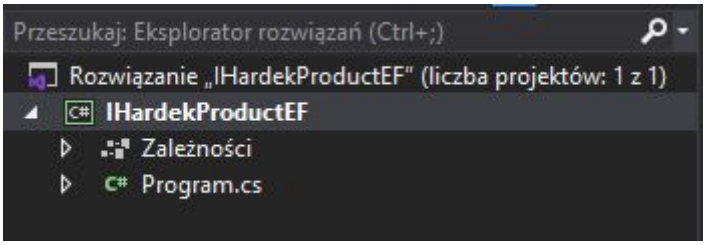
Lab 3 Entity Framework

I. Code First

a. Stwórz projekt typu ConsoleApplication .Net Core. Nazwij go INazwiskoProdutcEF



b. Dodaj klasę Product z polami int ProductID, string Name, int UnitsInStock. (Dodając do klasy property napisz prop I naciśnij dwa razy tabulator)

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

namespace IHardekProductEF
{
    Odwołania: 3
    class Product
    {
        Odwołania: 0
        public int ProductID { get; set; }
        Odwołania: 0
        public string Name { get; set; }
        Odwołania: 0
        public int UnitsInStock { get; set; }
    }
}
```

c. Stwórz klasę ProdContext dziedziczącą po DbContext.

```
namespace IHardekProductEF
{
    Odwołania: 2
    class ProdContext : DbContext
    {

    }
}
```

d. Dodaj do klasy kontekstowej zbiór (DbSet) produktów i nazwij go Products

```
public DbSet<Product> Products { get; set; }
```

e. W Mainie

i. poproś użytkownika o podanie nazwy produktu i zczytaj podana przez użytkownika nazwę

```
Console.WriteLine("Podaj nazwe produktu");
string prodName = Console.ReadLine();
Product product = new Product { Name = prodName };
```

ii. zainstancjonuj obiekt produktu ustawiając mu nazwę na tą zczytaną od użytkownika

```
Product product = new Product { Name = prodName };
```

iii. Stwórz instancje ProdContext'u

```
ProdContext prodContext = new ProdContext();
```

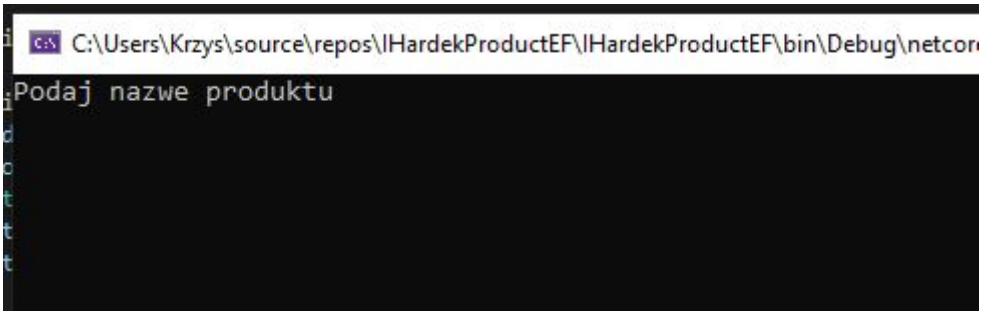
iv. dodaj zainstancjonowany obiekt do kontekstowej kolekcji Produktów

```
prodContext.Products.Add(product);
```

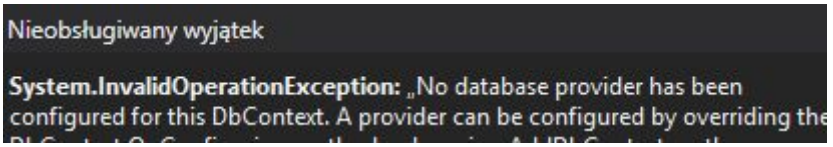
v. zapisz zmiany na kontekście

```
prodContext.SaveChanges();
```

vi. Zbuduj i uruchom aplikacje



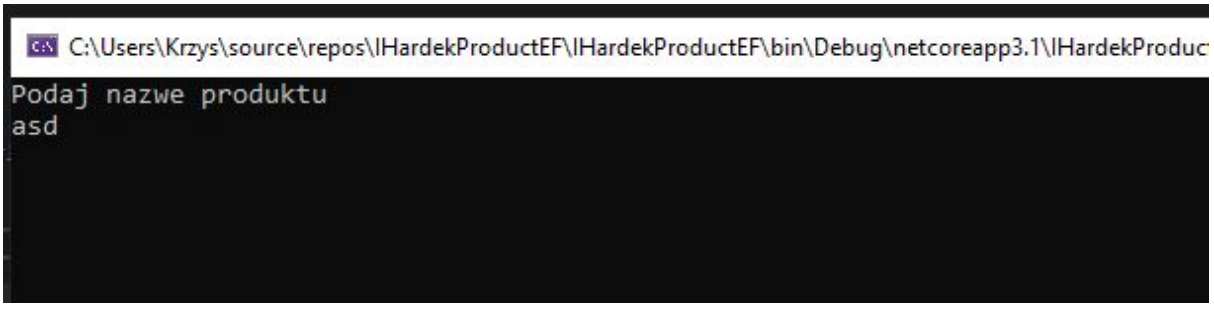
vii. Wyjątek: No database provider has been configured



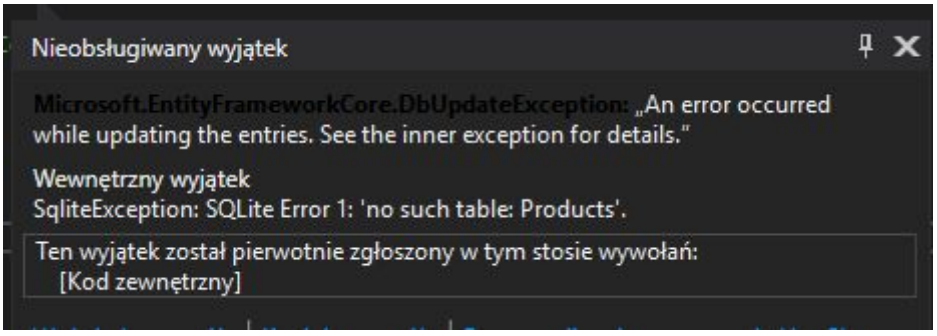
viii. Konfiguracja kontekstu

```
protected override void OnConfiguring(DbContextOptionsBuilder options) => options.UseSqlite("DataSource=Product.db");
```

ix. Zbuduj i uruchom aplikacje



x. Wyjątek mówiący o tym, że nie istnieje tabela produktów

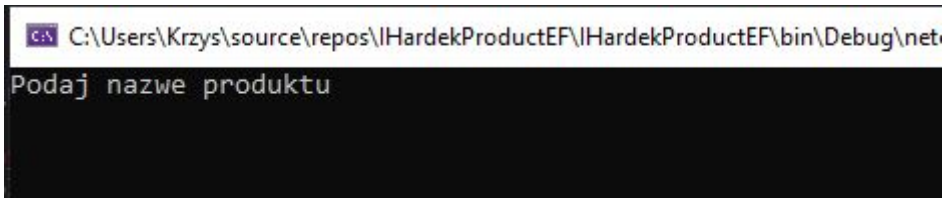



```
<DIR>          bin
           639 IHardekProductEF.
<DIR>          obj
           379 ProdContext.cs
           275 Product.cs
        12 288 Product.db
           614 Program.cs
(s)         14 195 bytes
s)  16 721 477 632 bytes free
```

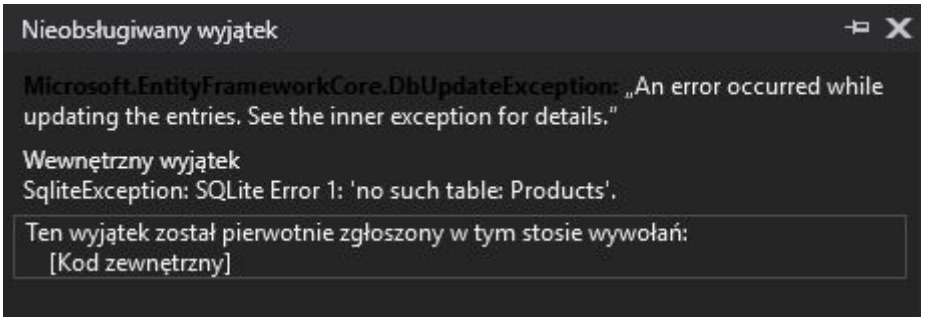
f. Dopisz w mainie fragment kodu pobierający oraz wyświetlający dostępne Produkty.

```
var query = from p in prodContext.Products
            select p.Name;
```

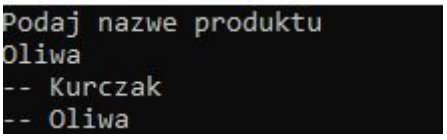
g. Zbuduj, uruchom i przetestuj aplikacje



h. Wyjątek o braku tabeli Products.

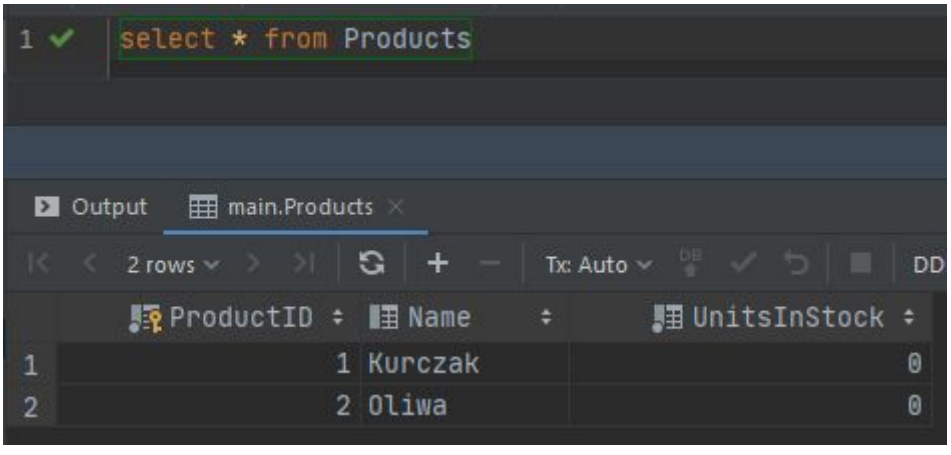


i. Po tej zmianie aplikacja powinna już zacząć działać zgodnie z oczekiwaniami

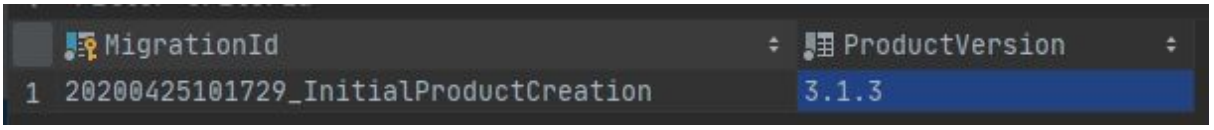


j. Widok bazy z datagrip

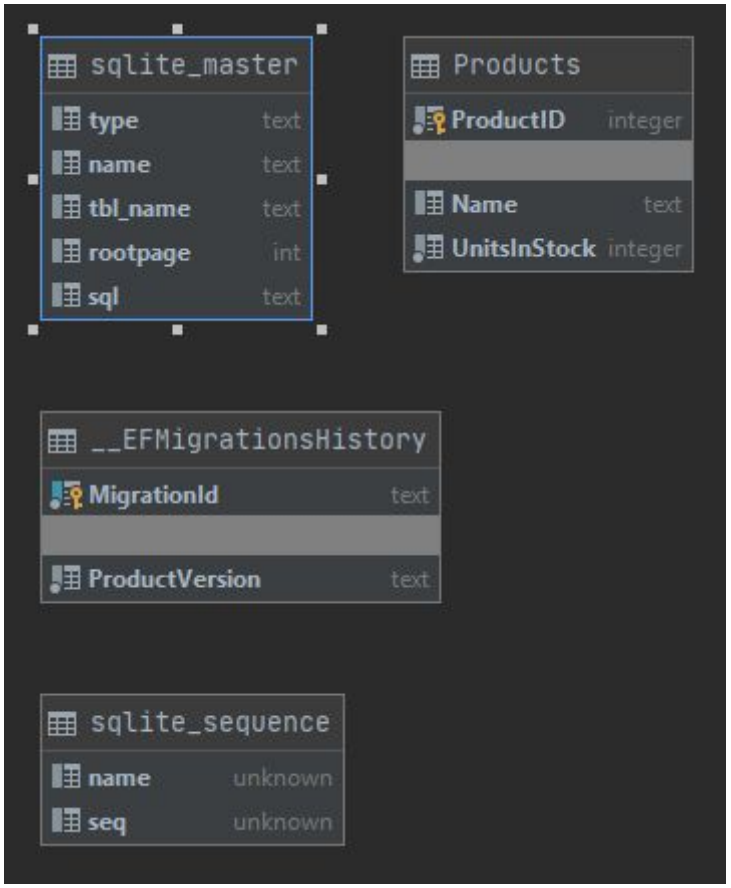
Wszystkie produkty



Migracje



Diagram



II. Modyfikacja modelu przez wprowadzenie dostawcy

a. Dodanie Klasy supplier

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

namespace IHardekProductEF
{
    Odwołania: 4
    class Supplier
    {
        1 odwołanie
        public int SupplierID { get; set; }
        1 odwołanie
        public string CompanyName { get; set; }
        1 odwołanie
        public string Street { get; set; }
        1 odwołanie
        public string City { get; set; }
    }
}
```

b. Dodanie pola klucza obcego w klasie Product

```
[ForeignKey("SupplierID")]
Odwołania: 3
public Supplier Supplier { get; set; }
```

c. Dodanie DbSet w ProdContext, aby stworzyć tabele

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

d. Migracje

```
C:\Users\Krzys\source\repos\IHardekProductEF\IHardekProductEF>dotnet ef migrations add SupplierCreation
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'

C:\Users\Krzys\source\repos\IHardekProductEF\IHardekProductEF>dotnet ef database update
Build started...
Build succeeded.
Applying migration '20200425111030_SupplierCreation'.
Done.
```

e. Modyfikacja maina. Wstawienie dostawcy oraz wypisanie wszystkich produktów wraz z dostawcami

```
Podaj nazwe produktu
kurczak
-- kurczak, Google
```

f. Widok z perspektywy datagrip

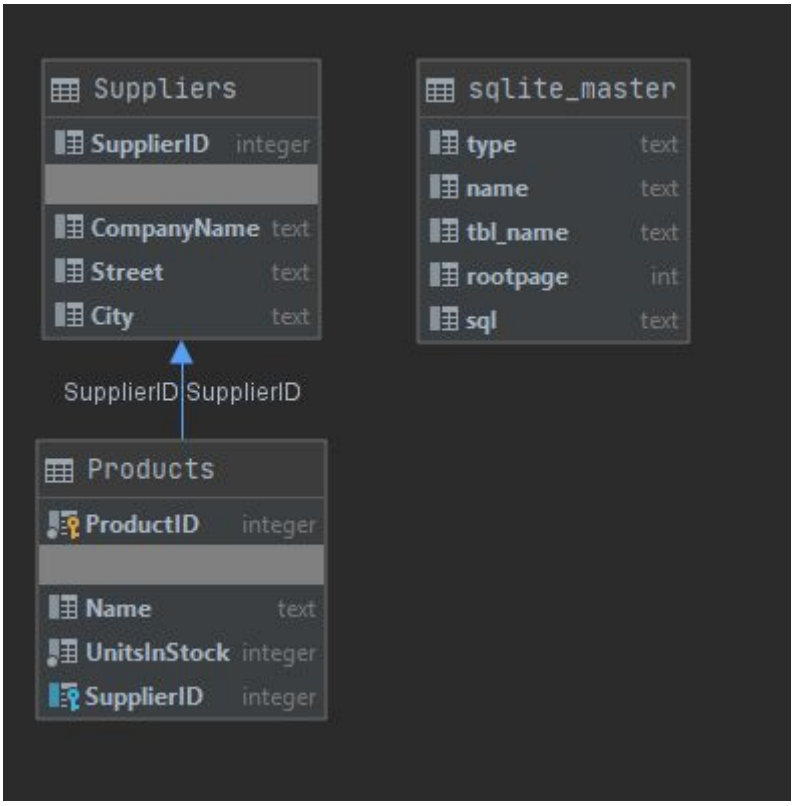
Tabela products

	ProductID	Name	UnitsInStock	SupplierID
1	1	kurczak	5	1

Tabela Suppliers

	SupplierID	CompanyName	Street	City
1	1	Google	Bema	Krakow

Diagram



III. Odwrócenie relacji

a. Dodaje pole typy ICollection<Product> w klasie Supplier oraz tworzę dla niego konstruktor

b.

```
1 odwołanie
public Supplier() { Products = new List<Product>(); }

Odwołania: 3
public ICollection<Product> Products { get; set; }
```

c. Usuwam pola z adnotacją klucza obcego w klasie Product

```
//[ForeignKey("SupplierID")]
//public Supplier Supplier { get; set; }
```

d. Odpowiednio zmieniam maina

```
static void Main(string[] args)
{
    ProdContext prodContext = new ProdContext();
    string prodName;

    Console.WriteLine("Podaj nazwe produktu");
    prodName = Console.ReadLine();
    Product product1 = new Product { Name = prodName, UnitsInStock = 5 };

    Console.WriteLine("Podaj nazwe produktu");
    prodName = Console.ReadLine();
    Product product2 = new Product { Name = prodName, UnitsInStock = 5 };


    Supplier supplier = new Supplier
    {
        City = "Oswiecim",
        CompanyName = "Skala",
        Street = "Bema",
    };
    supplier.Products.Add(product1);
    supplier.Products.Add(product2);

    prodContext.Suppliers.Add(supplier);
    prodContext.Products.Add(product1);
    prodContext.Products.Add(product2);
    prodContext.SaveChanges();
}
```

e. Dokonuje migracji, aby zapisać dane w modelu

```
C:\Users\Krzys\source\repos\IHardekProductEF\IHardekProductEF>dotnet ef migrations add RevertRelationProdSupp
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'

C:\Users\Krzys\source\repos\IHardekProductEF\IHardekProductEF>dotnet ef database update
Build started...
Build succeeded.
Applying migration '20200425120154_RevertRelationProdSupp'.
Done.
```

f. Uruchamiam program, aby dodać produkty

```
Podaj nazwe produktu
kotek
Podaj nazwe produktu
piesek
```

g. Widok z perspektywy datagrip

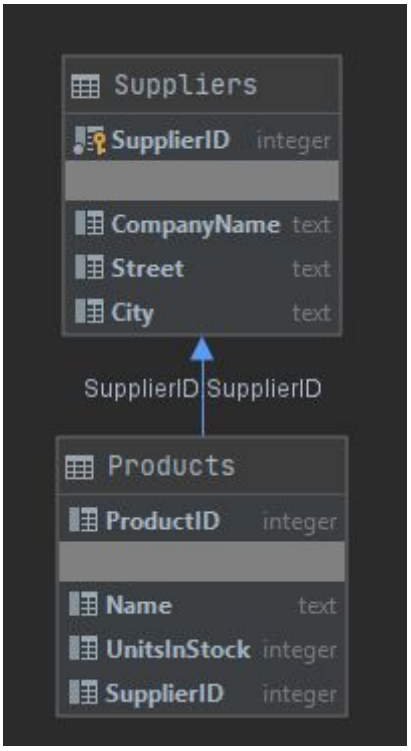
Tabela Products

	ProductID	Name	UnitsInStock	SupplierID
1	1	kotek	5	1
2	2	piesek	5	1

Tabela Suppliers

	SupplierID	CompanyName	Street	City
1	1	Skala	Bema	Oswiecim

Diagram



Jak widać rezultat jest taki sam jak w poprzednim podpunkcie.

IV. Relacja obustronna

a. Usuwam zakomentowane linie w Klasie produkt, w mainie dodaje suppliera do konstruktora produktów oraz dokonuje migracji.
b. Uruchamiam program oraz dodaje produkty

```
Podaj nazwe produktu
czipsy
Podaj nazwe produktu
piwo
```

c. Widok z datagrip

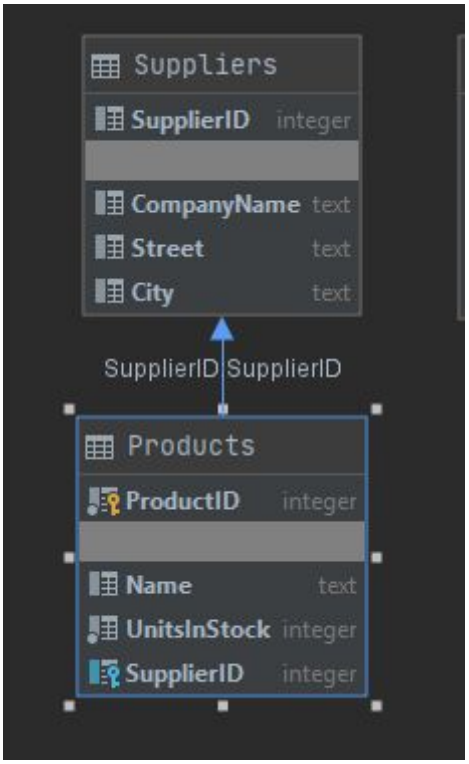
Klasa Product

	ProductID	Name	UnitsInStock	SupplierID
1	1	czipsy	5	1
2	2	piwo	5	1

Klasa Suppliers

	SupplierID	CompanyName	Street	City
1	1	Facebook	Sienkiewicza	Katowice

Diagram



Jak widać schemat nie zmienił się. W takim razie poznaliśmy trzy sposoby na stworzenie klucza obcego po stronie aplikacji.

VI. Relacja wiele do wielu

a. Tworzę klasę, która reprezentuje zjoinowaną tabele

```
namespace IHardekProductEF
{
    Odwołania: 15
    class ProductInvoice
    {

        1 odwołanie
        public int ProductID { get; set; }

        Odwołania: 3
        public int InvoiceID { get; set; }

        Odwołania: 4
        public Product Product { get; set; }
        Odwołania: 4
        public Invoice Invoice { get; set; }
    }
}
```

b. Tworzę referencje do tej klasy w klasach Product oraz Invoice

```
public ICollection<ProductInvoice> ProductInvoices{ get; set; }
```

c. Zawartość klasy Invoice

```
Odwołania: 8
class Invoice
{
    Odwołania: 0
    public int InvoiceID { get; set; }
    Odwołania: 2
    public int Quantity { get; set; }

    Odwołania: 2
    public Invoice()
    {
        ProductInvoices = new List<ProductInvoice>();
    }

    Odwołania: 3
    public ICollection<ProductInvoice> ProductInvoices { get; set; }

}
```

d. Stworzenie klucza kompozytowego w ProdContext oraz dodanie tam tabeli Invoices i ProductInvoices

```
Odwołania: 0
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<ProductInvoice>().HasKey(pi => new { pi.ProductID, pi.InvoiceID});
}

Odwołania: 2
public DbSet<Product> Products { get; set; }

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

Odwołania: 0
public DbSet<Invoice> Invoices { get; set; }

Odwołania: 0
public DbSet<Invoice> ProductInvoice { get; set; }
```

e. Dodanie produktów w aplikacji oraz ich wyświetlanie

```
Podaj nazwe produktu
skarpetki
Podaj nazwe produktu
kurtka
Product:skarpetki
1
2
Product:kurtka
1
2
Invoice:1
skarpetki
kurtka
Invoice:2
skarpetki
kurtka
```

f. Widok z perspektywy datagrip

Tabela Invoice

	InvoiceID	Quantity
1	1	2
2	2	2

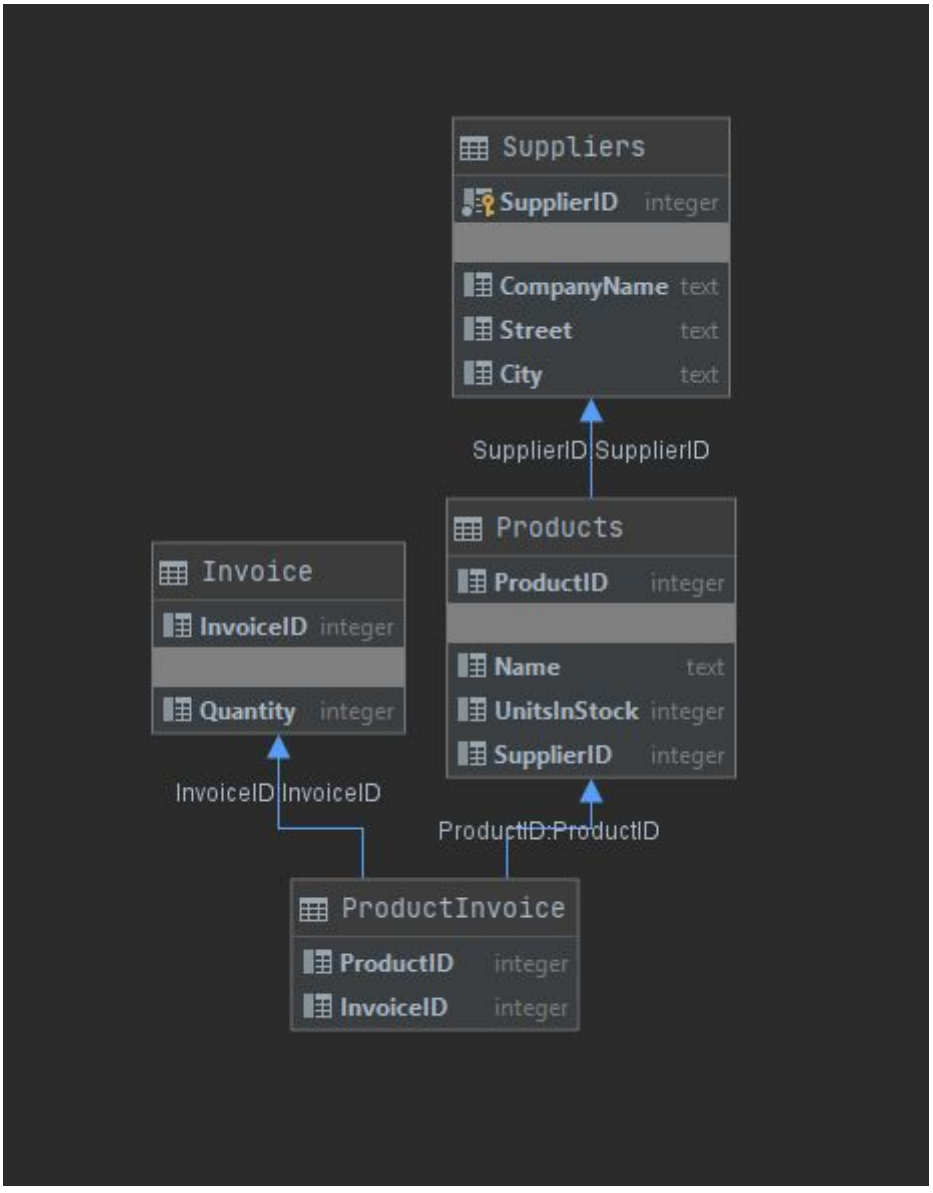
Zjoinowana Tabela ProductInvoice

	ProductID	InvoiceID
1	1	1
2	1	2
3	2	1
4	2	2

Tabela Product

	ProductID	Name	UnitsInStock	SupplierID
1	1	skarpetki	5	1
2	2	kurtka	5	1

Diagram



V. Klasa Category (Przypadkiem zrobiłem podpunkt po punkcie VI)

a. Stworzenie klasy Category

```
Odwołania: 6
class Category
{
    Odwołania: 0
    public int CategoryID { get; set; }
    Odwołania: 8
    public string Name { get; set; }

    Odwołania: 2
    public Category()
    {
        Products = new List<Product>();
    }
    Odwołania: 11
    public ICollection<Product> Products { get; set; }

}
```

b. Dodanie kilku produktów oraz wyświetlenie ich wraz z kategoriami

```
Podaj nazwe produktu
parowki
Podaj nazwe produktu
kotlet
Podaj nazwe produktu
cola
Podaj nazwe produktu
harnas
Product:parowki
Jedzenie
Product:kotlet
Jedzenie
Category:Jedzenie
parowki
kotlet
Category:Picie
cola
harnas
```

c. Widok z perspektywy datagrip

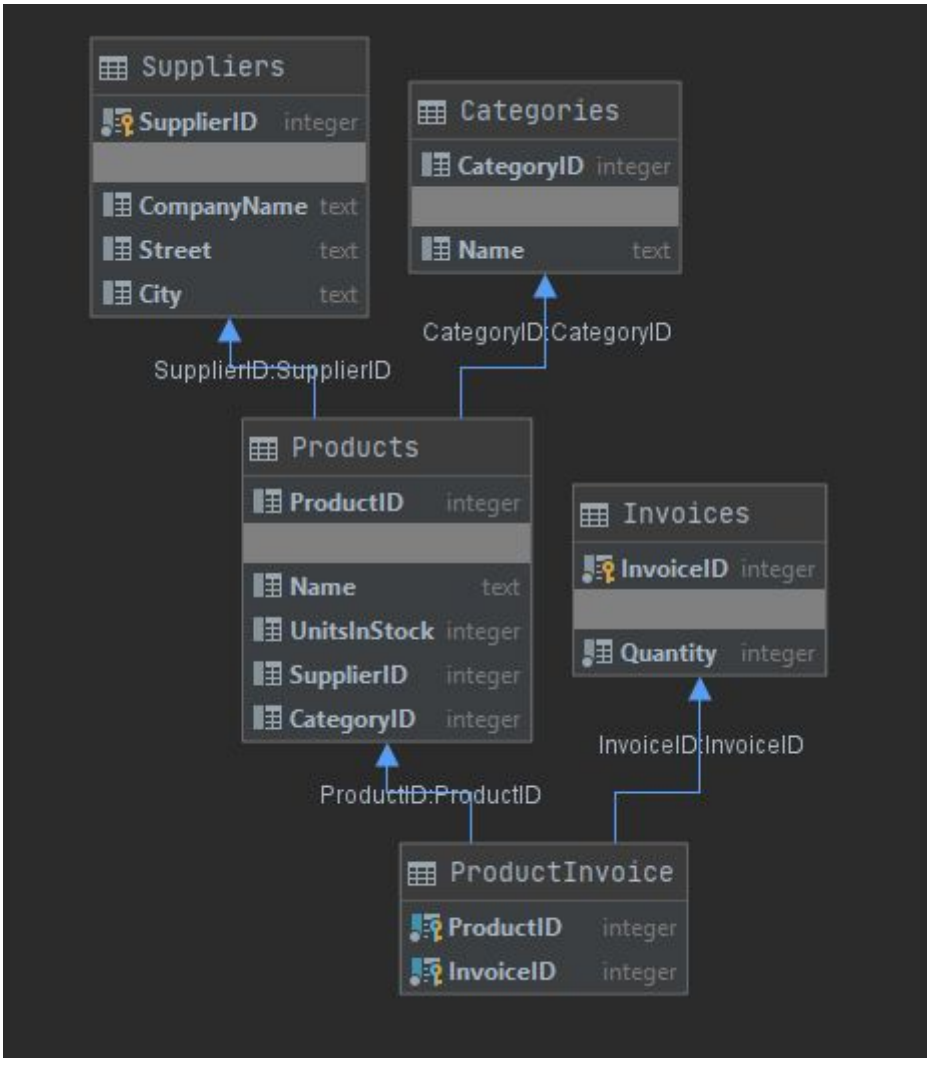
Tabela Products

	ProductID	Name	UnitsInStock	SupplierID	CategoryID
1	1	parowki	5	1	1
2	2	kotlet	5	1	1
3	3	cola	5	1	2
4	4	harnas	5	1	2

Tabela Categories

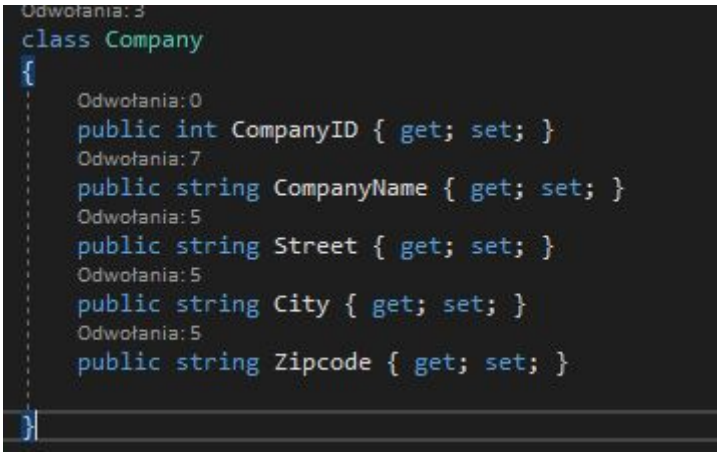
	CategoryID	Name
1	1	Jedzenie
2	2	Picie

Diagram

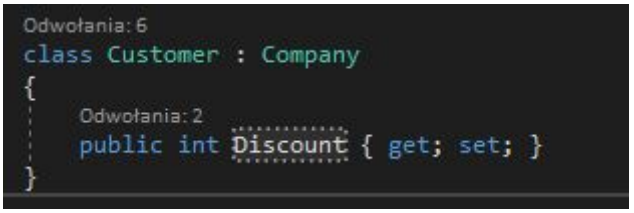


VII. Dziedziczenie - Podejście TablePerHierarchy

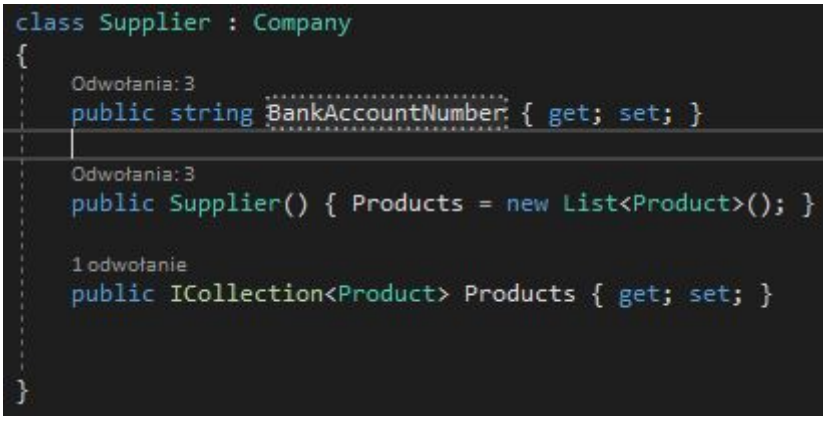
a. Klasa Company



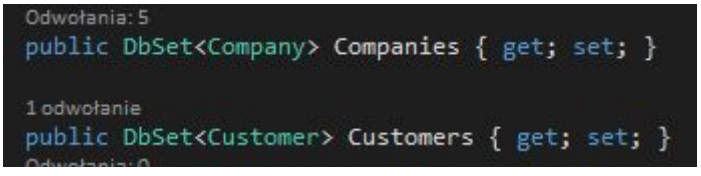
b. Klasa Customer



c. Klasa Supplier



d. Klasa ProdContext - dodanie tabeli Companies oraz Customers

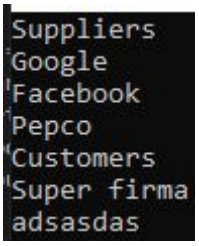


e. Wpisywanie oraz wyświetlanie firm

```
static void Main(string[] args)
{
    ProdContext prodContext = new ProdContext();

    Supplier supplier1 = new Supplier
    {
        City = "Katowice",
        CompanyName = "Google",
        Street = "Sienkiewicza",
        Zipcode = "32-611",
        BankAccountNumber = "1231231231232131"
    };

    Supplier supplier2 = new Supplier
    {
        City = "Krakow",
        CompanyName = "Facebook",
        Street = "Bracka",
        Zipcode = "32-600",
        BankAccountNumber = "1238912839123831921231"
    };

    Supplier supplier3 = new Supplier
    {
        City = "Chrzanow",
        CompanyName = "Pepco",
        Street = "Glowna",
        Zipcode = "32-123",
        BankAccountNumber = "123891283912381231239"
    };

    Customer customer1 = new Customer
    {
        City = "Lodz",
        CompanyName = "Super firma",
        Street = "asdasd",
        Zipcode = "23-022",
        Discount = 15
    };

    Customer customer2 = new Customer
    {
        City = "Gdansk",
        CompanyName = "adsasdas",
        Street = "Super ulica",
        Zipcode = "11-111",
        Discount = 1
    };
```

```
prodContext.Companies.Add(supplier1);
prodContext.Companies.Add(supplier2);
prodContext.Companies.Add(supplier3);
prodContext.Companies.Add(customer1);
prodContext.Companies.Add(customer2);
prodContext.SaveChanges();

Console.WriteLine("Suppliers");
foreach(Supplier c in prodContext.Suppliers)
{
    Console.WriteLine(c.CompanyName);
}
Console.WriteLine("Customers");
foreach (Customer c in prodContext.Customers)
{
    Console.WriteLine(c.CompanyName);
}
```

f. Wykonanie programu

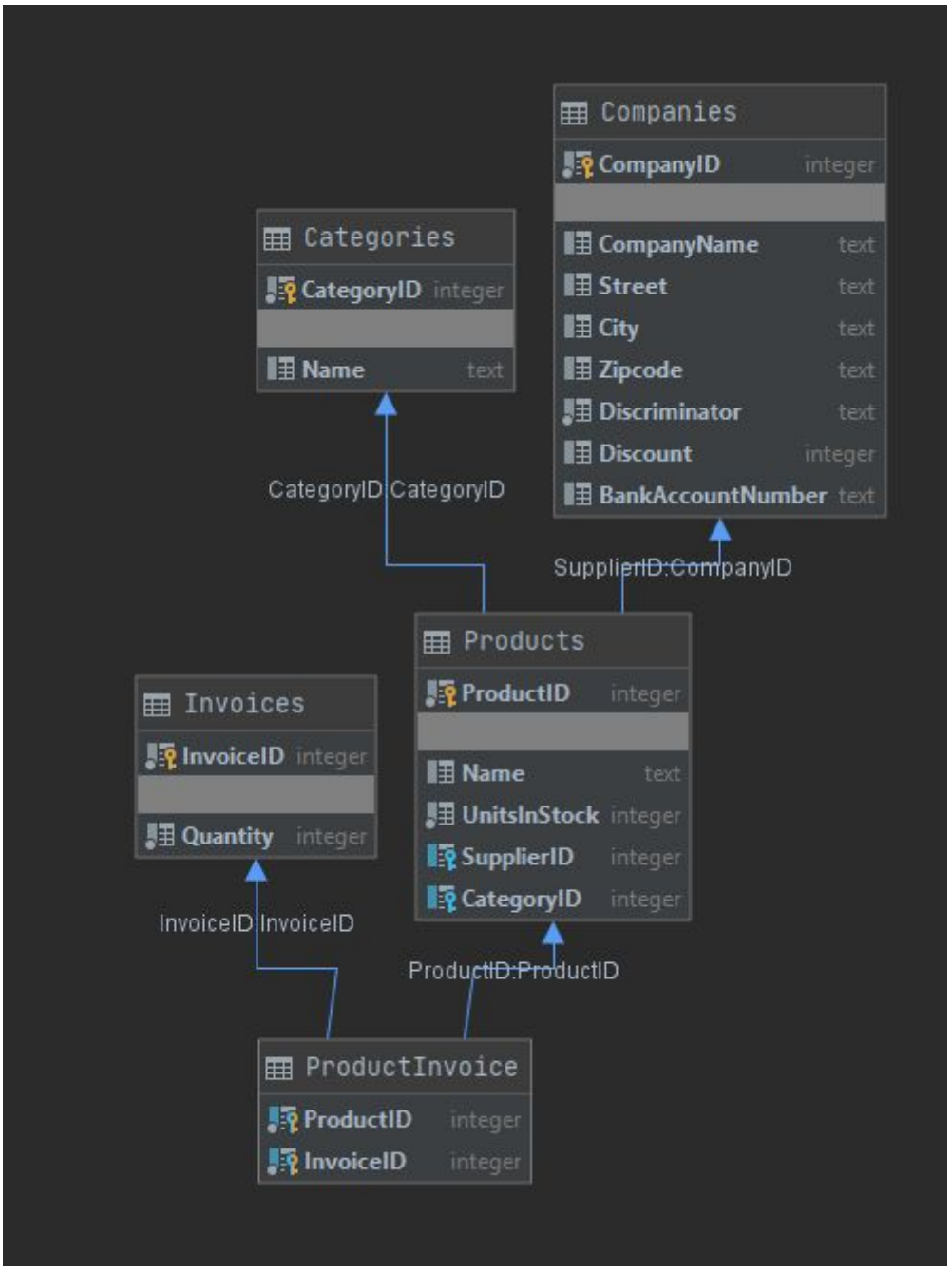


g. Widok z perspektywy datagrip

Tabela Companies

	CompanyID	CompanyName	Street	City	Zipcode	Discriminator	Discount	BankAccountNumber
1	1	Google	Sienkiewicza	Katowice	32-611	Supplier	<null>	1231231231232131
2	2	Facebook	Bracka	Krakow	32-600	Supplier	<null>	1238912839123831921231
3	3	Pepco	Glowna	Chrzanow	32-123	Supplier	<null>	123891283912381231239
4	4	Super firma	asdasd	Lodz	23-022	Customer	15	<null>
5	5	adsasdas	Super ulica	Gdansk	11-111	Customer	1	<null>

Diagram


Companies
CompanyID integer
CompanyName text
Street text
City text
Zipcode text
Discriminator text
Discount integer
BankAccountNumber text
Categories
CategoryID integer
Name text
CategoryID:CategoryID
SupplierID:CompanyID
Products
ProductID integer
Name text
UnitsInStock integer
SupplierID integer
CategoryID integer
Invoices
InvoiceID integer
Quantity integer
InvoiceID:InvoiceID
ProductID:ProductID
ProductInvoice
ProductID integer
InvoiceID integer