrany produkt

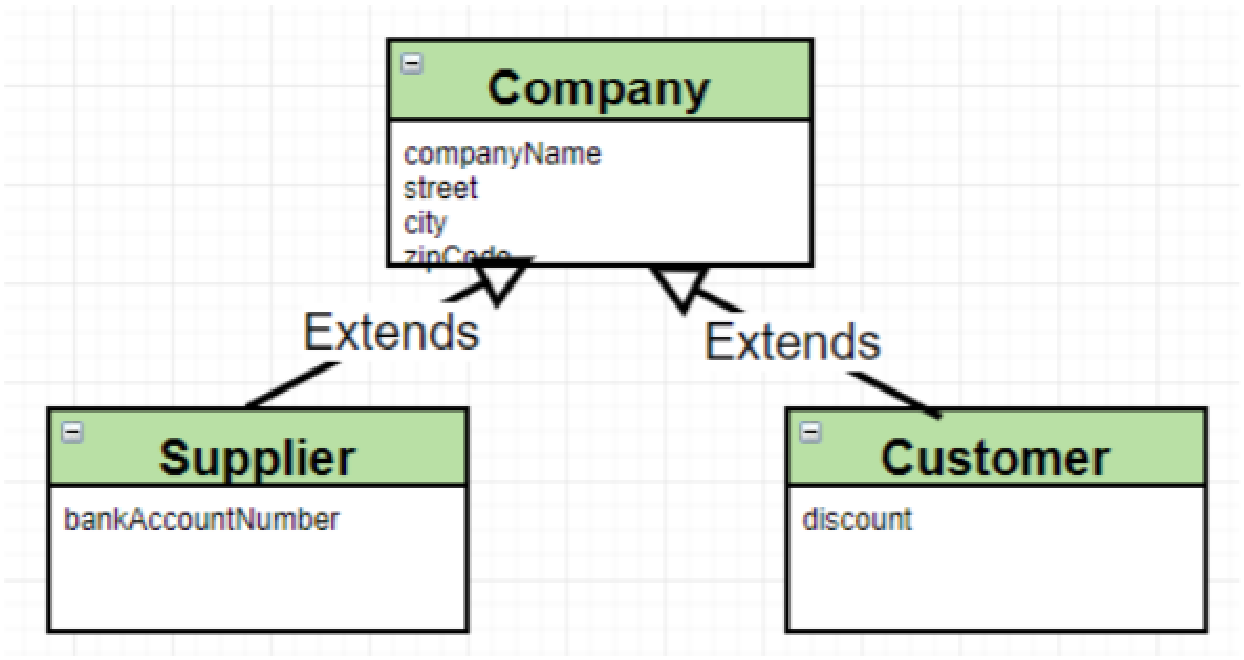
Kod do tego podpunktu znajduje się pozyżej w klasie Program w linii 68


Konsola debugowania programu Microsoft Visual Studio

```
Kurczak
Cukinia
1

C:\Users\surja\Documents\Programowanie\NET\ISurjakProductEF\ISurjakProductEF\bin\Debug\netcoreapp3.1\ISurjakProductEF.e
e (proces 21044) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

11 Zadanie VII

Wprowadź do modelu następującą hierarchie:



```csharp
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ISurjakProductEF
{
    class Supplier : Company
    {
        public Supplier()
        {
            Products = new Collection<Product>();
        }
        public int BankAccountNumber { get; set; }
        public virtual ICollection<Product> Products { get; set; }
    }
}
```

```csharp
using System;
using System.Collections.Generic;
using System.Text;

namespace ISurjakProductEF
{
    class Customer : Company
    {
        public int Discount { get; set; }
    }
}
```

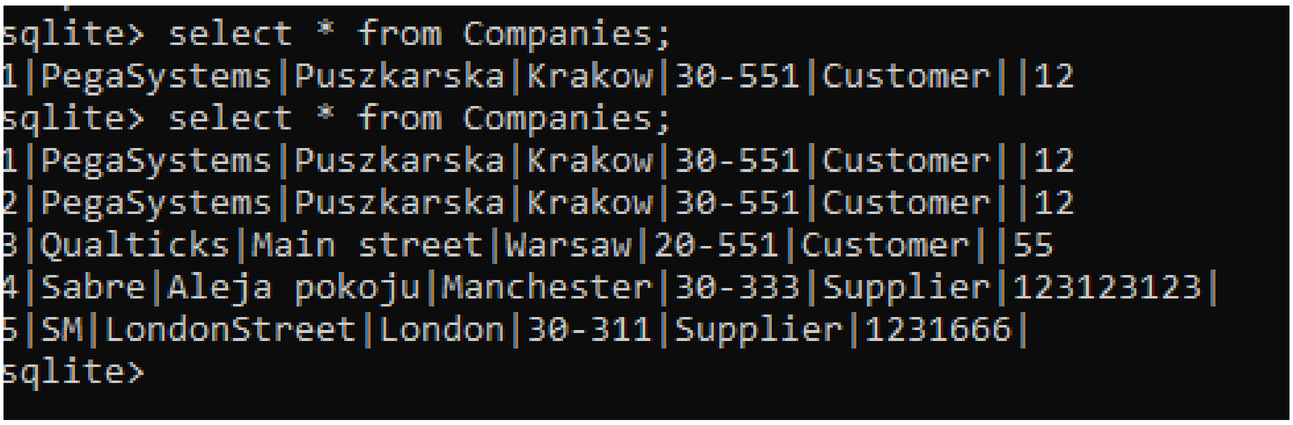
```csharp
using System;
using System.Collections.Generic;
using System.Text;

namespace ISurjakProductEF
{
    class Company
    {
        public int CompanyId { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }

        public string ZipCode { get; set; }
    }
}
```

11.1 TablePerHierarchy

Do wykonanania TablePerHierarchy wprowadziłem zmiany w Contexie w metodzie OnModel-Creating.

```
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using System.Text;
using Microsoft.EntityFrameworkCore.Sqlite;
namespace ISurjakProductEF
{
    class ProdContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder options)
    => options.UseSqlite("Data Source=Product.db");

        public DbSet<Company> Companies { get; set; }


        public DbSet<Category> Categories { get; set; }


        public DbSet<Invoice> Invoices { get; set; }

        public DbSet<InvoiceProduct> invoiceProducts { get; set; }

        protected override void
         OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<InvoiceProduct>()
                .HasKey(x => new { x.ProductId, x.InvoiceId});
            modelBuilder.Entity<Customer>();
            modelBuilder.Entity<Supplier>();
        }


    }
}
```

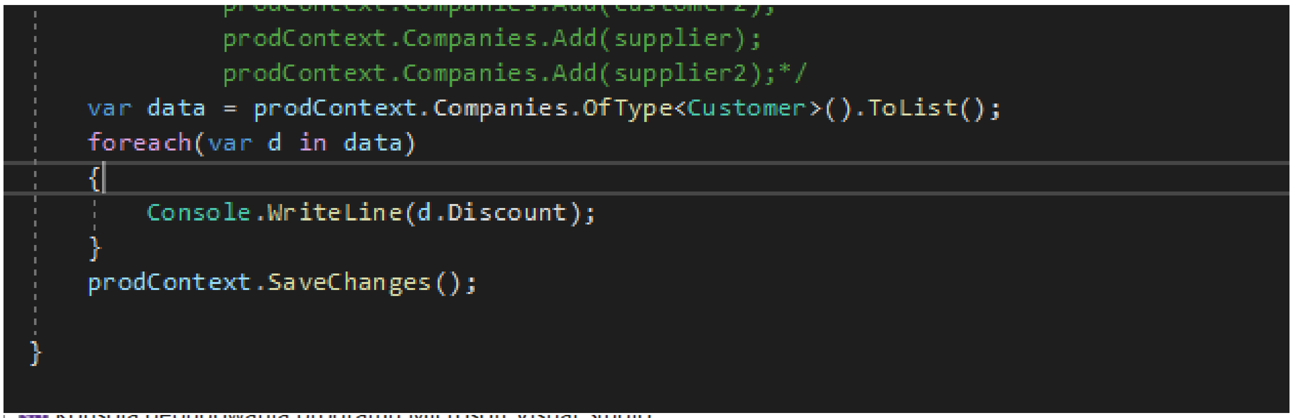
```

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.ObjectModel;
using System.Linq;
namespace ISurjakProductEF
{
    class Program
    {
        static void Main(string[] args)
        {

            ProdContext prodContext = new ProdContext();
            Customer customer = new Customer
            {
                City = "Krakow",
                CompanyName = "PegaSystems",
                Street = "Puszkarska",
                ZipCode = "30-551",
                Discount = 12,
            };
            Supplier supplier = new Supplier
            {
                City = "Manchester",
                CompanyName = "Sabre",
                Street = "Aleja pokoju",
                ZipCode = "30-333",
                BankAccountNumber = 123123123

            };
            Customer customer2 = new Customer
            {
                City = "Warsaw",
                CompanyName = "Qualticks",
                Street = "Main street",
                ZipCode = "20-551",
                Discount = 55

            };
            Supplier supplier2 = new Supplier
            {
                City = "London",
                CompanyName = "SM",
                Street = "LondonStreet",
                ZipCode = "30-311",
                BankAccountNumber = 1231666

            };
            prodContext.Companies.Add(customer);
            prodContext.Companies.Add(customer2);
            prodContext.Companies.Add(supplier);
            prodContext.Companies.Add(supplier2);

            prodContext.SaveChanges();

        }
    }
}
```



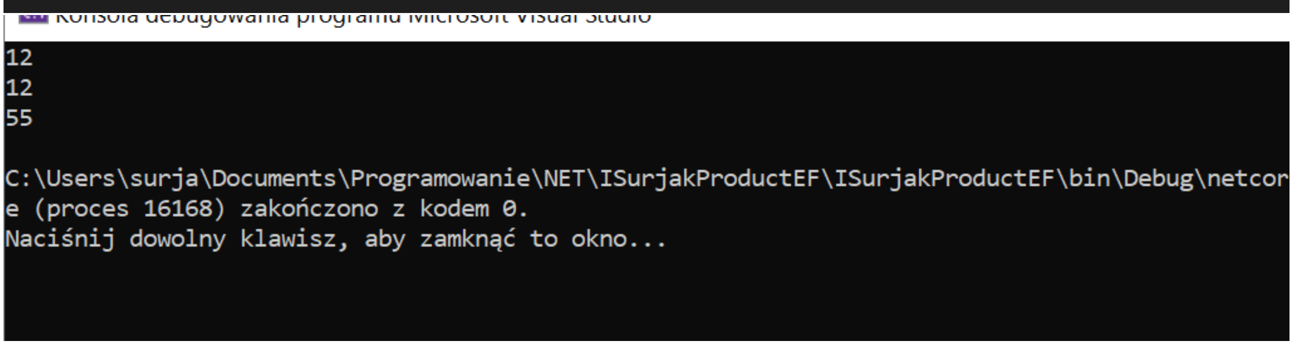
```
sqlite> select * from Companies;
1|PegaSystems|Puszkarska|Krakow|30-551|Customer||12
sqlite> select * from Companies;
1|PegaSystems|Puszkarska|Krakow|30-551|Customer||12
2|PegaSystems|Puszkarska|Krakow|30-551|Customer||12
3|Qualticks|Main street|Warsaw|20-551|Customer||55
4|Sabre|Aleja pokoju|Manchester|30-333|Supplier|123123123|
5|SM|LondonStreet|London|30-311|Supplier|1231666|
sqlite>
```

11.2 Pobranie danych

Pobrałem wszystkich Customerów w celu zbadania działania metody OfType, metoda wybrała tylko pola powiązane z Customerem z Dazy danych pomiojając obiekty które są typu Supplier. Wyświetliłem property które posiadają tylko Customerzy.



```
                prodContext.Companies.Add(customer2);
                prodContext.Companies.Add(supplier);
                prodContext.Companies.Add(supplier2);*/
        var data = prodContext.Companies.OfType<Customer>().ToList();
        foreach(var d in data)
        {
            Console.WriteLine(d.Discount);
        }
        prodContext.SaveChanges();

    }
```



Konsola debugowania programu Microsoft Visual Studio

```
12
12
55

C:\Users\surja\Documents\Programowanie\NET\ISurjakProductEF\ISurjakProductEF\bin\Debug\netcor
e (proces 16168) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

11.3 TablePerType

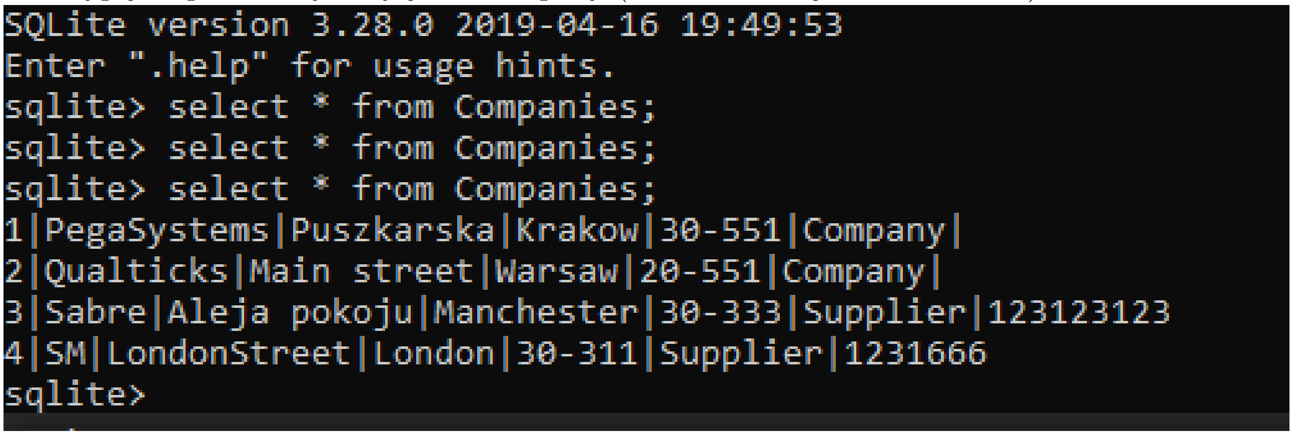
Starałem się zamodelować bazę zgodnie z dokumentacją. Niestety jak się okazało w wersji ponad 3.0 Entity Framework nie da się skorzystać z TPT. Pokażę co otrzymałem po próbach insertu do bazy.

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ISurjakProductEF
{
    [Table("Customers")]
    class Customer : Company
    {
        public int Discount { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ISurjakProductEF
{
    [Table("Suppliers")]
    class Supplier : Company
    {
        public Supplier()
        {
            Products = new Collection<Product>();
        }

        public int BankAccountNumber { get; set; }
        public virtual ICollection<Product> Products { get; set; }

    }
}
```

```
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using System.Text;
using Microsoft.EntityFrameworkCore.Sqlite;
namespace ISurjakProductEF
{
    class ProdContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder options)
    => options.UseSqlite("Data Source=Product.db");

        public DbSet<Company> Companies { get; set; }


        public DbSet<Category> Categories { get; set; }


        public DbSet<Invoice> Invoices { get; set; }

        public DbSet<InvoiceProduct> invoiceProducts { get; set; }

        protected override void
         OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<InvoiceProduct>()
                .HasKey(x => new { x.ProductId, x.InvoiceId });
        }


    }
}
```

Spójrzmy na wygląd bazy poniżej. Typ Supplier zmapował się dobrze i jest przechowywany, lecz typ Customer nie jest przechowywany, jego property (Discount) jest ignorowane prze insercie. Sam typ jest przechowywany jako Company (Na co wskazuje 6-ta kolumna).

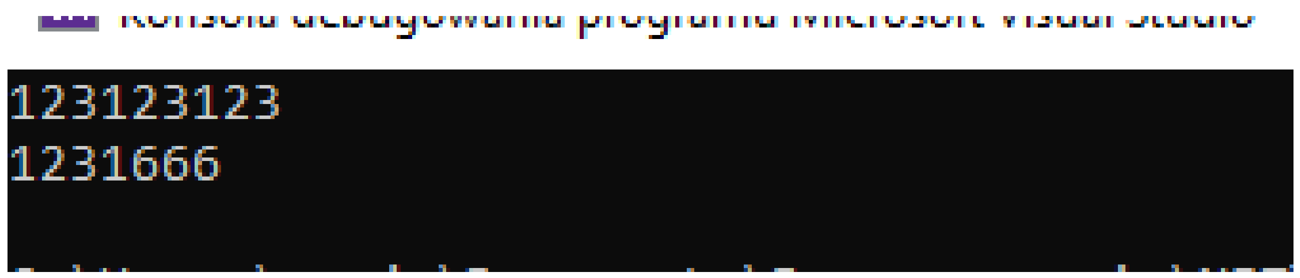


```
SQLite version 3.28.0 2019-04-16 19:49:53
Enter ".help" for usage hints.
sqlite> select * from Companies;
sqlite> select * from Companies;
sqlite> select * from Companies;
1|PegaSystems|Puszkarska|Krakow|30-551|Company|
2|Qualticks|Main street|Warsaw|20-551|Company|
3|Sabre|Aleja pokoju|Manchester|30-333|Supplier|123123123
4|SM|LondonStreet|London|30-311|Supplier|1231666
sqlite>
```

11.4 Wypisanie danych

Wypisanie danych dla Suppliera jest dobre, lecz dla Customera dostajemy błędy.

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.ObjectModel;
using System.Linq;
namespace ISurjakProductEF
{
    class Program
    {
        static void Main(string[] args)
        {

            ProdContext prodContext = new ProdContext();
            Customer customer = new Customer
            {
                City = "Krakow",
                CompanyName = "PegaSystems",
                Street = "Puszkarska",
                ZipCode = "30-551",
                Discount = 12

            };
            Supplier supplier = new Supplier
            {
                City = "Manchester",
                CompanyName = "Sabre",
                Street = "Aleja pokoju",
                ZipCode = "30-333",
                BankAccountNumber = 123123123

            };
            Customer customer2 = new Customer
            {
                City = "Warsaw",
                CompanyName = "Qualticks",
                Street = "Main street",
                ZipCode = "20-551",
                Discount = 55

            };
            Supplier supplier2 = new Supplier
            {
                City = "London",
                CompanyName = "SM",
                Street = "LondonStreet",
                ZipCode = "30-311",
                BankAccountNumber = 1231666

            };
            prodContext.Companies.Add(customer);
            prodContext.Companies.Add(customer2);
            prodContext.Companies.Add(supplier);
            prodContext.Companies.Add(supplier2);
            var data = prodContext.Companies.OfType<Supplier>().ToList();
            foreach (var d in data)
            {
                Console.WriteLine(d.BankAccountNumber);
            }
            prodContext.SaveChanges();

        }
    }
}
```

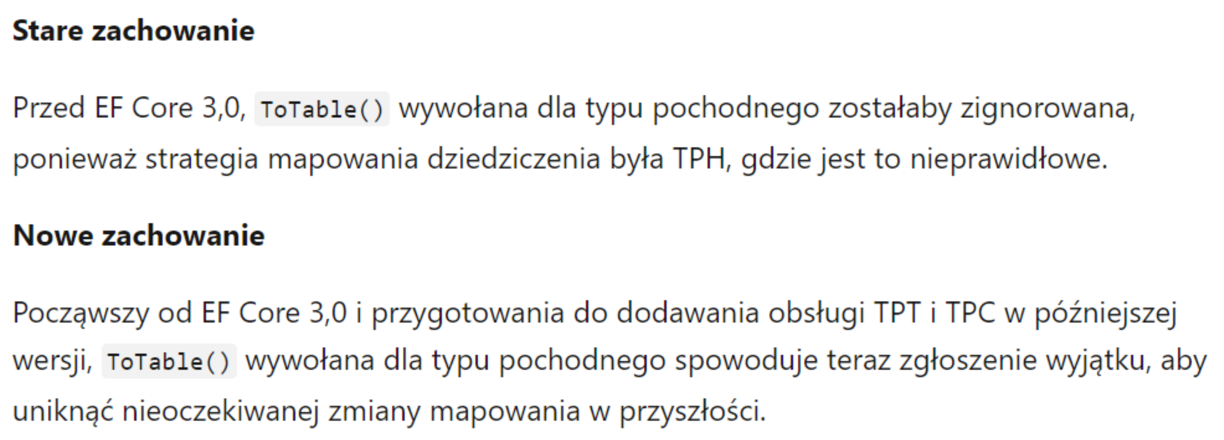


Konsola debugowania programu Microsoft Visual Studio

```
123123123
1231666
```

11.5 TablePerClass

Niestety nie byłem w stanie wykonać tej części zadania gdyż EntityFramework od wersji 3.0 podczas używania metody ToTable() która jest niezbędna do wykonania tego podpunktu , rzuca błąd zgodnie z dokumentacją.
Poniżej prezentuję screena z dokumentacji który potwierdza takie działanie aplikacji.



Stare zachowanie

Przed EF Core 3,0, `ToTable()` wywołana dla typu pochodnego zostałaby zignorowana, ponieważ strategia mapowania dziedziczenia była TPH, gdzie jest to nieprawidłowe.

Nowe zachowanie

Począwszy od EF Core 3,0 i przygotowania do dodawania obsługi TPT i TPC w późniejszej wersji, `ToTable()` wywołana dla typu pochodnego spowoduje teraz zgłoszenie wyjątku, aby uniknąć nieoczekiwanej zmiany mapowania w przyszłości.