# Entity Framework

## Aleksandra Mazur

### Zadanie I

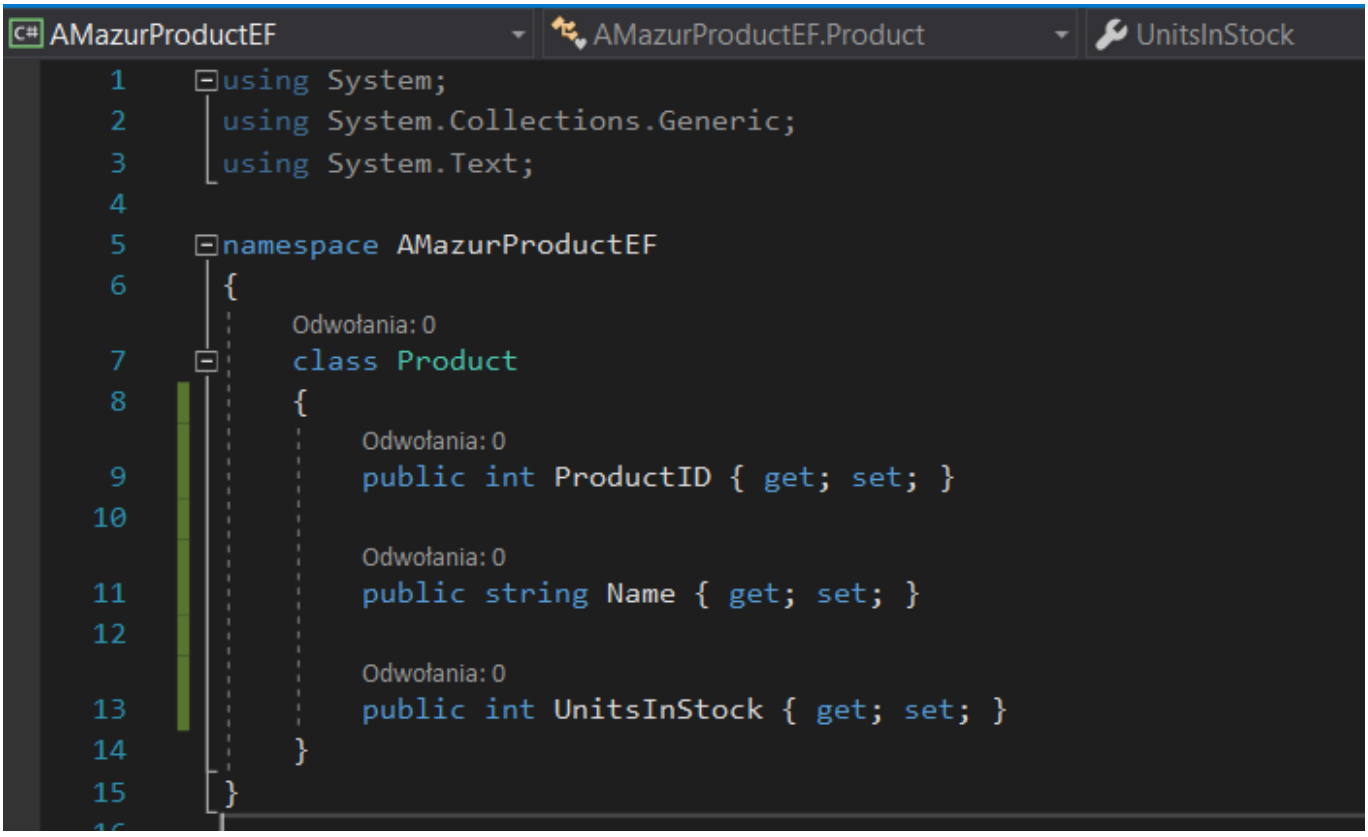
**a)**

Stworzono projekt typu Console Application .Net Core o nazwie **AMazurProductEF**.

**b)**

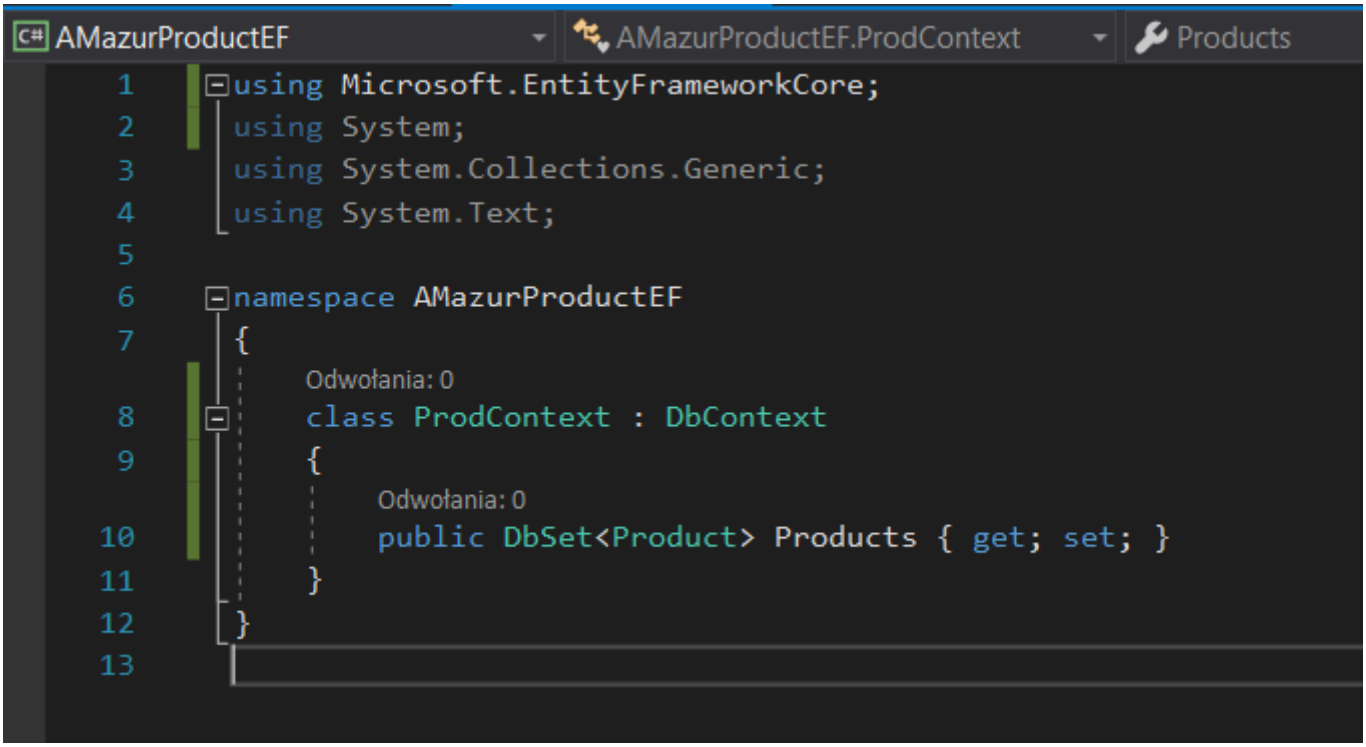
Dodano klasę *Product* z polami:

- int ProductID
- string Name
- int UnitsInStock



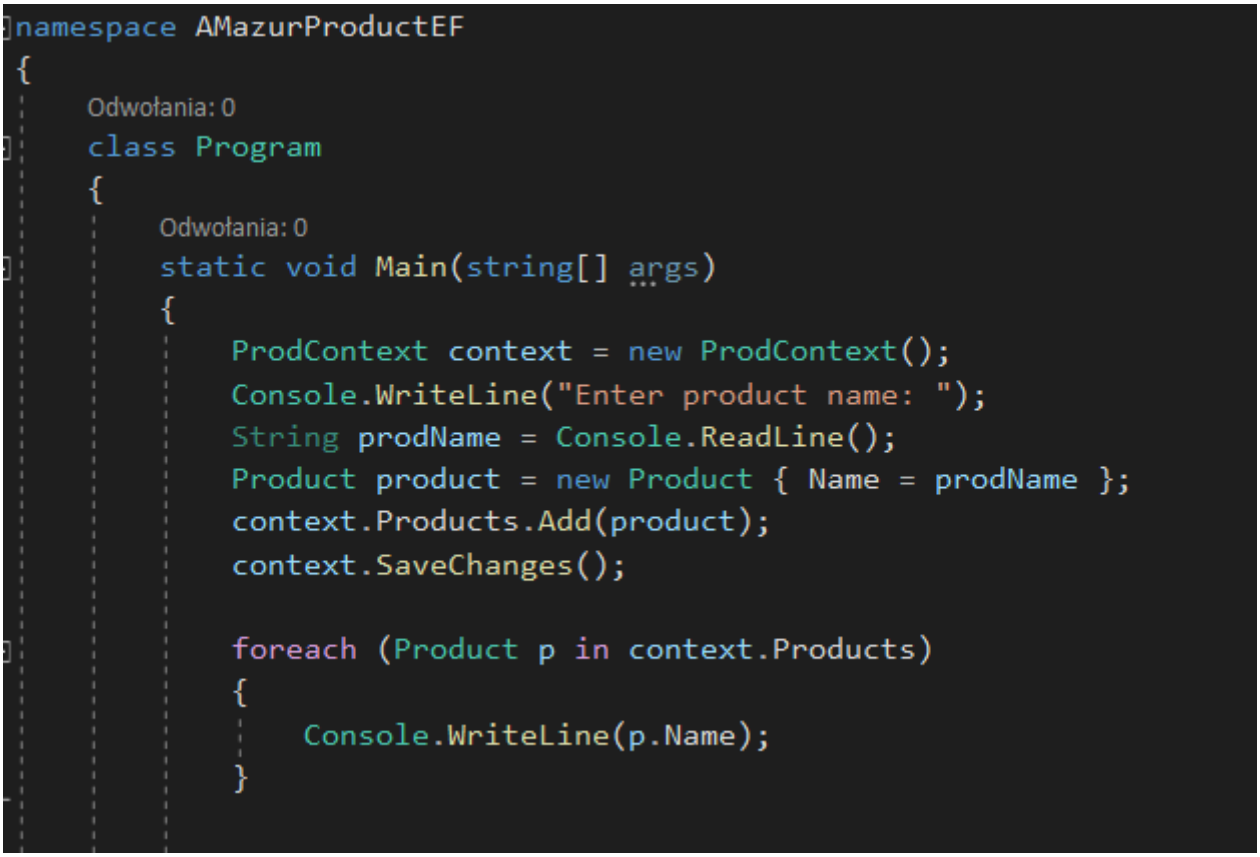
**c), d)**

Stworzono klasę *ProdContext* dziedziczącą po *DbContext* i dodano do niej zbiór (*DbSet*) produktów.



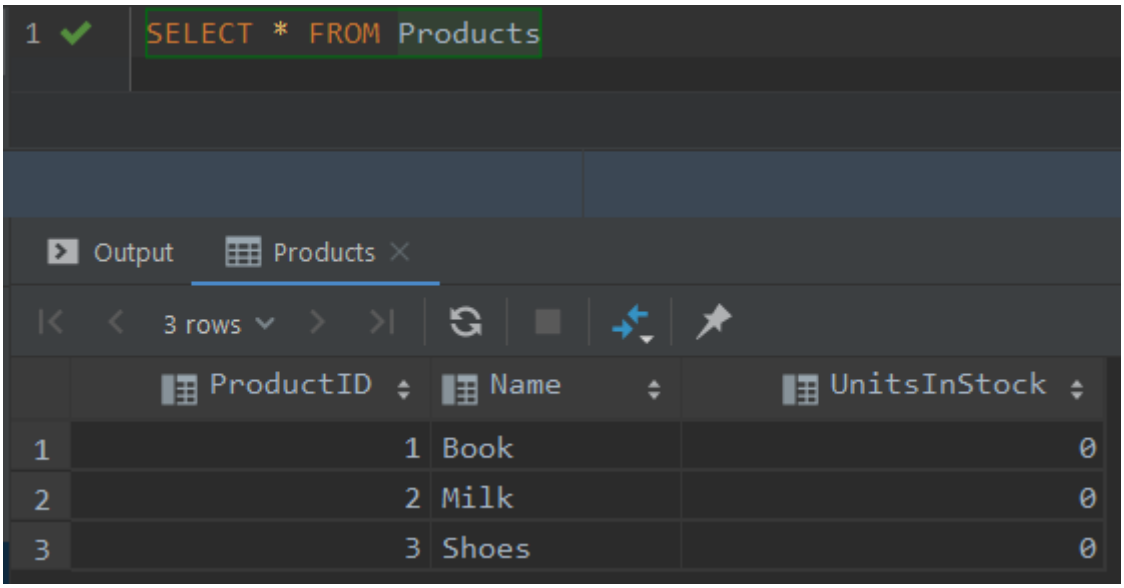

**e), f)**

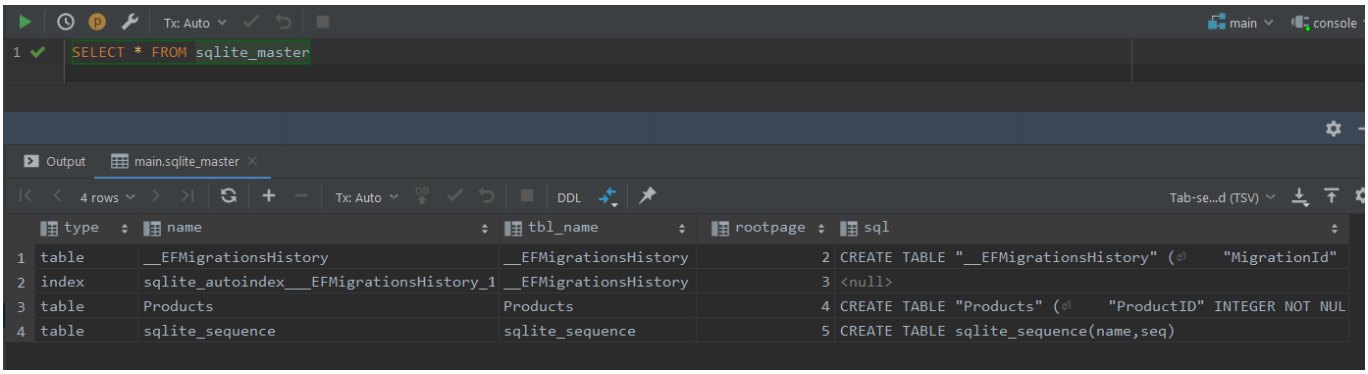
W Mainie (plik Program.cs) poproszono użytkownika o podanie nazwy produktu i utworzono obiekt produktu o wczytanej nazwie. Następnie dopisano kod pobierający oraz wyświetlający wszystkie produkty.

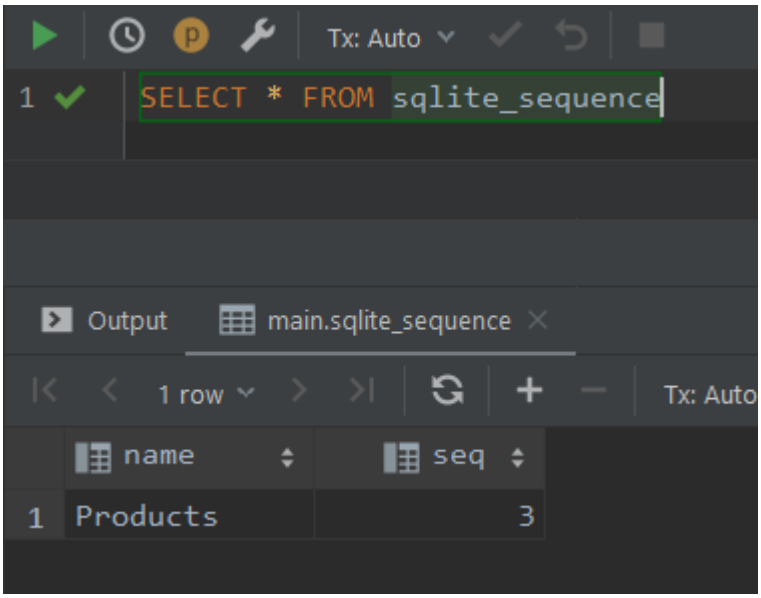


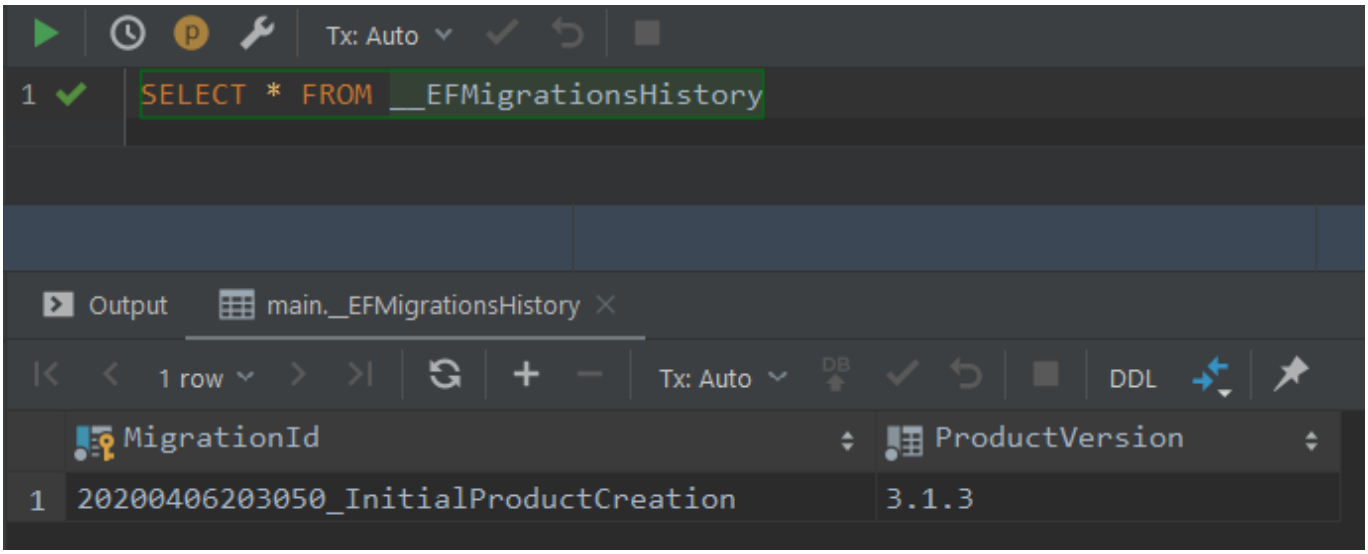
Konsola debugowania programu Microsoft Visual Studio

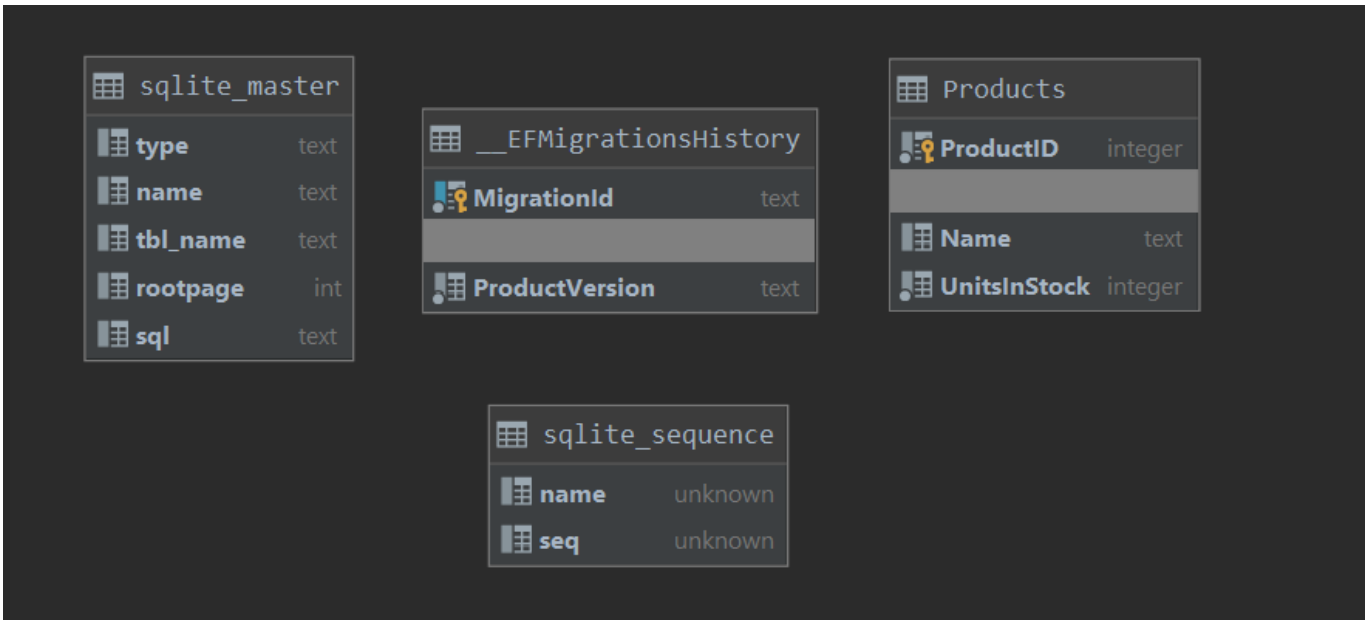
```
Enter product name:
Shoes
Book
Milk
Shoes
```



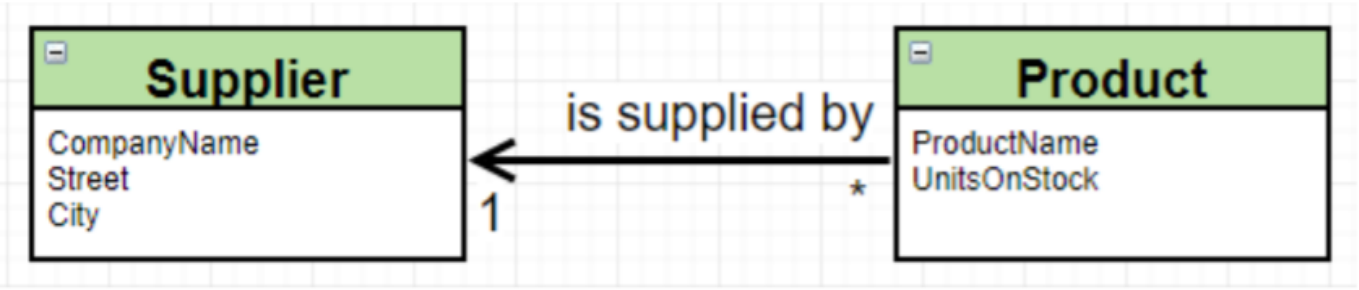




Tx: Auto
1
SELECT * FROM __EFMigrationsHistory
Output
main.__EFMigrationsHistory
1 row
Tx: Auto
DDL
MigrationId
ProductVersion
1
20200406203050_InitialProductCreation
3.1.3


sqlite_master
type text
name text
tbl_name text
rootpage int
sql text
__EFMigrationsHistory
MigrationId text
ProductVersion text
Products
ProductID integer
Name text
UnitsInStock integer
sqlite_sequence
name unknown
seq unknown

## Zadanie II

Zmodyfikowano model wprowadzając pojęcie Dostawcy - *Supplier* jak poniżej.

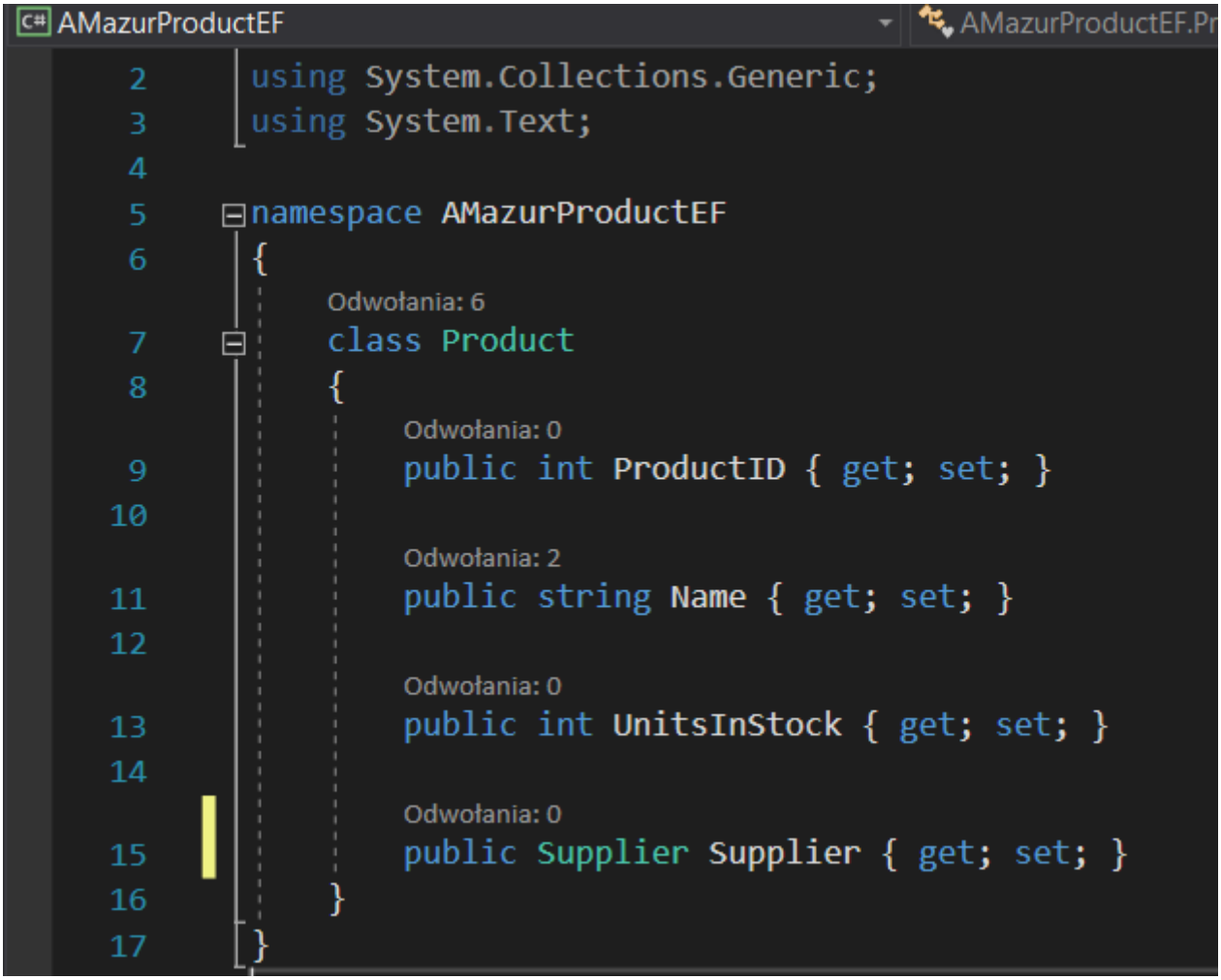



Klasa *Supplier* zawiera następujące pola:

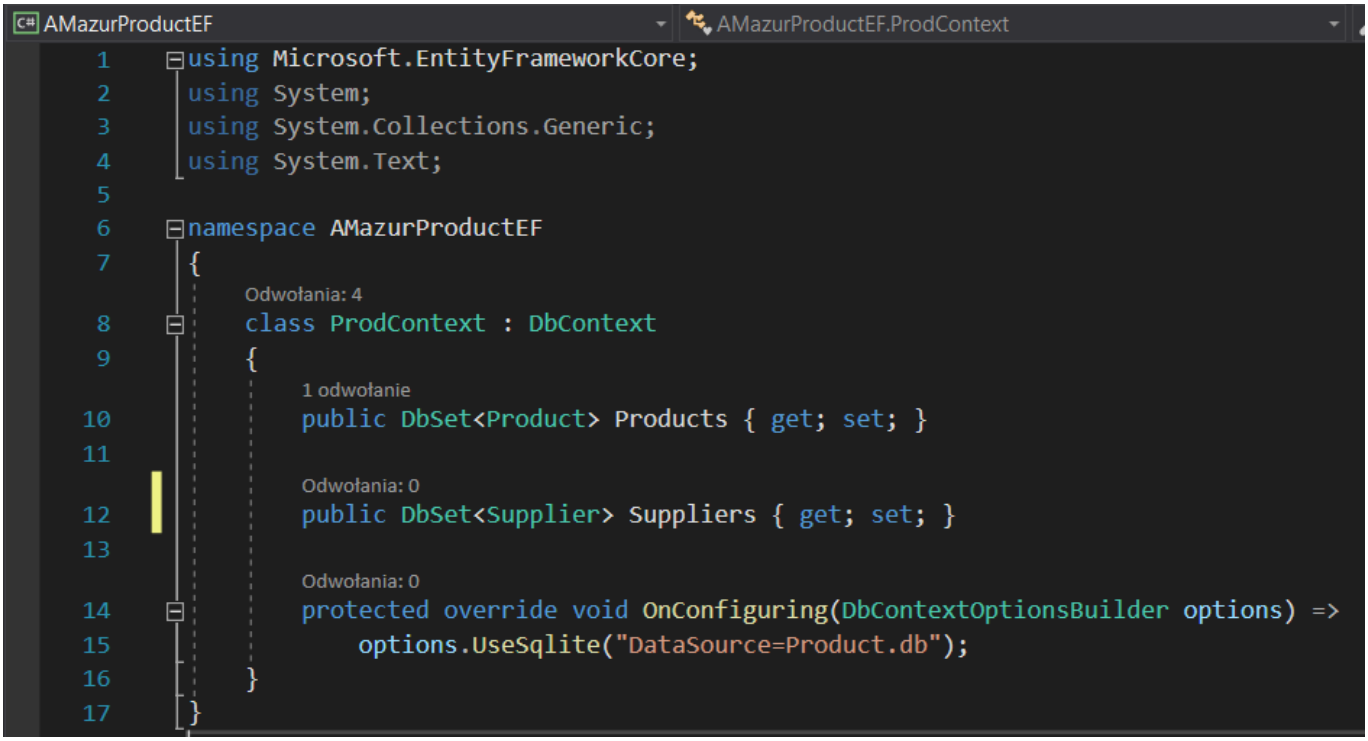
- int SupplierID
- string CompanyName
- string Street
- string City

```
Category.cs | Supplier.cs* | ProdContext.cs | Product.cs | Program.cs
AMazurProductEF | AMazurProductEF.Suppl

1   using System;
2   using System.Collections.Generic;
3   using System.Text;
4
5   namespace AMazurProductEF
6   {
        Odwołania: 2
7       class Supplier
8       {
9
            Odwołania: 0
10          public int SupplierID { get; set; }
            Odwołania: 0
11          public string CompanyName { get; set; }
12
            Odwołania: 0
13          public string Street { get; set; }
14
            Odwołania: 0
15          public string City { get; set; }
16
17      }
18  }
```

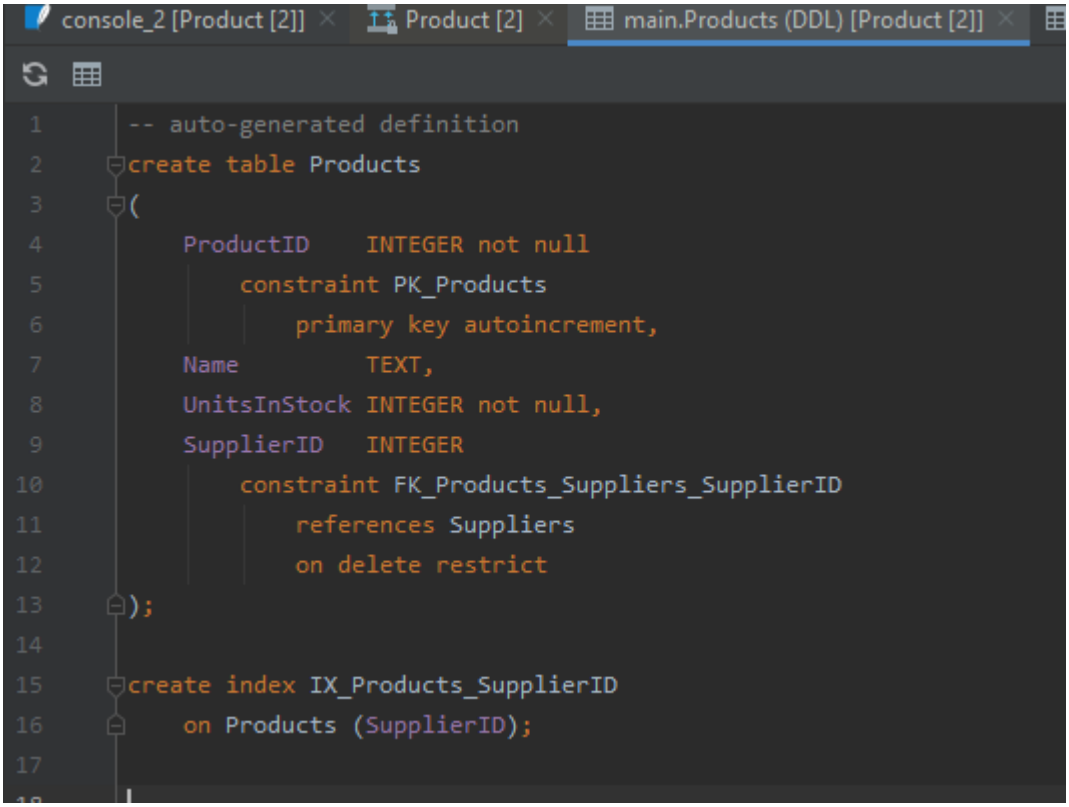
Do klasy Product dodano pole Supplier.

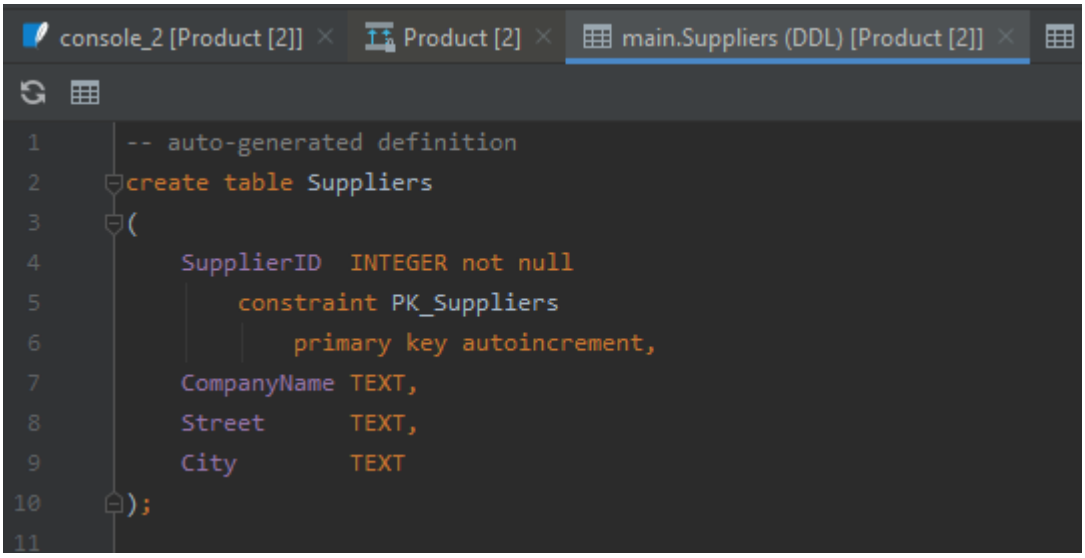


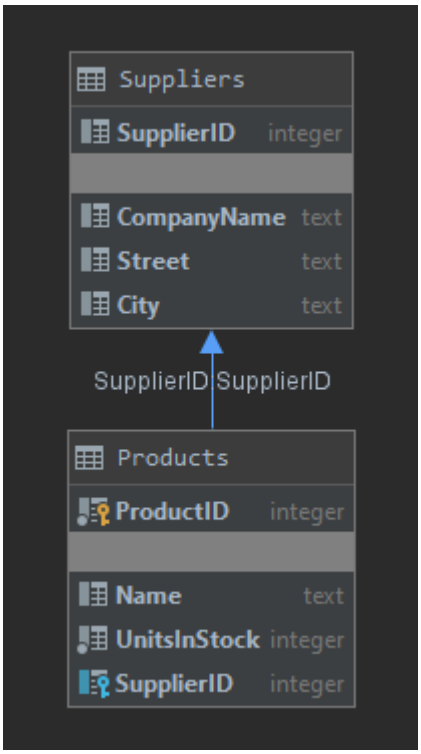

Do klasy *ProdContext* dodano kolejny *DbSet*, reprezentujący wszystkich dostawców.



Schemat w bazie danych wygląda następująco:



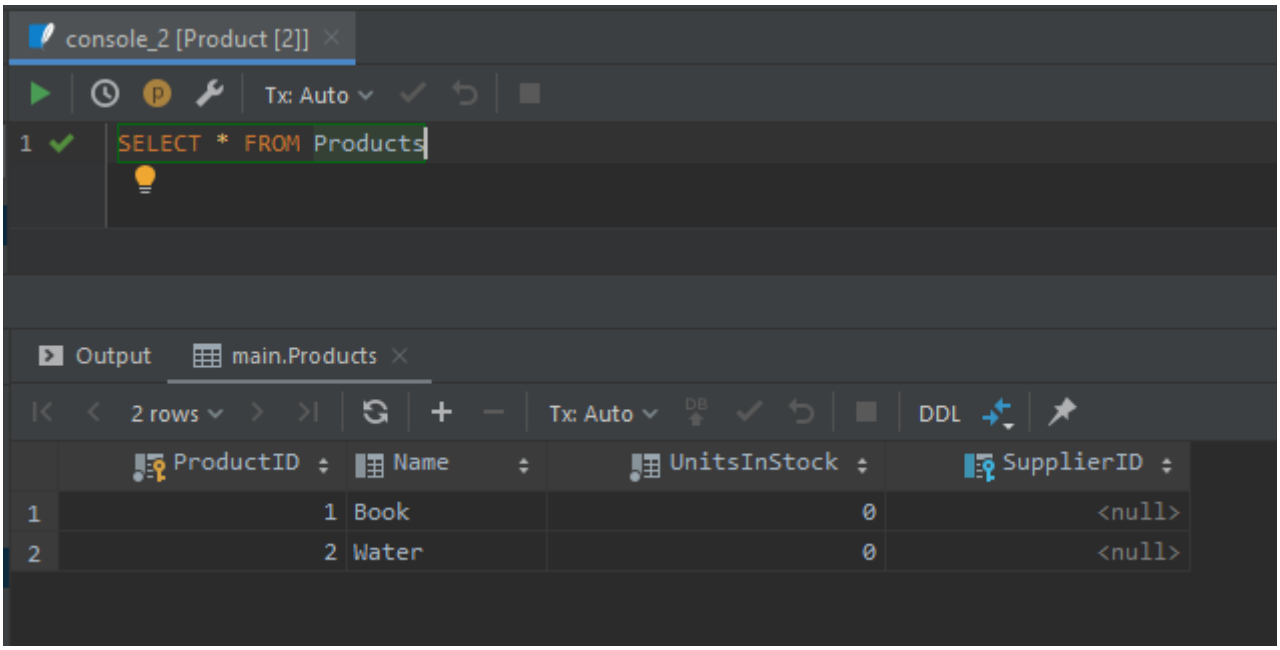


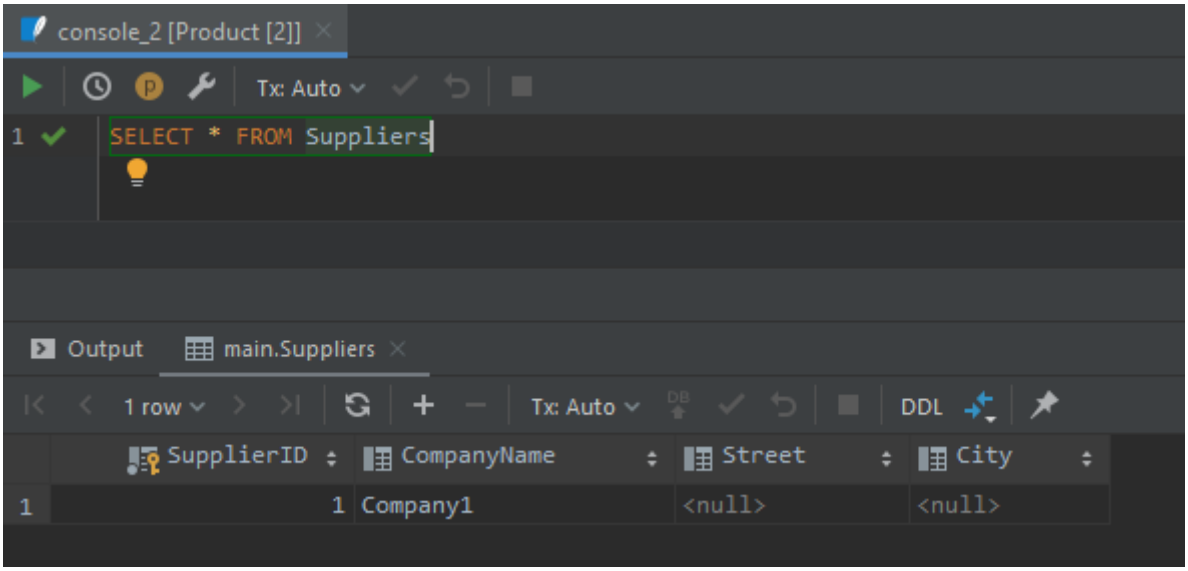

**d), e), f), g)**

Dodano do bazy dwa produkty oraz dostawcę. Następnie znaleziono poprzednio wprowadzone produkty i ustawiono dostawcę każdego z nich na wcześniej stworzonego.

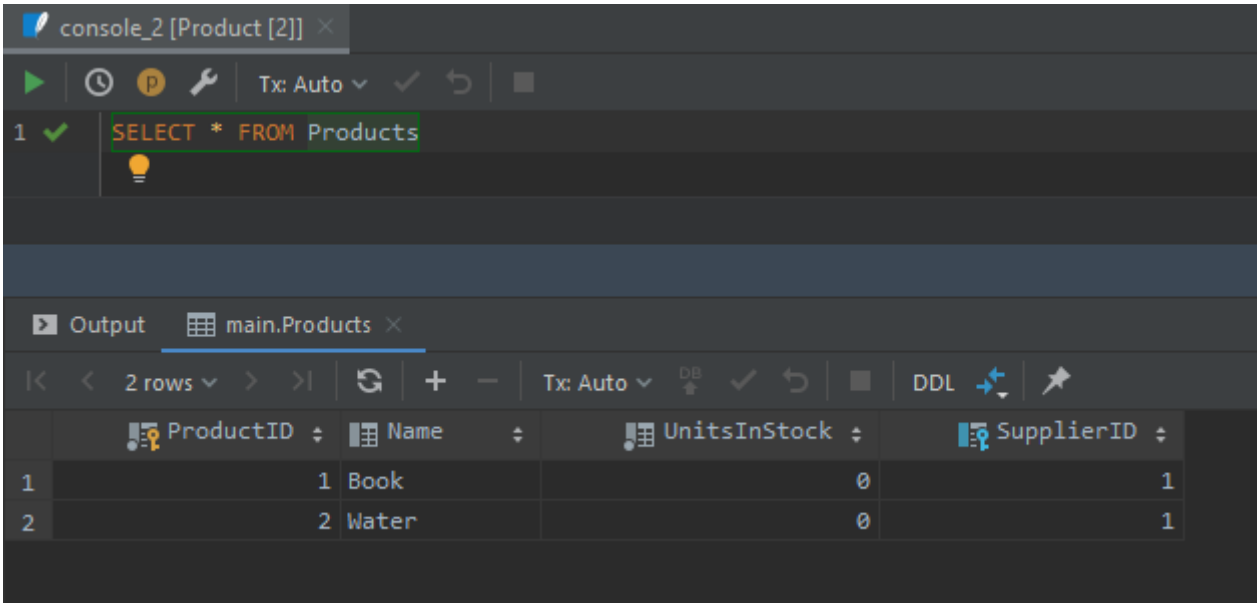
```
namespace AMazurProductEF
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            Console.WriteLine("Enter product name: ");
            String prodName = Console.ReadLine();
            // Product product = new Product { Name = prodName };

            Console.WriteLine("Enter supplier name: ");
            String supName = Console.ReadLine();
            // Supplier supplier = new Supplier { CompanyName = supName };

            ProdContext context = new ProdContext();


            Product product = context.Products.Where(p => p.Name == prodName).FirstOrDefault();
            Supplier supplier = context.Suppliers.Where(s => s.CompanyName == supName).FirstOrDefault();

            context.Entry(product).Property("SupplierID").CurrentValue = supplier.SupplierID;
            context.SaveChanges();

            IQueryable<string> query = context.Products.Select(p => p.Name);

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

Produkty przed dodaniem dostawcy:



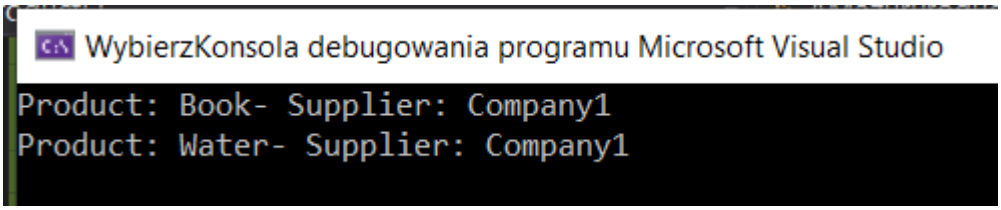

Produkty po przypisaniu dostawcy:



**g)**

Wyświetlono wszystkie produkty wraz z nazwą dostawcy.

```
ProdContext context = new ProdContext();
foreach (Product product in context.Products)
{
    context.Entry(product).Reference(prod => prod.Supplier).Load();

    if (product.Supplier != null)
    {
        Console.WriteLine("Product: "+ product.Name + "- Supplier: " +  product.Supplier.CompanyName);
    }
}
```



## Zadanie III

Odwrócono relację zgodnie z poniższym schematem.


Z klasy *Product* usunięto wcześniej dodane pole *Supplier*. Natomiast do klasy *Supplier* dodano listę produktów, dostarczanych przez danego dostawcę.

```
AMazurProductEF                                   AMazurProductEF.Product
1   using System;
2   using System.Collections.Generic;
3   using System.Text;
4
5   namespace AMazurProductEF
6   {
        Odwołania: 6
7       class Product
8       {
            Odwołania: 0
9           public int ProductID { get; set; }
10
            Odwołania: 2
11          public string Name { get; set; }
12
            Odwołania: 0
13          public int UnitsInStock { get; set; }
14
15      }
16  }
```

Supplier.cs | Product.cs | Program.cs

AMazurProductEF | AMazurProductEF.Supplier

```
1   using System;
2   using System.Collections.Generic;
3   using System.Text;
4
5   namespace AMazurProductEF
6   {
        Odwołania: 3
7       class Supplier
8       {
            Odwołania: 0
9           public Supplier()
10          {
11              Products = new List<Product>();
12          }
13
            Odwołania: 0
14          public int SupplierID { get; set; }
            Odwołania: 0
15          public string CompanyName { get; set; }
16
            Odwołania: 0
17          public string Street { get; set; }
18
            Odwołania: 0
19          public string City { get; set; }
20
            1 odwołanie
21          public List<Product> Products { get; set; }
22      }
23  }
```

**a), b)**

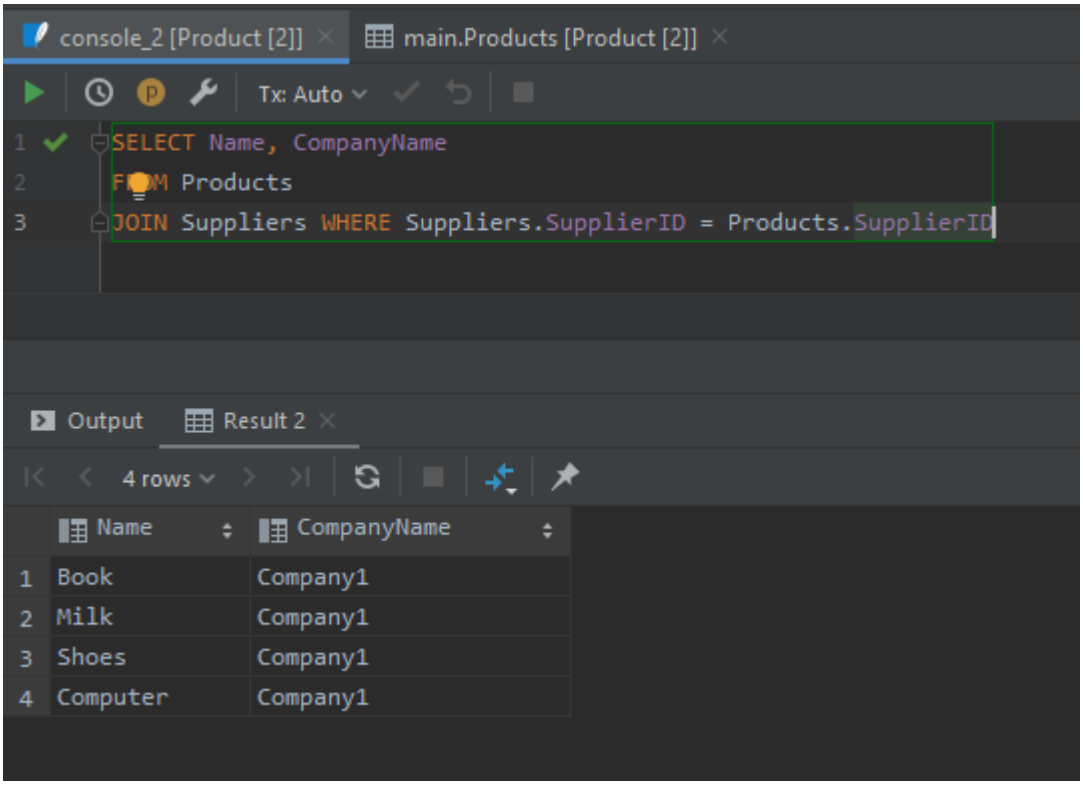
Dodano do bazy nowe produkty i stworzono dostawcę. Następnie znaleziono wcześniej wprowadzone produkty i dodano je do produktów dostarczanych przez nowo stworzonego dostawcę.

```
class Program
{
    Odwołania: 0
    static void Main(string[] args)
    {
        ProdContext context = new ProdContext();
        Supplier supplier = context.Suppliers.Where(s => s.CompanyName == "Company1").FirstOrDefault();
        foreach (Product p in context.Products)
        {
            supplier.Products.Add(p);
            context.SaveChanges();
        }
```

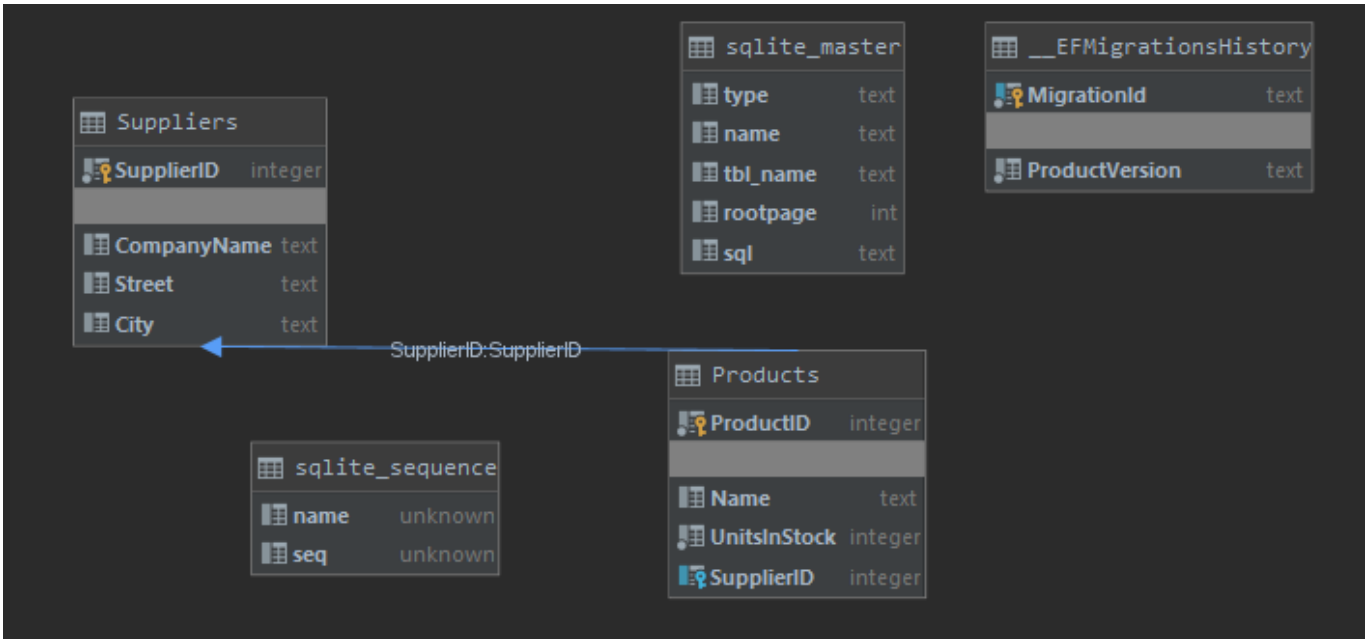
Wypisano wszystkie produkty dostarczane przez dostawcę.

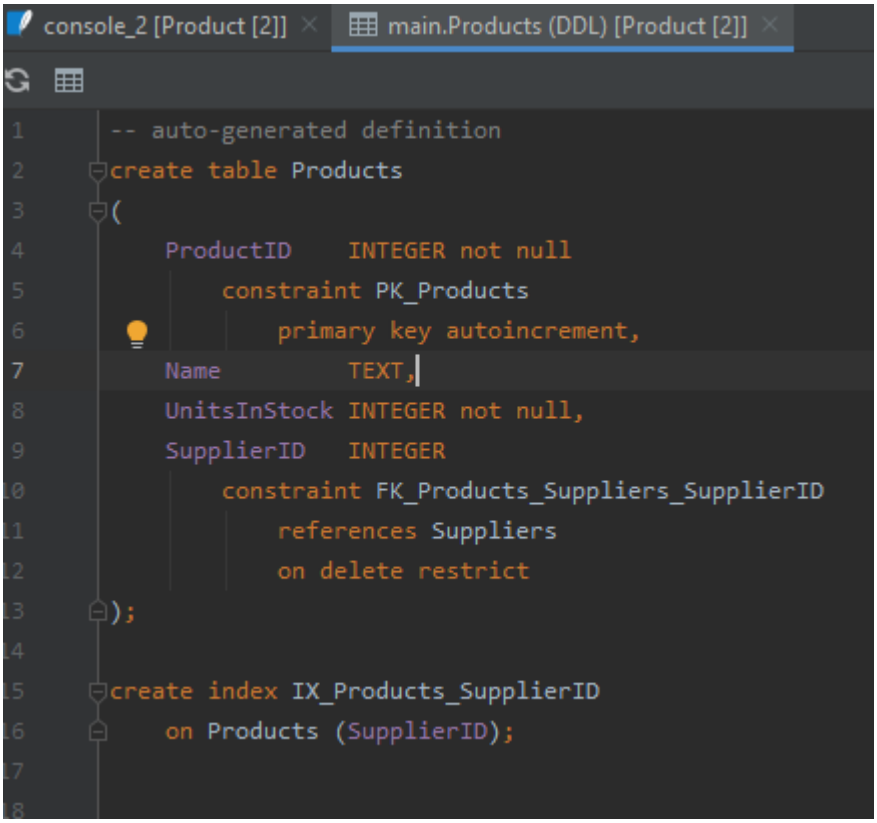
```
Odwołania: 0
class Program
{
    Odwołania: 0
    static void Main(string[] args)
    {
        ProdContext context = new ProdContext();
        var data = context.Suppliers.Include(s => s.Products).ToList();
        foreach (var s in data)
        {
            Console.WriteLine(s.CompanyName);
            foreach(var p in s.Products)
            {
                Console.WriteLine(p.Name);
            }
        }
```

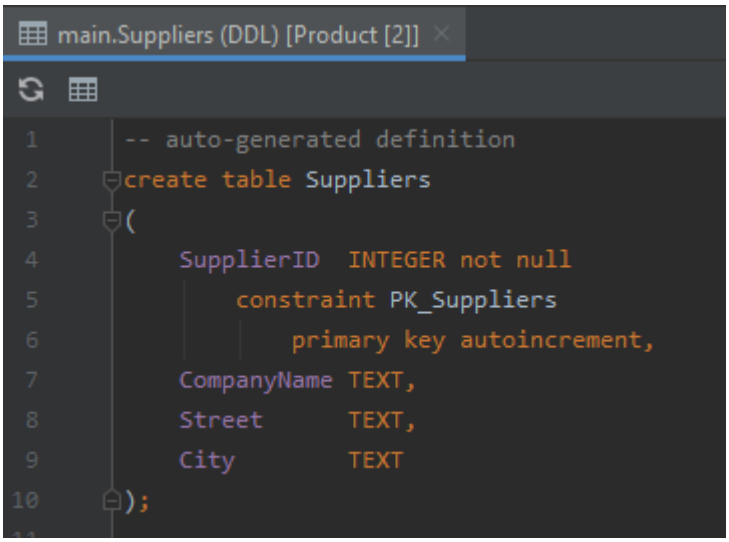
```
Konsola debugowania programu Microsoft Visual Studio
Company1
Book
Milk
Shoes
Computer
```



Schemat bazy danych wygląda jak poniżej:

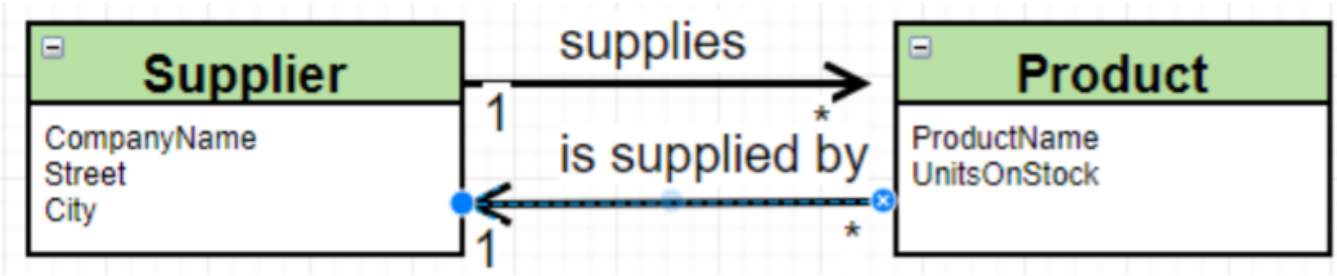



main.Suppliers (DDL) [Product [2]]
-- auto-generated definition
create table Suppliers
(
SupplierID INTEGER not null
constraint PK_Suppliers
primary key autoincrement,
CompanyName TEXT,
Street TEXT,
City TEXT
);

## Zadanie IV

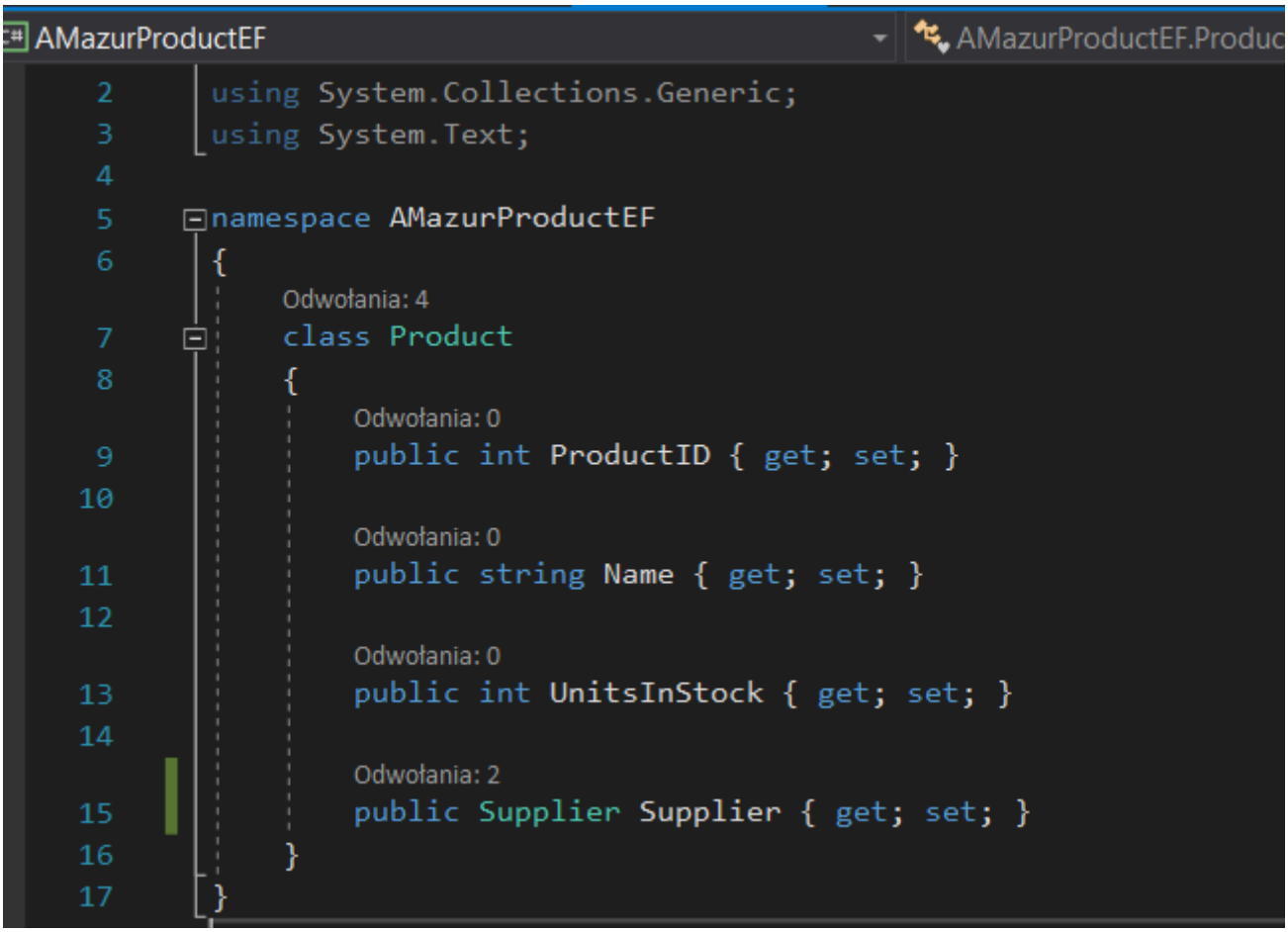
Zamodelowano relację dwustronną jak poniżej.




Klasa *Supplier* pozostała bez zmian (z listą dostarczanych przez dostawcę produktów). Z kolei do klasy *Product* dodano pole *Supplier*.

AMazurProductEF — AMazurProductEF.Supplier

```
1   using System;
2   using System.Collections.Generic;
3   using System.Text;
4
5   namespace AMazurProductEF
6   {
        Odwołania: 4
7       class Supplier
8       {
            Odwołania: 0
9           public Supplier()
10          {
11              Products = new List<Product>();
12          }
13
            1 odwołanie
14          public int SupplierID { get; set; }
            1 odwołanie
15          public string CompanyName { get; set; }
16
            Odwołania: 0
17          public string Street { get; set; }
18
            Odwołania: 0
19          public string City { get; set; }
20
            Odwołania: 2
21          public List<Product> Products { get; set; }
22      }
23  }
```

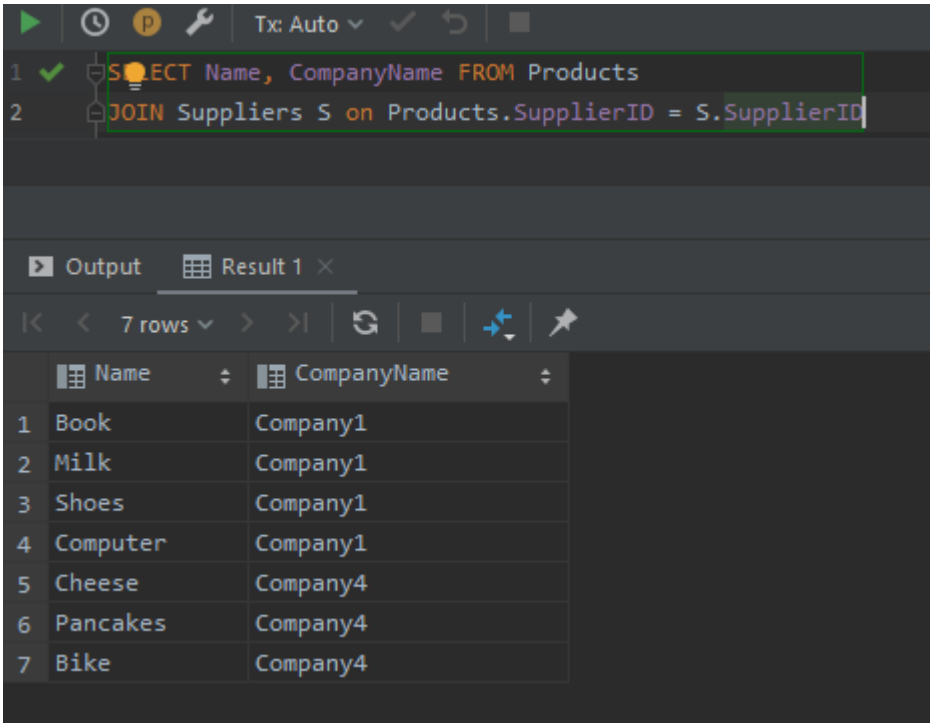



**a), b)**

Stworzono kilka produktów oraz dodano je do produktów dostarczanych przez nowo stworzonego dostawcę (pamiętając o dwustronności relacji).

```
ProdContext context = new ProdContext();
Supplier supplier = context.Suppliers.Where(s => s.CompanyName == "Company4").FirstOrDefault();
foreach (Product product in context.Products)
{
    context.Entry(product).Reference(prod => prod.Supplier).Load();

    if (product.Supplier == null)
    {
        context.Entry(product).Property("SupplierID").CurrentValue = supplier.SupplierID;
        supplier.Products.Add(product);
        context.SaveChanges();
    }
}
```

Następnie wyświetlono wszystkie produkty wraz z nazwą dostawcy.



```
SELECT Name, CompanyName FROM Products
JOIN Suppliers S on Products.SupplierID = S.SupplierID
```

Output | Result 1

7 rows

| | Name | CompanyName |
|---|---|---|
| 1 | Book | Company1 |
| 2 | Milk | Company1 |
| 3 | Shoes | Company1 |
| 4 | Computer | Company1 |
| 5 | Cheese | Company4 |
| 6 | Pancakes | Company4 |
| 7 | Bike | Company4 |

```
ProdContext context = new ProdContext();
foreach (Product product in context.Products)
{
    context.Entry(product).Reference(prod => prod.Supplier).Load();

    if (product.Supplier != null)
    {
        Console.WriteLine("Product: " + product.Name + " - Supplier: " + product.Supplier.CompanyName);
    }
}
```

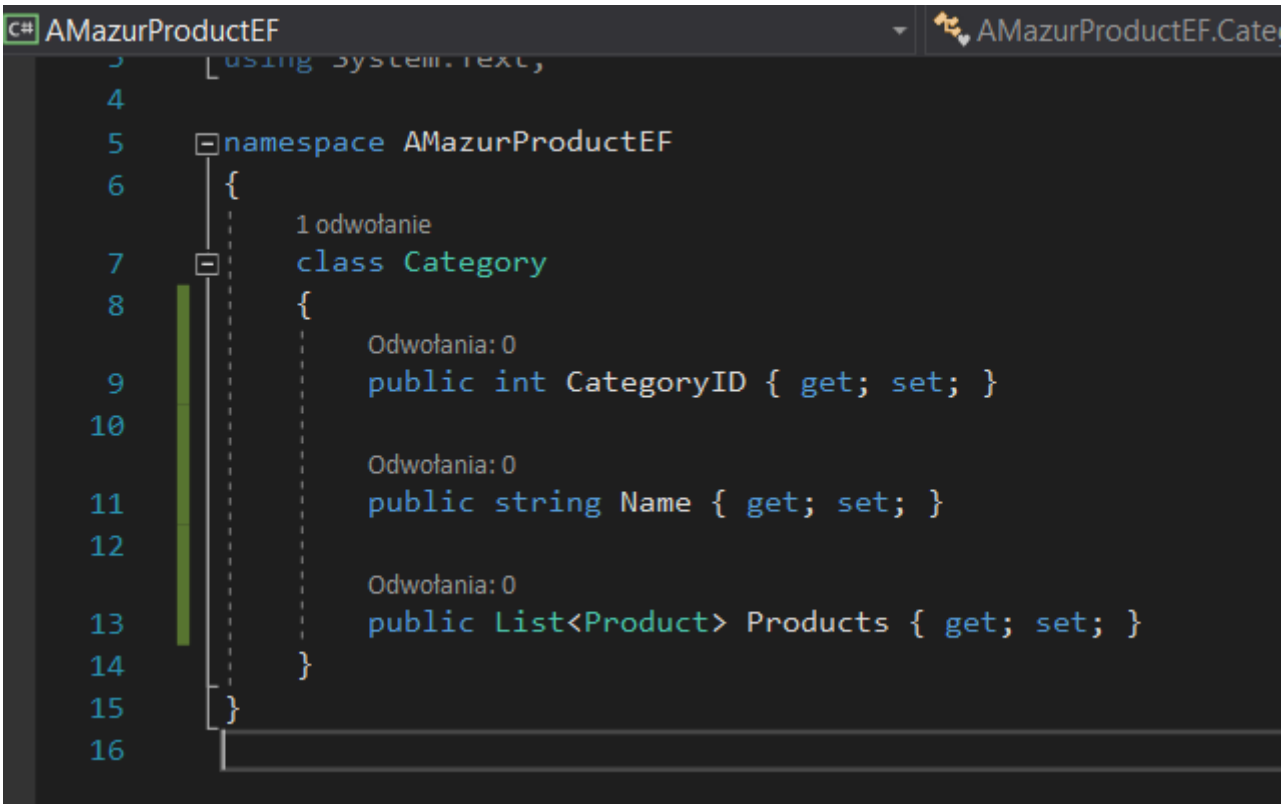
Konsola debugowania programu Microsoft Visual Studio

```
Product: Book - Supplier: Company1
Product: Milk - Supplier: Company1
Product: Shoes - Supplier: Company1
Product: Computer - Supplier: Company1
Product: Cheese - Supplier: Company4
Product: Pancakes - Supplier: Company4
Product: Bike - Supplier: Company4
```

## Zadanie V

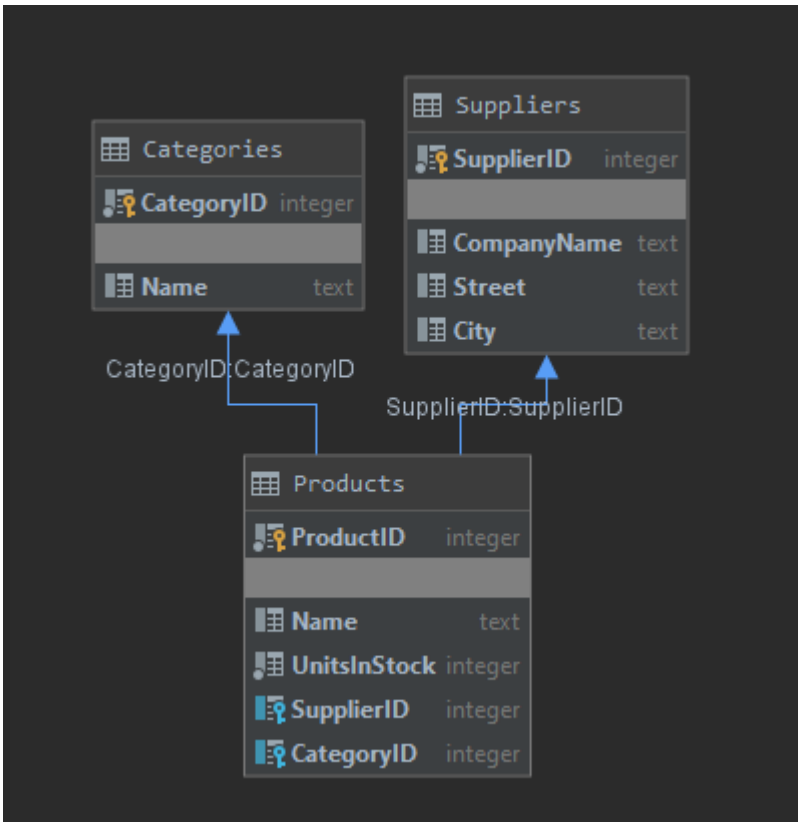
Dodano klasę *Category* z polami:

- int CategoryID
- string Name
- List <Product> Products




Schemat bazy danych wygląda jak poniżej.

```
-- auto-generated definition
create table Categories
(
    CategoryID INTEGER not null
        constraint PK_Categories
            primary key autoincrement,
    Name       TEXT
);
```




Do klasy *ProdContext* dodano kolejny *DbSet*, odwzorowujący zbiór kategorii.

Category.cs Supplier.cs ProdContext.cs Product.cs Program.cs*

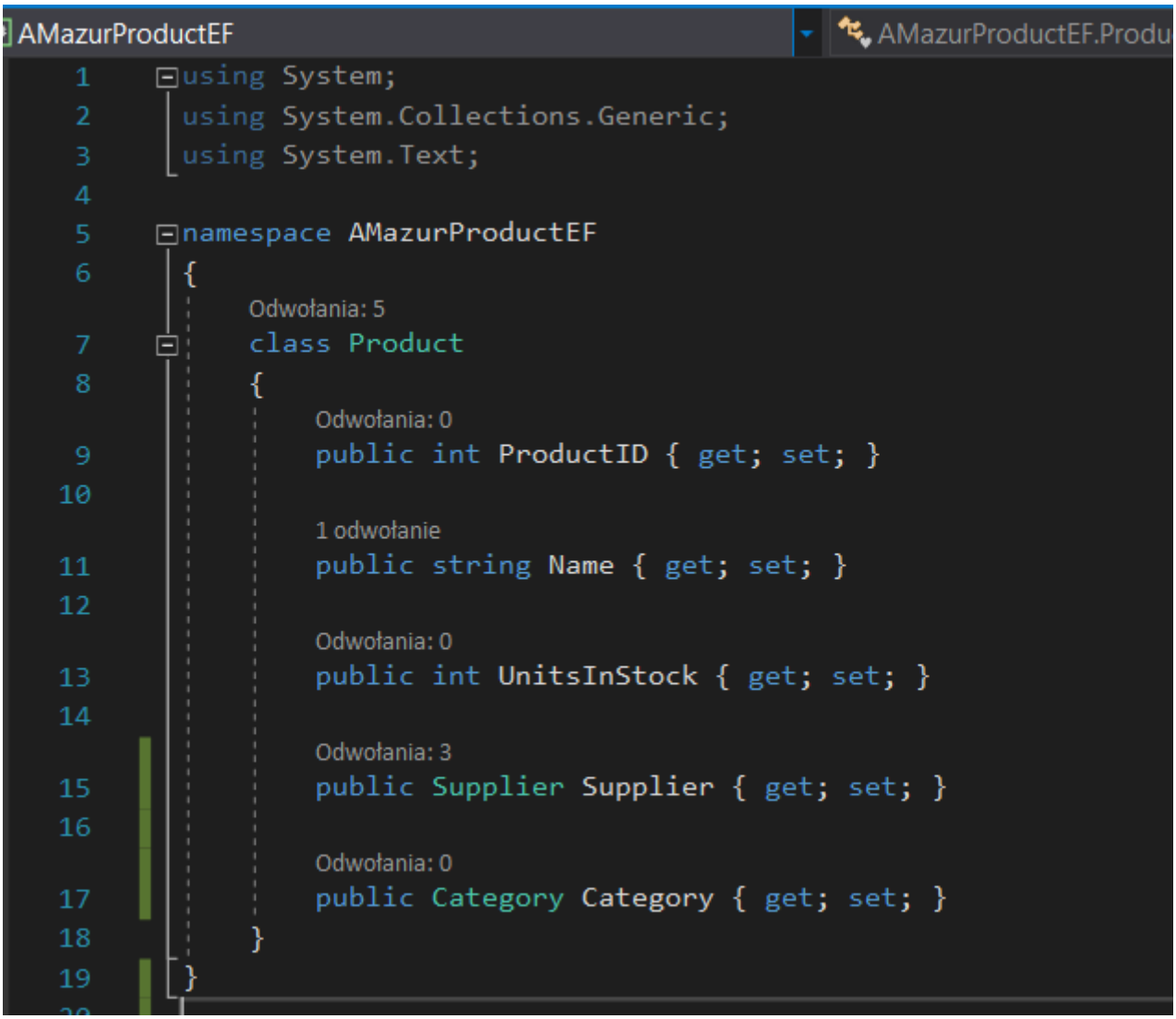
AMazurProductEF AMazurProductEF.ProdContext

```
2   using System;
3   using System.Collections.Generic;
4   using System.Text;
5
6   namespace AMazurProductEF
7   {
        Odwołania: 4
8       class ProdContext : DbContext
9       {
            Odwołania: 0
10          public DbSet<Product> Products { get; set; }
11
            Odwołania: 0
12          public DbSet<Supplier> Suppliers { get; set; }
13
            Odwołania: 0
14          public DbSet<Category> Categories { get; set; }
15
            Odwołania: 0
16          protected override void OnConfiguring(DbContextOptionsBuilder options) =>
17              options.UseSqlite("DataSource=Product.db");
18      }
19  }
20
```

**a)**

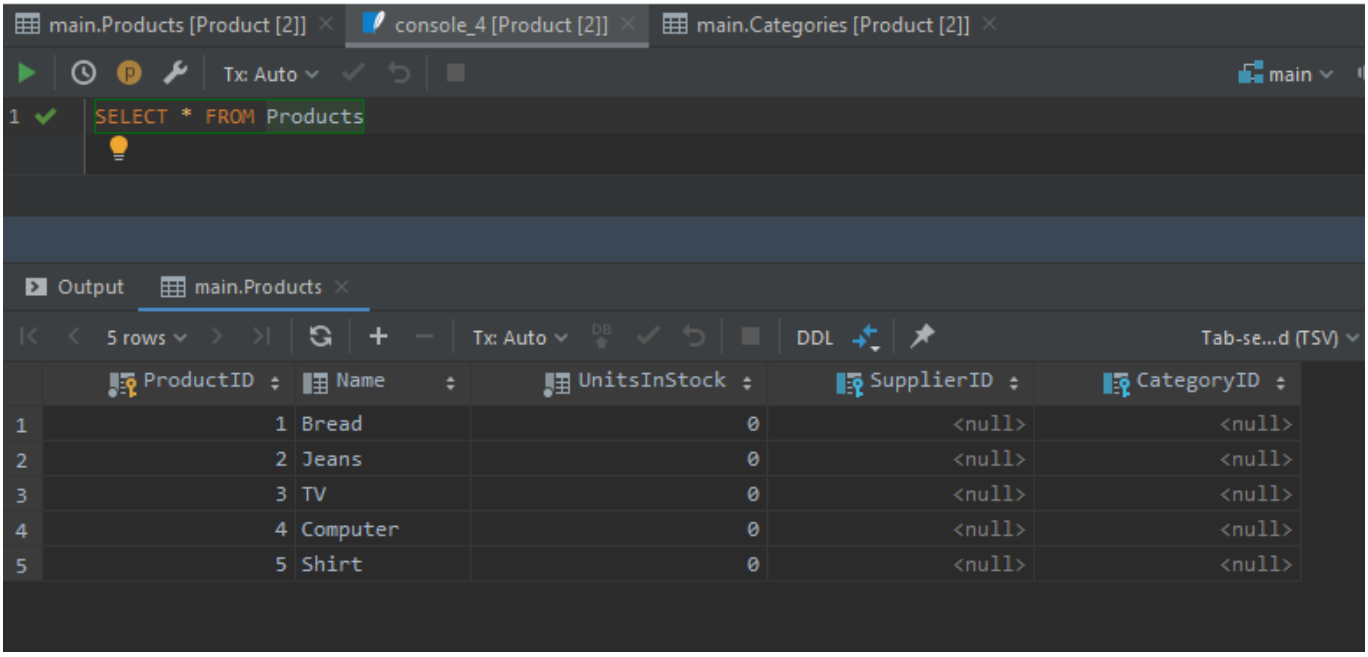
Zmodyfikowano klasę *Product*, dodając do niej pole *Category*.

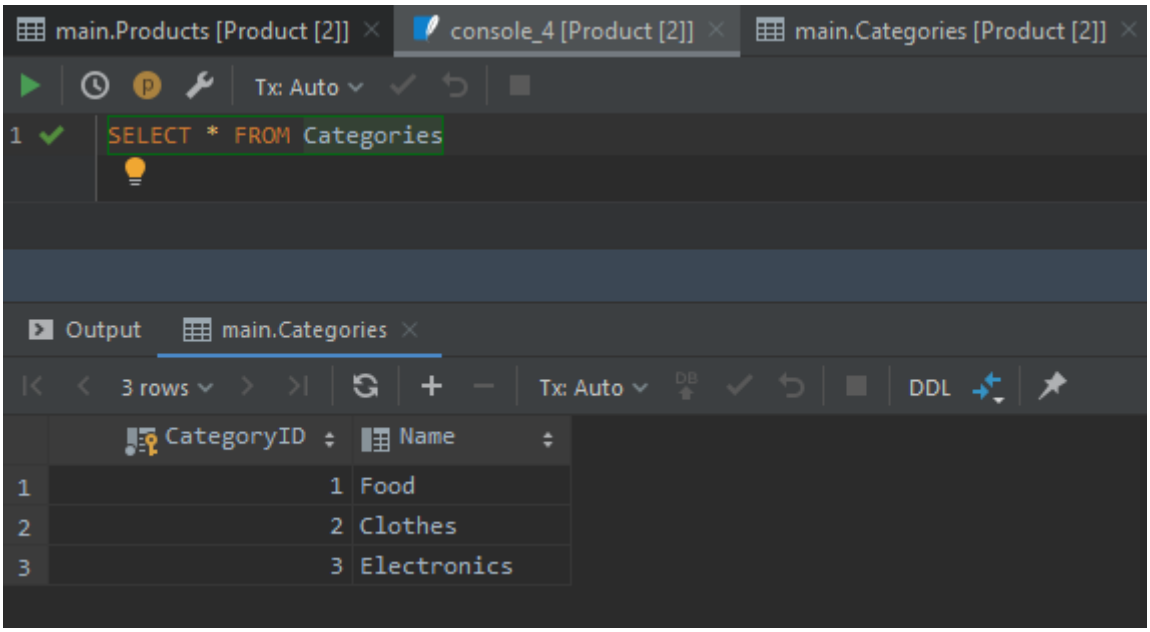
```
-- auto-generated definition
create table Products
(
    ProductID    INTEGER not null
        constraint PK_Products
            primary key autoincrement,
    Name         TEXT,
    UnitsInStock INTEGER not null,
    SupplierID   INTEGER
        constraint FK_Products_Suppliers_SupplierID
            references Suppliers
            on delete restrict,
    CategoryID   INTEGER
        constraint FK_Products_Categories_CategoryID
            references Categories
            on delete restrict
);

create index IX_Products_CategoryID
    on Products (CategoryID);

create index IX_Products_SupplierID
    on Products (SupplierID);
```


AMazurProductEF
AMazurProductEF.Produ
using System;
using System.Collections.Generic;
using System.Text;
namespace AMazurProductEF
{
Odwołania: 5
class Product
{
Odwołania: 0
public int ProductID { get; set; }
1 odwołanie
public string Name { get; set; }
Odwołania: 0
public int UnitsInStock { get; set; }
Odwołania: 3
public Supplier Supplier { get; set; }
Odwołania: 0
public Category Category { get; set; }
}
}

**b)**

Stworzono kilka produktów i kilka kategorii.

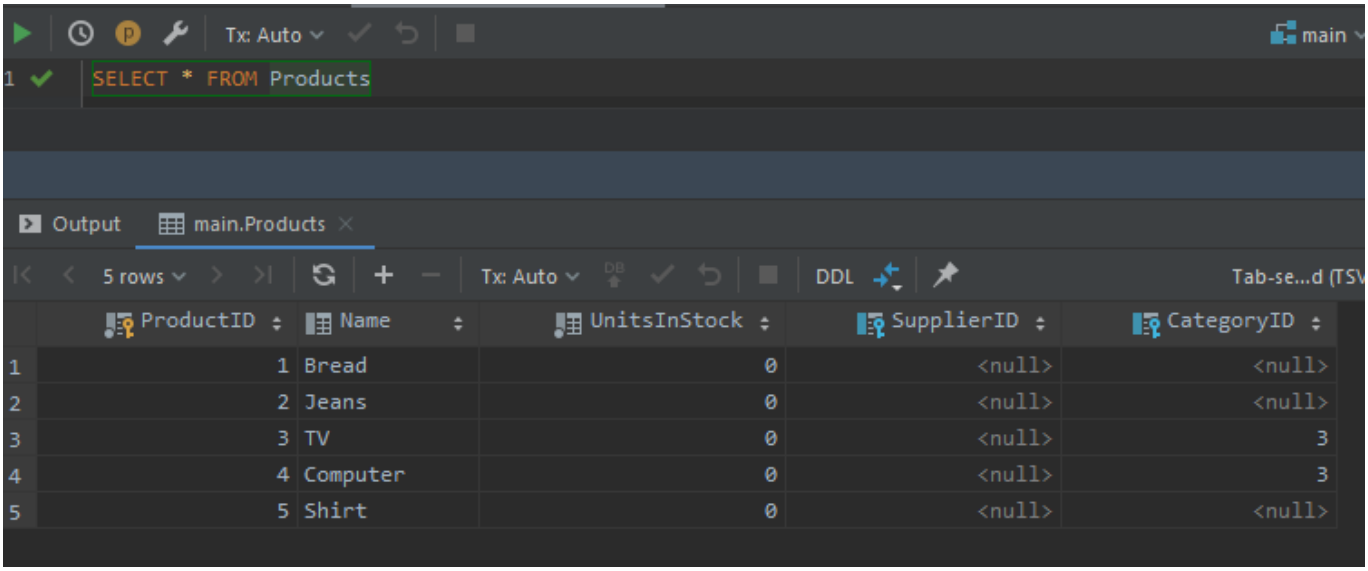


**c)**

Dodano kilka produktów do wybranej kategorii.

```
ProdContext context = new ProdContext();
Category category = context.Categories.Where(s => s.Name == "Electronics").FirstOrDefault();
Product prod1 = context.Products.Where(s => s.Name == "TV").FirstOrDefault();
Product prod2 = context.Products.Where(s => s.Name == "Computer").FirstOrDefault();

context.Entry(prod1).Reference(prod => prod.Category).Load();

if (prod1.Category == null)
{
    context.Entry(prod1).Property("CategoryID").CurrentValue = category.CategoryID;
    category.Products.Add(prod1);
    context.SaveChanges();
}

context.Entry(prod2).Reference(prod => prod.Category).Load();

if (prod2.Category == null)
{
    context.Entry(prod2).Property("CategoryID").CurrentValue = category.CategoryID;
    category.Products.Add(prod2);
    context.SaveChanges();
}
```



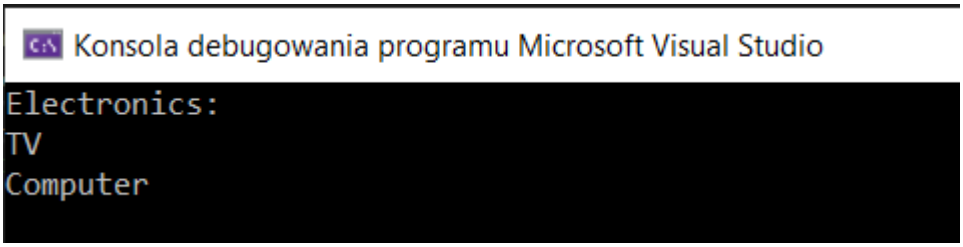

| | ProductID | Name | UnitsInStock | SupplierID | CategoryID |
|---|---|---|---|---|---|
| 1 | 1 | Bread | 0 | <null> | <null> |
| 2 | 2 | Jeans | 0 | <null> | <null> |
| 3 | 3 | TV | 0 | <null> | 3 |
| 4 | 4 | Computer | 0 | <null> | 3 |
| 5 | 5 | Shirt | 0 | <null> | <null> |

**d)**

Wypisano wszystkie produkty należące do kategorii *Electronics*.

```
ProdContext context = new ProdContext();
Category category = context.Categories.Include(c => c.Products).ToList().Where(s => s.Name == "Electronics").FirstOrDefault();
Console.WriteLine(category.Name + ": ");
foreach (Product p in category.Products)
{
    Console.WriteLine(p.Name);
}
```

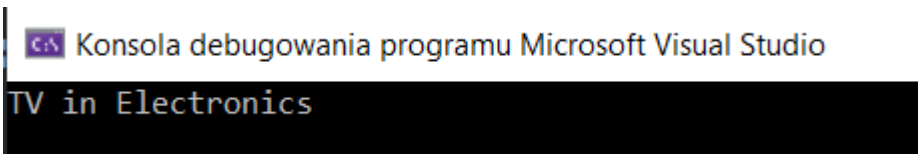
Jak widać produkty zostały wypisane poprawnie:




Następnie wypisano kategorię, do której należy *TV*.

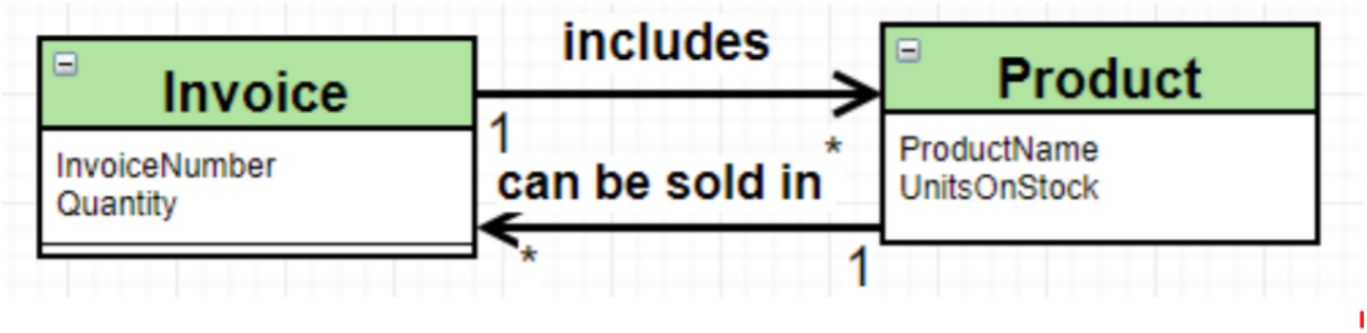
```
ProdContext context = new ProdContext();

Product prod = context.Products.Where(s => s.Name == "TV").FirstOrDefault();

context.Entry(prod).Reference(prod => prod.Category).Load();

if (prod.Category != null)
{
    Console.WriteLine(prod.Name + " in " + prod.Category.Name);
}
```

Kategoria produktu również została poprawnie wypisana:



## Zadanie VI

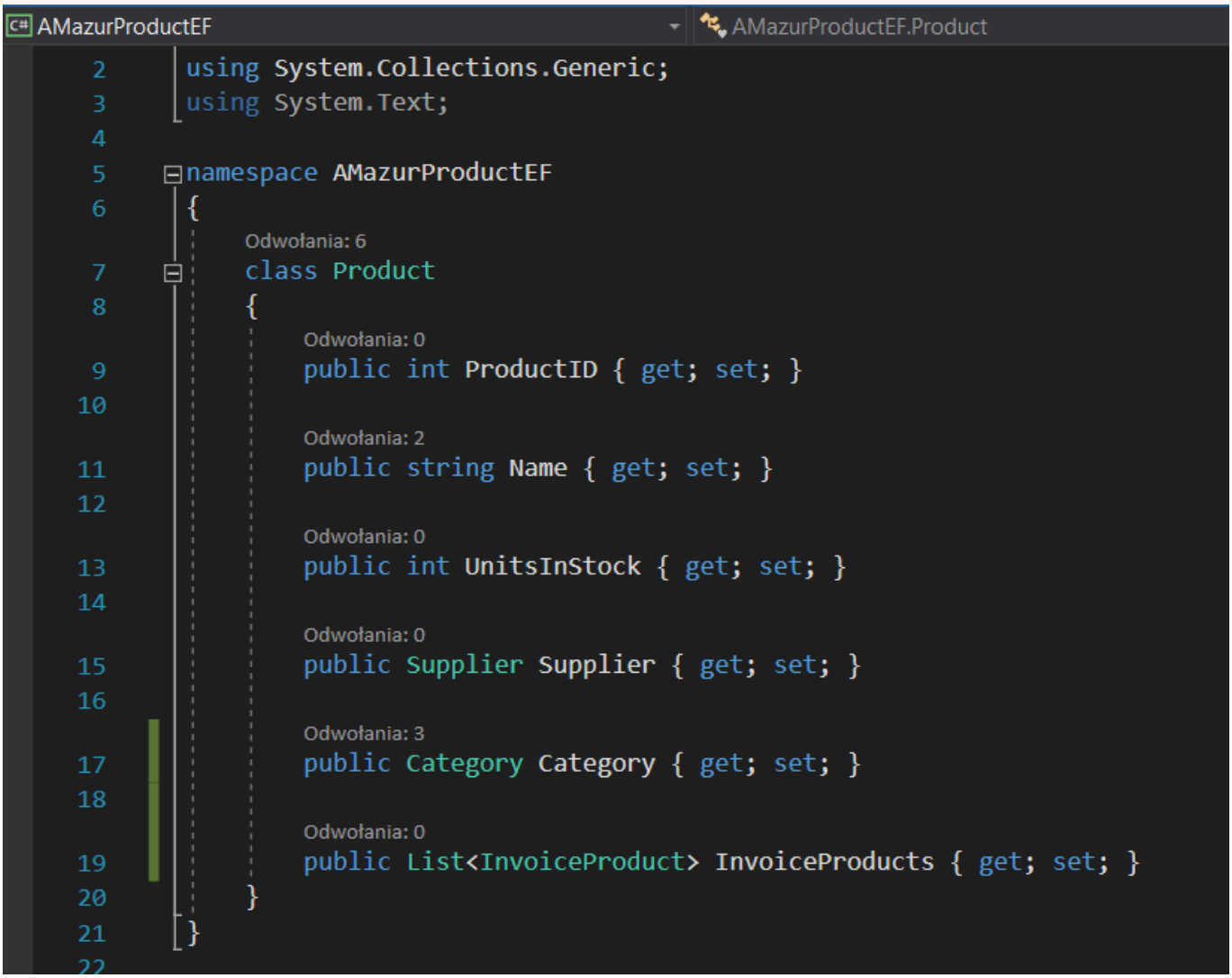
Zmodyfikowano relację wiele-do-wielu, jak poniżej.

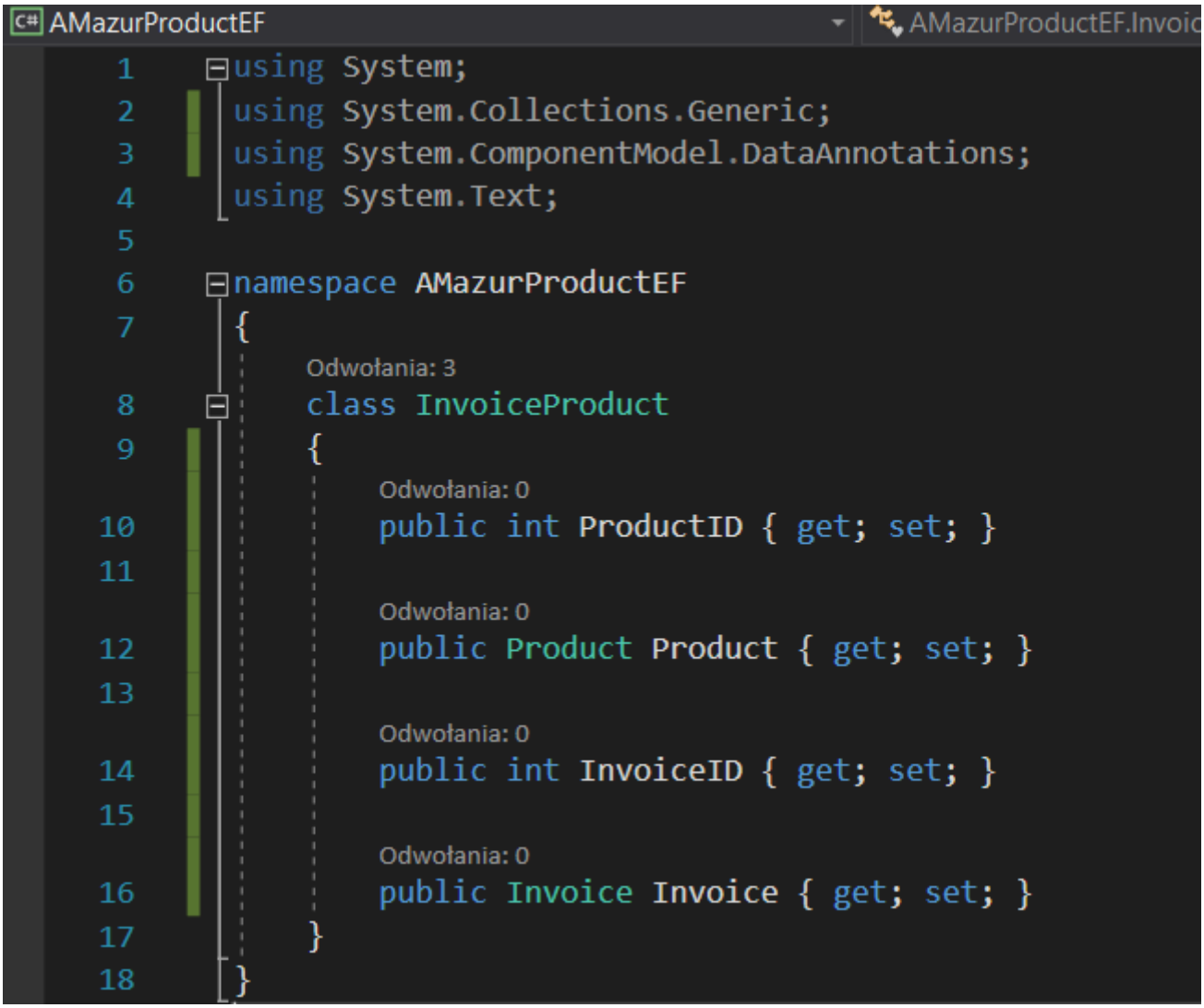



W celu wykonania powyższej relacji konieczne było stworzenie nowej klasy *InvoiceProduct*, przechowującej relacje między *Invoice* a *Product*. Co więcej do klasy *Product* i *Invoice* dodano listę obiektów *InvoiceProducts*.

AMazurProductEF — AMazurProductEF.Invoice

```
1   using System;
2   using System.Collections.Generic;
3   using System.Text;
4
5   namespace AMazurProductEF
6   {
        1 odwołanie
7       class Invoice
8       {
            Odwołania: 0
9           public Invoice()
10          {
11              InvoiceProducts = new List<InvoiceProduct>();
12          }
13
            Odwołania: 0
14          public int InvoiceID { get; set; }
15
            Odwołania: 0
16          public int InvoiceNumber { get; set; }
17
            Odwołania: 0
18          public int Quantity { get; set; }
19
20          public List<InvoiceProduct> InvoiceProducts { get; set; }
21      }
22  }
```

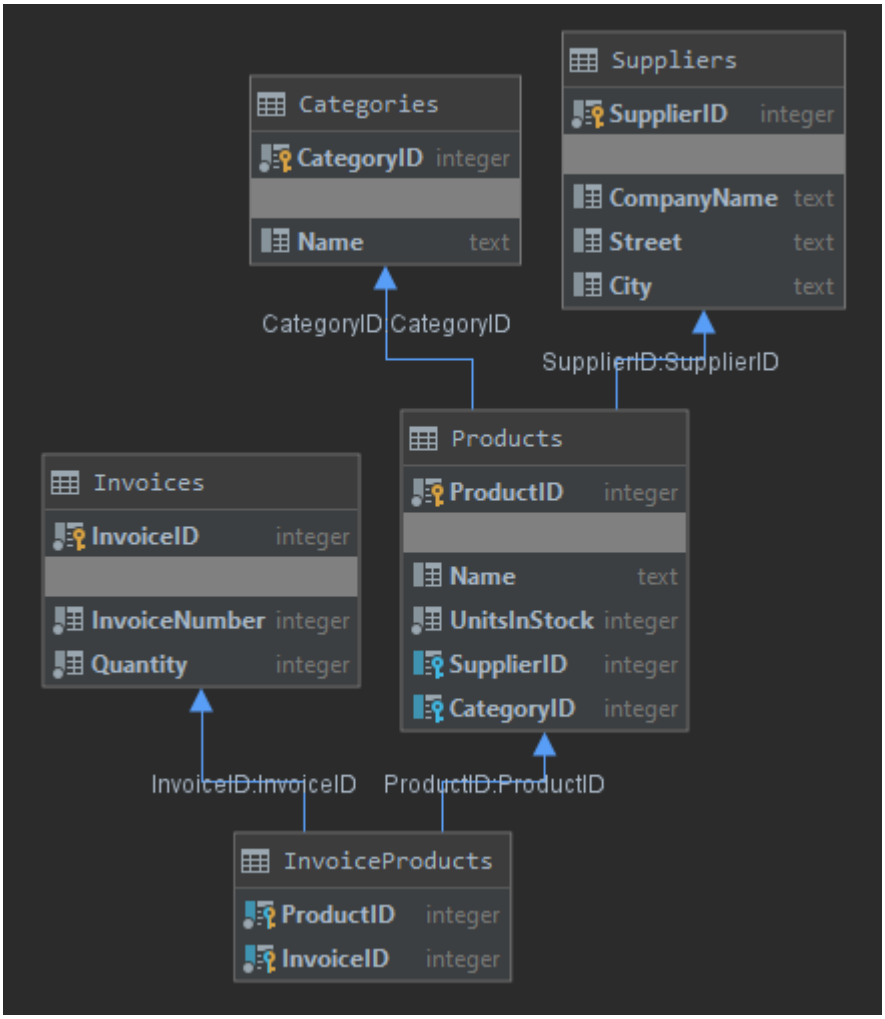

AMazurProductEF
AMazurProductEF.Product
using System.Collections.Generic;
using System.Text;
namespace AMazurProductEF
{
Odwołania: 6
class Product
{
Odwołania: 0
public int ProductID { get; set; }
Odwołania: 2
public string Name { get; set; }
Odwołania: 0
public int UnitsInStock { get; set; }
Odwołania: 0
public Supplier Supplier { get; set; }
Odwołania: 3
public Category Category { get; set; }
Odwołania: 0
public List<InvoiceProduct> InvoiceProducts { get; set; }
}
}


AMazurProductEF
AMazurProductEF.Invoic
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Text;
namespace AMazurProductEF
{
Odwołania: 3
class InvoiceProduct
{
Odwołania: 0
public int ProductID { get; set; }
Odwołania: 0
public Product Product { get; set; }
Odwołania: 0
public int InvoiceID { get; set; }
Odwołania: 0
public Invoice Invoice { get; set; }
}
}

Do klasy *ProdContext* dodano jeden *DbSet* odzwierciedlający zbiór faktur oraz drugi - przedstawiający zbiór relacji między produktami a fakturami. Nadpisano również metodę *OnModelCreating*.

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

namespace AMazurProductEF
{
    Odwołania: 4
    class ProdContext : DbContext
    {
        1 odwołanie
        public DbSet<Product> Products { get; set; }

        Odwołania: 0
        public DbSet<Supplier> Suppliers { get; set; }

        Odwołania: 0
        public DbSet<Category> Categories { get; set; }

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

        Odwołania: 0
        public DbSet<InvoiceProduct> InvoiceProducts { get; set; }

        Odwołania: 0
        protected override void OnConfiguring(DbContextOptionsBuilder options) =>
            options.UseSqlite("DataSource=Product.db");

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

Schemat bazy danych wygląda następująco:




```
-- auto-generated definition
create table Invoices
(
    InvoiceID     INTEGER not null
        constraint PK_Invoices
            primary key autoincrement,
    InvoiceNumber INTEGER not null,
    Quantity      INTEGER not null
);
```

```
-- auto-generated definition
create table InvoiceProducts
(
    ProductID INTEGER not null
        constraint FK_InvoiceProducts_Products_ProductID
            references Products
            on delete cascade,
    InvoiceID INTEGER not null
        constraint FK_InvoiceProducts_Invoices_InvoiceID
            references Invoices
            on delete cascade,
    constraint PK_InvoiceProducts
        primary key (ProductID, InvoiceID)
);

create index IX_InvoiceProducts_InvoiceID
    on InvoiceProducts (InvoiceID);
```

**a)**

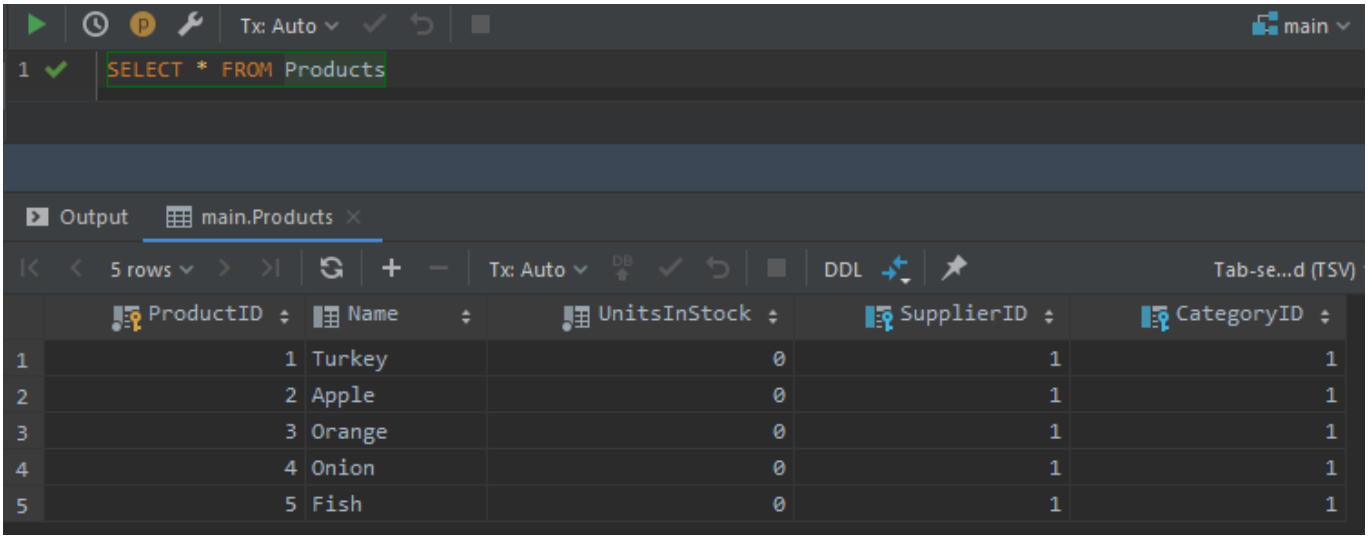
Stworzono kilka produktów, dodano kategorię, dostawcę i "sprzedano" dane produkty na kilku transakcjach.

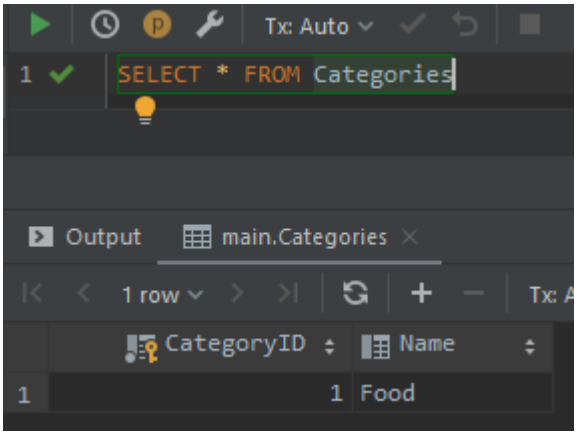
```
using System;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Design;

namespace AMazurProductEF
{
    class Program
    {
        static void Main(string[] args)
        {
            ProdContext context = new ProdContext();
            // create products, category, supplier and invoices
            Product product1 = new Product { Name = "Turkey" };
            Product product2 = new Product { Name = "Apple" };
            Product product3 = new Product { Name = "Orange" };
            Product product4 = new Product { Name = "Onion" };
            Product product5 = new Product { Name = "Fish" };
            Supplier supplier = new Supplier { CompanyName = "FoodCompany" };
            Category category = new Category { Name = "Food" };
            Invoice invoice1 = new Invoice { InvoiceNumber = 1, Quantity = 3 };
            Invoice invoice2 = new Invoice { InvoiceNumber = 2, Quantity = 3 };

            // add category to products and products to category
            category.Products.Add(product1);
            category.Products.Add(product2);
            category.Products.Add(product3);
            category.Products.Add(product4);
            category.Products.Add(product5);
```

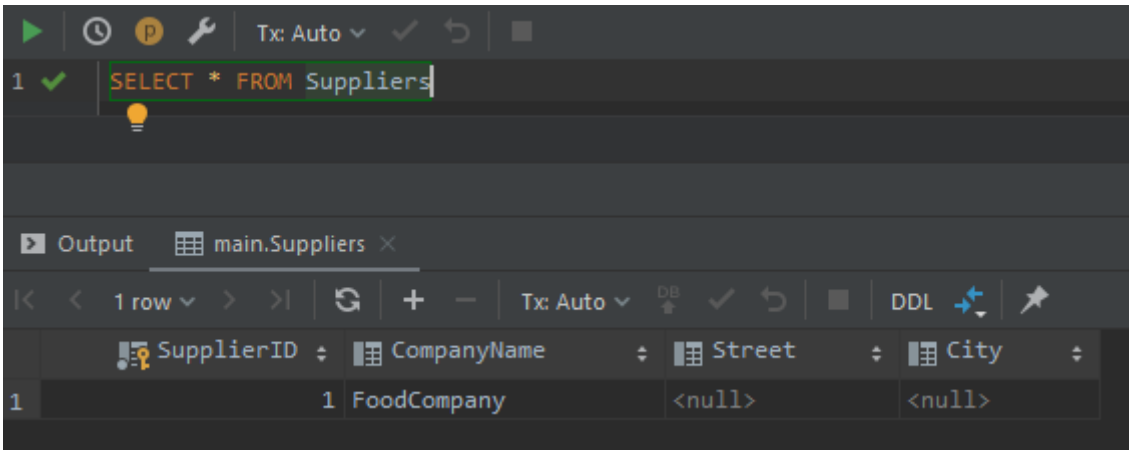
```
            product1.Category = category;
            product2.Category = category;
            product3.Category = category;
            product4.Category = category;
            product5.Category = category;

            // add supplier to products and products to supplier
            supplier.Products.Add(product1);
            supplier.Products.Add(product2);
            supplier.Products.Add(product3);
            supplier.Products.Add(product4);
            supplier.Products.Add(product5);
            product1.Supplier = supplier;
            product2.Supplier = supplier;
            product3.Supplier = supplier;
            product4.Supplier = supplier;
            product5.Supplier = supplier;

            // add products, category, supplier and invoices to ProdContext
            context.Products.Add(product1);
            context.Products.Add(product2);
            context.Products.Add(product3);
            context.Products.Add(product4);
            context.Products.Add(product5);
            context.Categories.Add(category);
            context.Suppliers.Add(supplier);
            context.Invoices.Add(invoice1);
            context.Invoices.Add(invoice2);

            // create invoiceProducts
            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 =
invoice1, Product = product3 };
            invoice1.InvoiceProducts.Add(invoiceProduct3);
            InvoiceProduct invoiceProduct4 = new InvoiceProduct { Invoice =
invoice2, Product = product3 };
            invoice2.InvoiceProducts.Add(invoiceProduct4);
            InvoiceProduct invoiceProduct5 = new InvoiceProduct { Invoice =
invoice2, Product = product4 };
            invoice2.InvoiceProducts.Add(invoiceProduct5);
            InvoiceProduct invoiceProduct6 = new InvoiceProduct { Invoice =
invoice2, Product = product5 };
            invoice2.InvoiceProducts.Add(invoiceProduct6);

            // add invoiceProducts to ProdContext
            context.InvoiceProducts.Add(invoiceProduct1);
            context.InvoiceProducts.Add(invoiceProduct2);
            context.InvoiceProducts.Add(invoiceProduct3);
            context.InvoiceProducts.Add(invoiceProduct4);
```

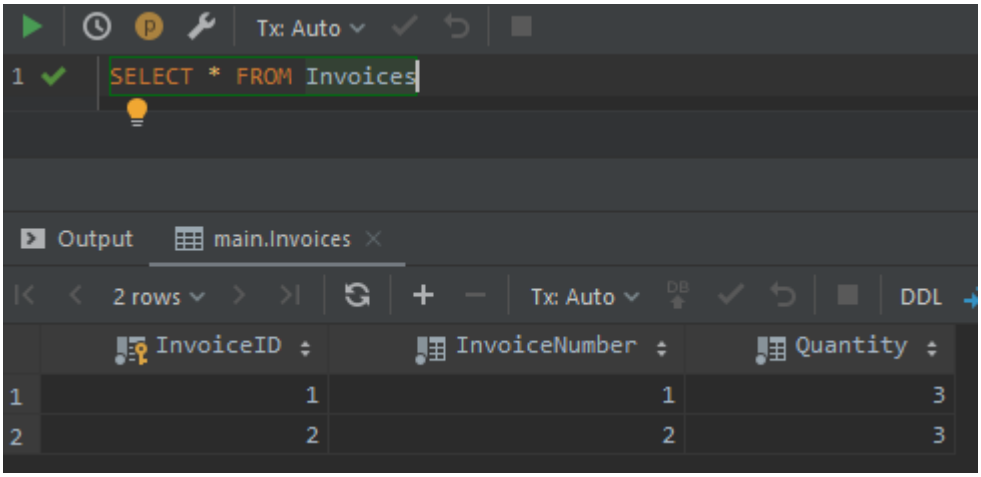
```
            context.InvoiceProducts.Add(invoiceProduct5);
            context.InvoiceProducts.Add(invoiceProduct6);

            product1.InvoiceProducts.Add(invoiceProduct1);
            product2.InvoiceProducts.Add(invoiceProduct2);
            product3.InvoiceProducts.Add(invoiceProduct3);
            product3.InvoiceProducts.Add(invoiceProduct4);
            product4.InvoiceProducts.Add(invoiceProduct5);
            product5.InvoiceProducts.Add(invoiceProduct6);

            context.SaveChanges();
        }
    }
}
```

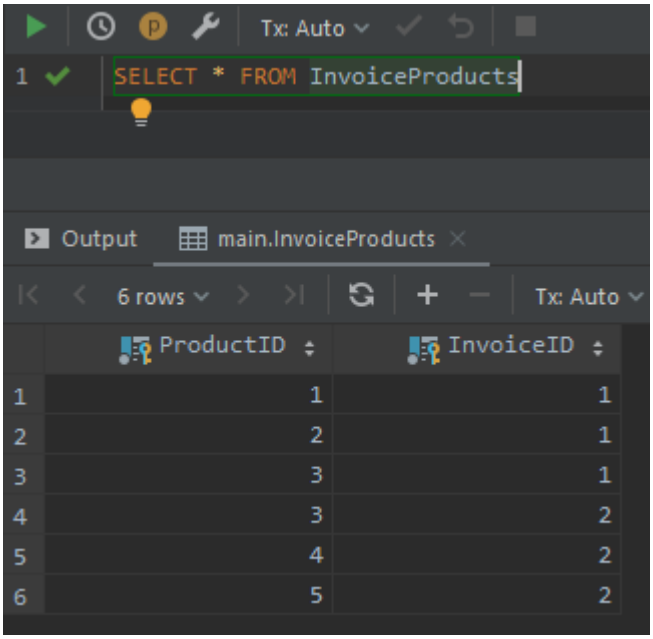
Poniżej widać, że dane poprawnie dodały się do bazy.

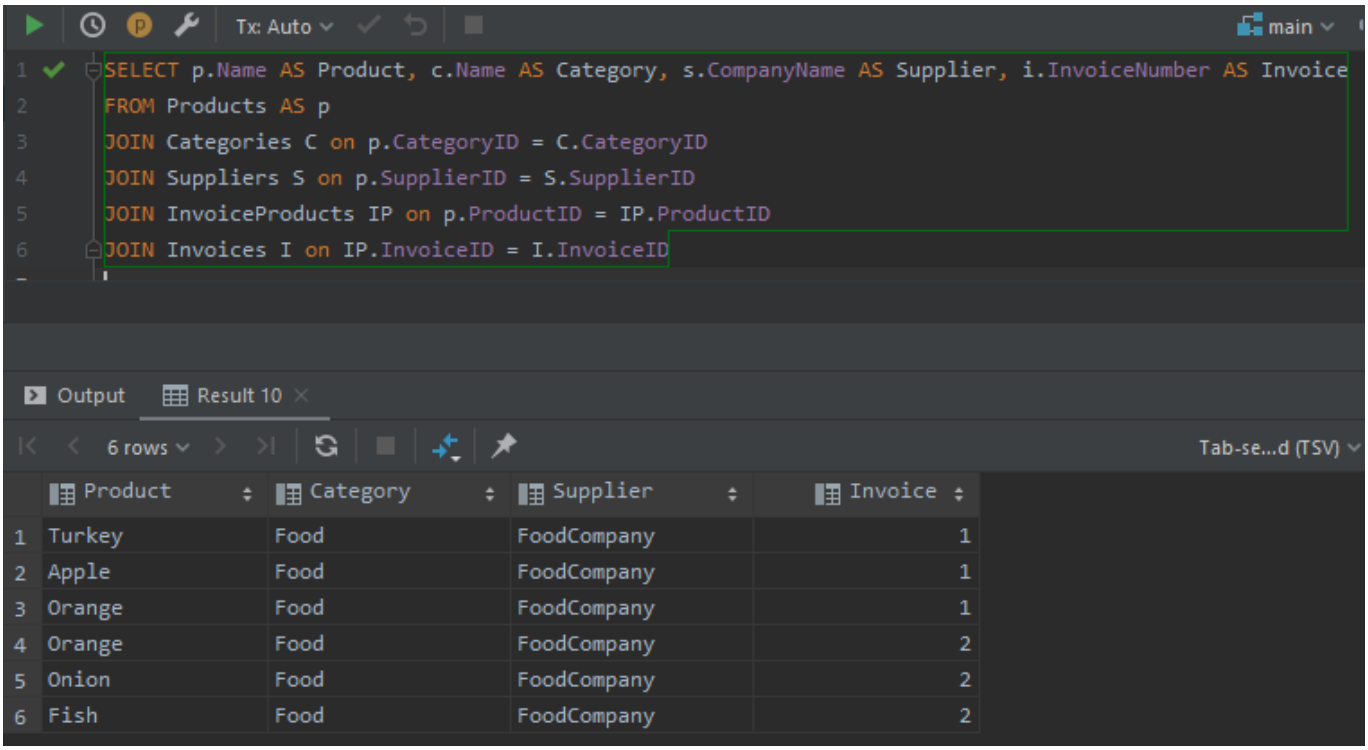



Tx: Auto
1
SELECT * FROM Suppliers
Output
main.Suppliers
1 row
Tx: Auto
DDL
SupplierID
CompanyName
Street
City
1
1
FoodCompany
<null>
<null>


Tx: Auto
1
SELECT * FROM Invoices
Output
main.Invoices
2 rows
Tx: Auto
DDL
InvoiceID
InvoiceNumber
Quantity
1
1
1
3
2
2
2
3


Tx: Auto
1
SELECT * FROM InvoiceProducts
Output
main.InvoiceProducts
6 rows
Tx: Auto
ProductID
InvoiceID
1
1
1
2
1
3
1
3
2
4
2
5
2
6
2



```
SELECT p.Name AS Product, c.Name AS Category, s.CompanyName AS Supplier, i.InvoiceNumber AS Invoice
FROM Products AS p
JOIN Categories C on p.CategoryID = C.CategoryID
JOIN Suppliers S on p.SupplierID = S.SupplierID
JOIN InvoiceProducts IP on p.ProductID = IP.ProductID
JOIN Invoices I on IP.InvoiceID = I.InvoiceID
```

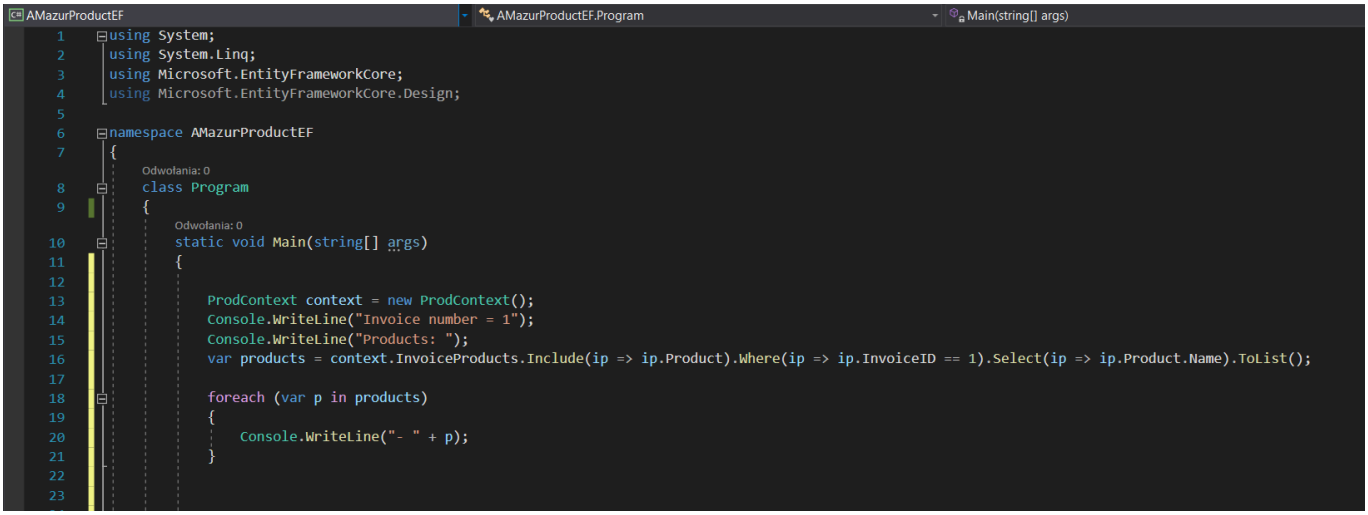
Output | Result 10

6 rows | Tab-se...d (TSV)

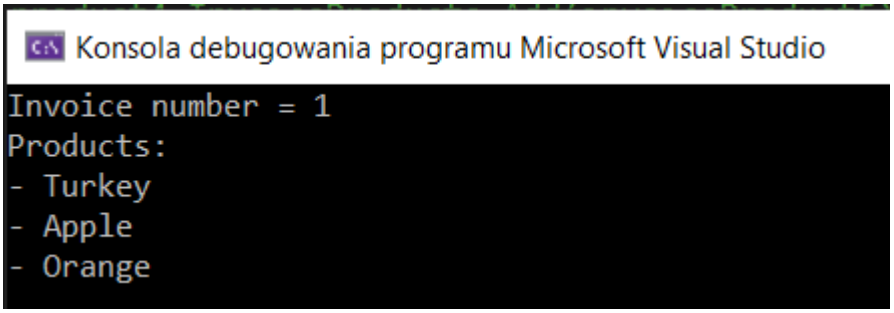
| | Product | Category | Supplier | Invoice |
|---|---|---|---|---|
| 1 | Turkey | Food | FoodCompany | 1 |
| 2 | Apple | Food | FoodCompany | 1 |
| 3 | Orange | Food | FoodCompany | 1 |
| 4 | Orange | Food | FoodCompany | 2 |
| 5 | Onion | Food | FoodCompany | 2 |
| 6 | Fish | Food | FoodCompany | 2 |

**b)**

Wypisano produkty sprzedane w ramach wybranej faktury/transakcji.



```
using System;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Design;

namespace AMazurProductEF
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {

            ProdContext context = new ProdContext();
            Console.WriteLine("Invoice number = 1");
            Console.WriteLine("Products: ");
            var products = context.InvoiceProducts.Include(ip => ip.Product).Where(ip => ip.InvoiceID == 1).Select(ip => ip.Product.Name).ToList();

            foreach (var p in products)
            {
                Console.WriteLine("- " + p);
            }
```



```
Konsola debugowania programu Microsoft Visual Studio
Invoice number = 1
Products:
- Turkey
- Apple
- Orange
```

**d)**

Wypisano faktury, w ramach których był sprzedany wybrany produkt.

```
AMazurProductEF                    AMazurProductEF.Program                    Main(string[] args)
1   using System;
2   using System.Linq;
3   using Microsoft.EntityFrameworkCore;
4   using Microsoft.EntityFrameworkCore.Design;
5
6   namespace AMazurProductEF
7   {
        Odwołania: 0
8       class Program
9       {
            Odwołania: 0
10          static void Main(string[] args)
11          {
12
13              ProdContext context = new ProdContext();
14              Console.WriteLine("Product name: Orange");
15              Console.WriteLine("Invoices numbers: ");
16              var invoices = context.InvoiceProducts.Include(ip => ip.Invoice).Where(ip => ip.ProductID == 3).Select(ip => ip.Invoice.InvoiceNumber).ToList();
17
18              foreach (var i in invoices)
19              {
20                  Console.WriteLine("- " + i);
21              }
22
```

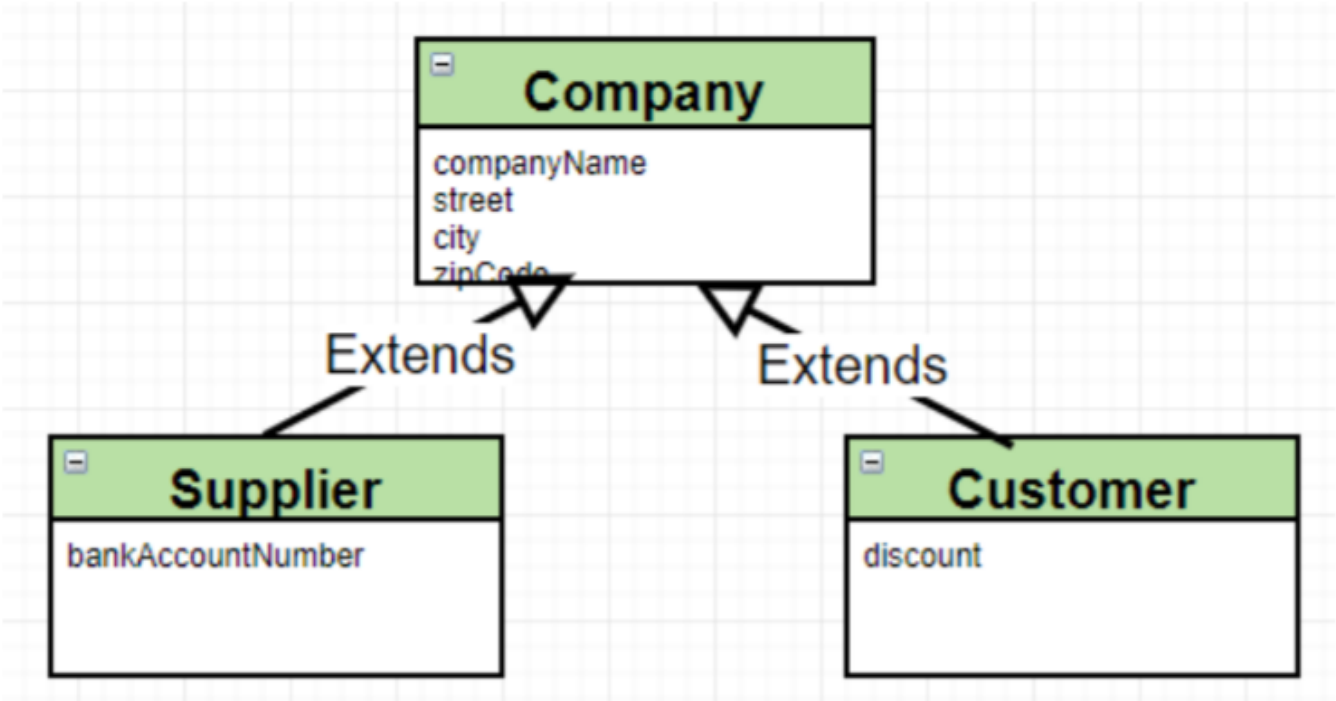
```
Konsola debugowania programu Microsoft Visual Studio
Product name: Orange
Invoices numbers:
- 1
- 2
```

# Zadanie VII

**Dziedziczenie**

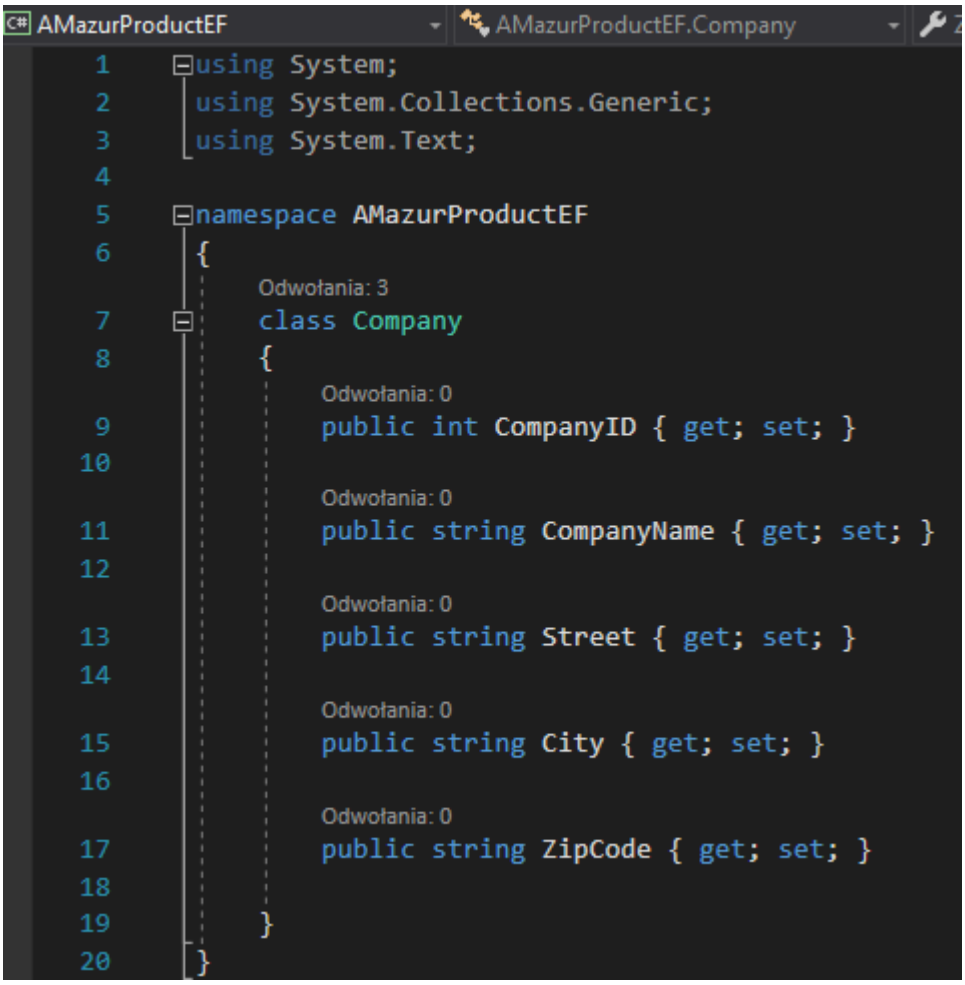
**a)**

Wprowadzono do modelu następujące hierarchie.



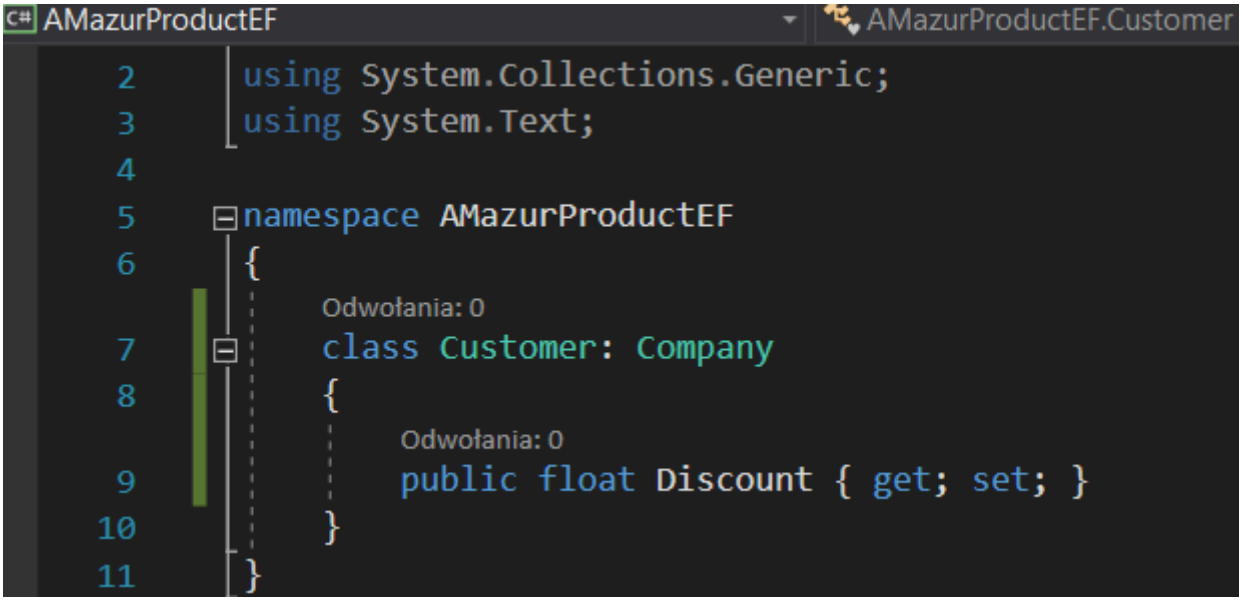

Klasa *Company* zawiera pola:

- int CompanyID
- string CompanyName
- string Street
- string City
- string ZipCode



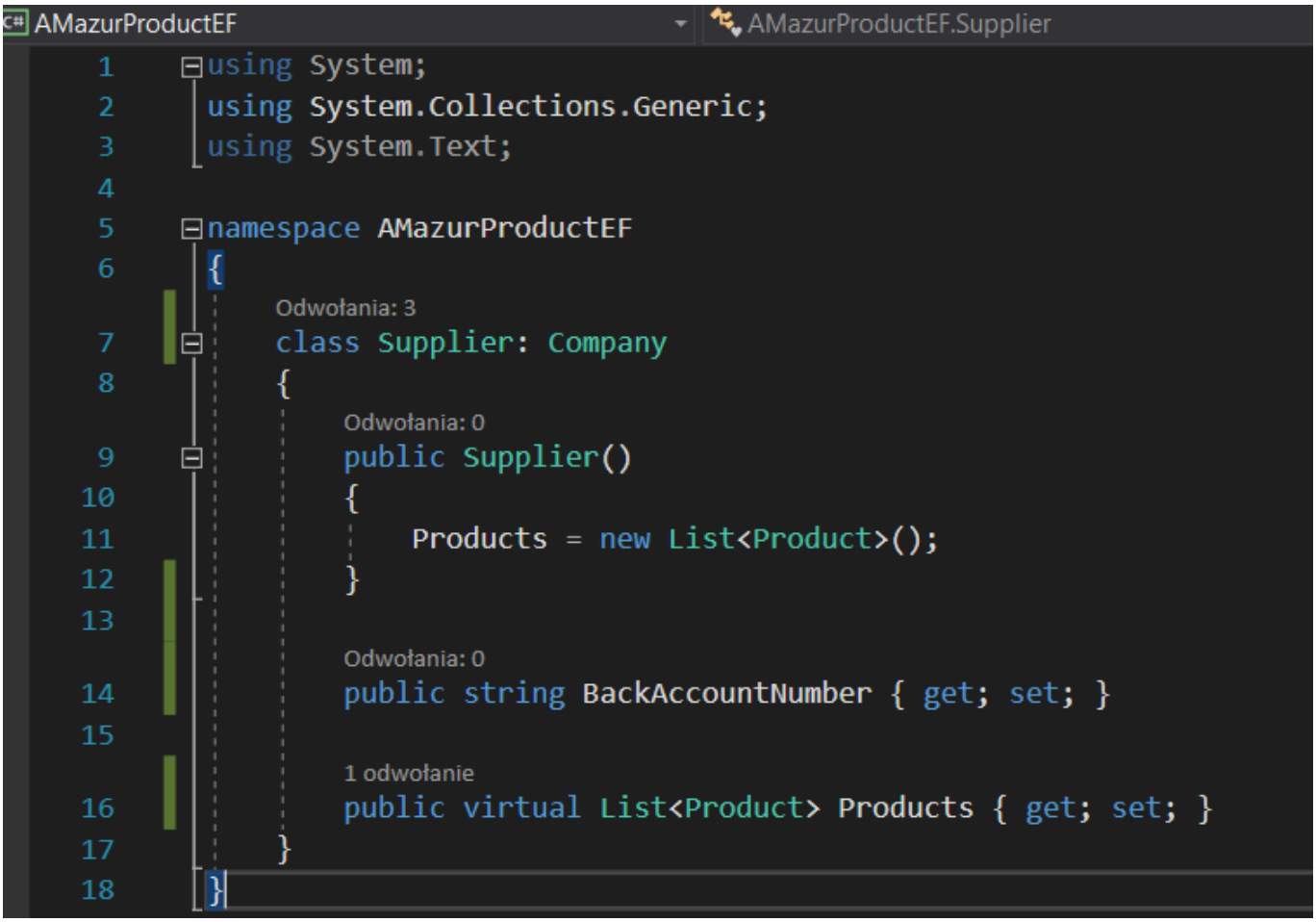

Klasa *Customer* zawiera pole:

- float Discount

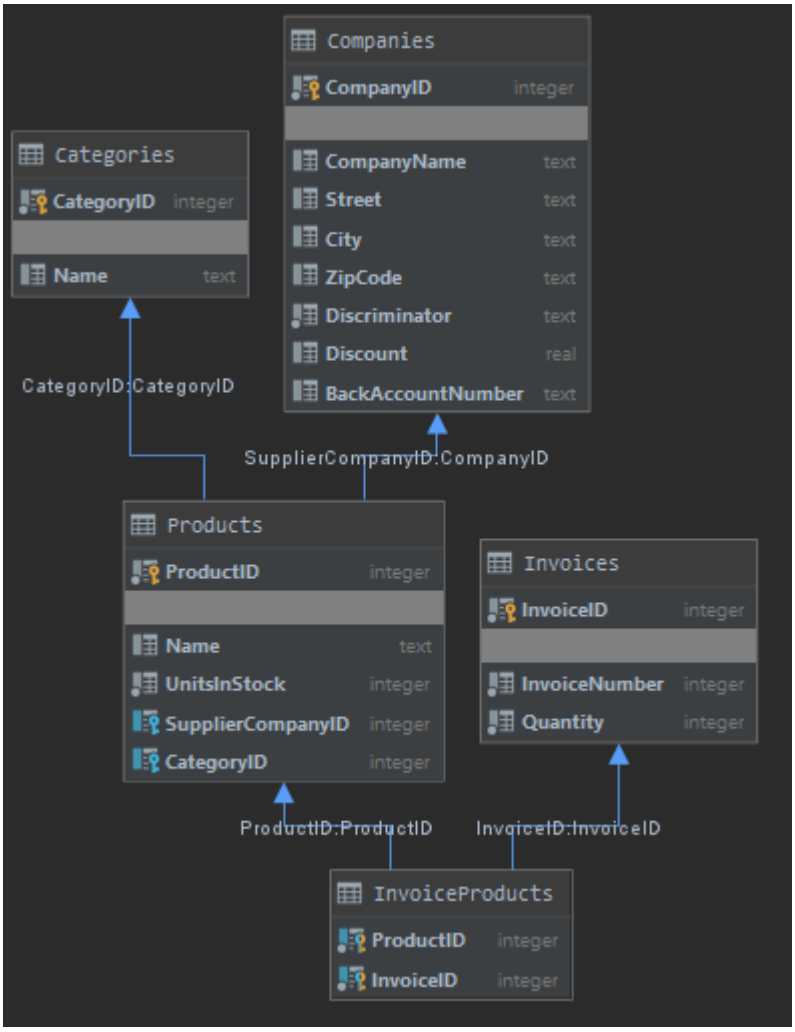


Klasa *Supplier* zawiera pola:

- string BankAccountNumber
- List <Product> Products



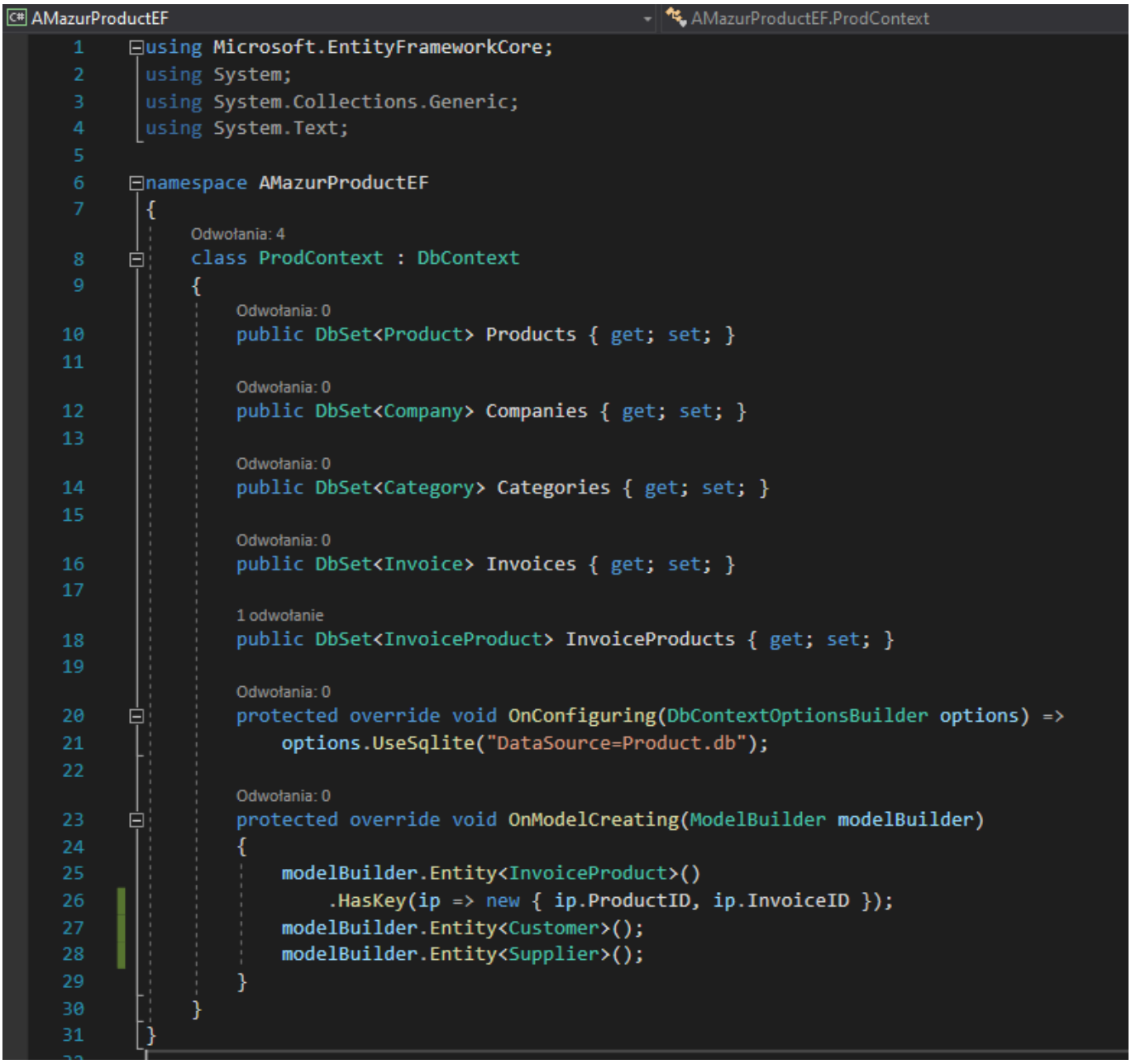
Baza danych wygląda następująco:



**b)**

**TablePerHierarchy**

W celu dodania i pobrania z bazy firm, stosując strategię mapowania dziedziczenia *TablePerHierarchy*, zmieniono metodę *OnModelCreating* w klasie *ProdContext*. Zastąpiono również wcześniejszy DbSet dostawców, *DbSet'em* firm.



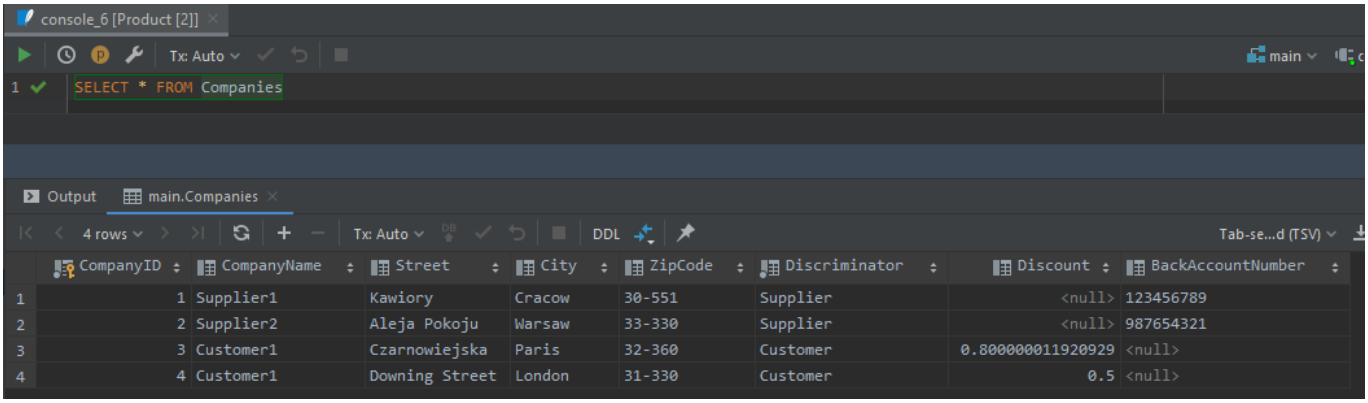
```
-- auto-generated definition
create table Companies
(
    CompanyID         INTEGER not null
        constraint PK_Companies
            primary key autoincrement,
    CompanyName       TEXT,
    Street            TEXT,
    City              TEXT,
    ZipCode           TEXT,
    Discriminator     TEXT    not null,
    Discount          REAL,
    BackAccountNumber TEXT
);
```

Do bazy dodano kilku dostawców i klientów.

```
using System;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Design;

namespace AMazurProductEF
{
    class Program
    {
        static void Main(string[] args)
        {

            ProdContext context = new ProdContext();
            Supplier supplier1 = new Supplier
            {
                CompanyName = "Supplier1",
                City = "Cracow",
                Street = "Kawiory",
                ZipCode = "30-551",
                BackAccountNumber = "123456789",
            };
            context.Add(supplier1);
            Supplier supplier2 = new Supplier
            {
                CompanyName = "Supplier2",
                City = "Warsaw",
                Street = "Aleja Pokoju",
                ZipCode = "33-330",
                BackAccountNumber = "987654321",
            };
            context.Add(supplier2);
            Customer customer1 = new Customer
            {
                CompanyName = "Customer1",
                City = "Paris",
```

```
                Street = "Czarnowiejska",
                ZipCode = "32-360",
                Discount = 0.8f,
            };
            context.Add(customer1);
            Customer customer2 = new Customer
            {
                CompanyName = "Customer1",
                City = "London",
                Street = "Downing Street",
                ZipCode = "31-330",
                Discount = 0.5f,
            };
            context.Add(customer2);
            context.SaveChanges();
        }
    }
}
```

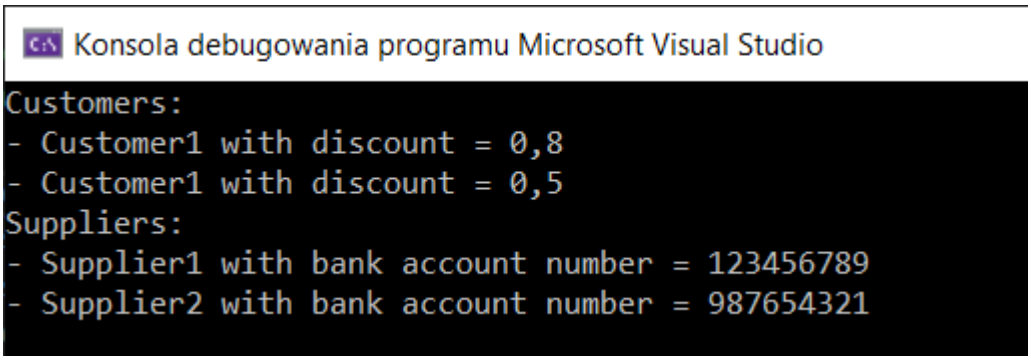
Jak widać dane dodały się prawidłowo.




| | CompanyID | CompanyName | Street | City | ZipCode | Discriminator | Discount | BackAccountNumber |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Supplier1 | Kawiory | Cracow | 30-551 | Supplier | <null> | 123456789 |
| 2 | 2 | Supplier2 | Aleja Pokoju | Warsaw | 33-330 | Supplier | <null> | 987654321 |
| 3 | 3 | Customer1 | Czarnowiejska | Paris | 32-360 | Customer | 0.800000011920929 | <null> |
| 4 | 4 | Customer1 | Downing Street | London | 31-330 | Customer | 0.5 | <null> |

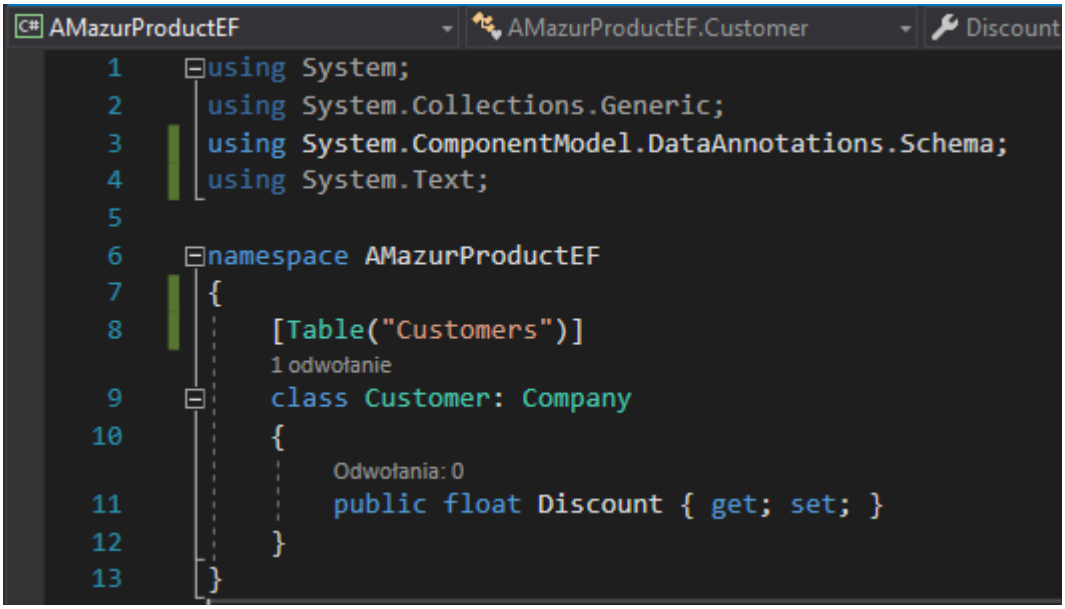
Poniżej pobrano z bazy dane dotyczące dostawców i klientów.

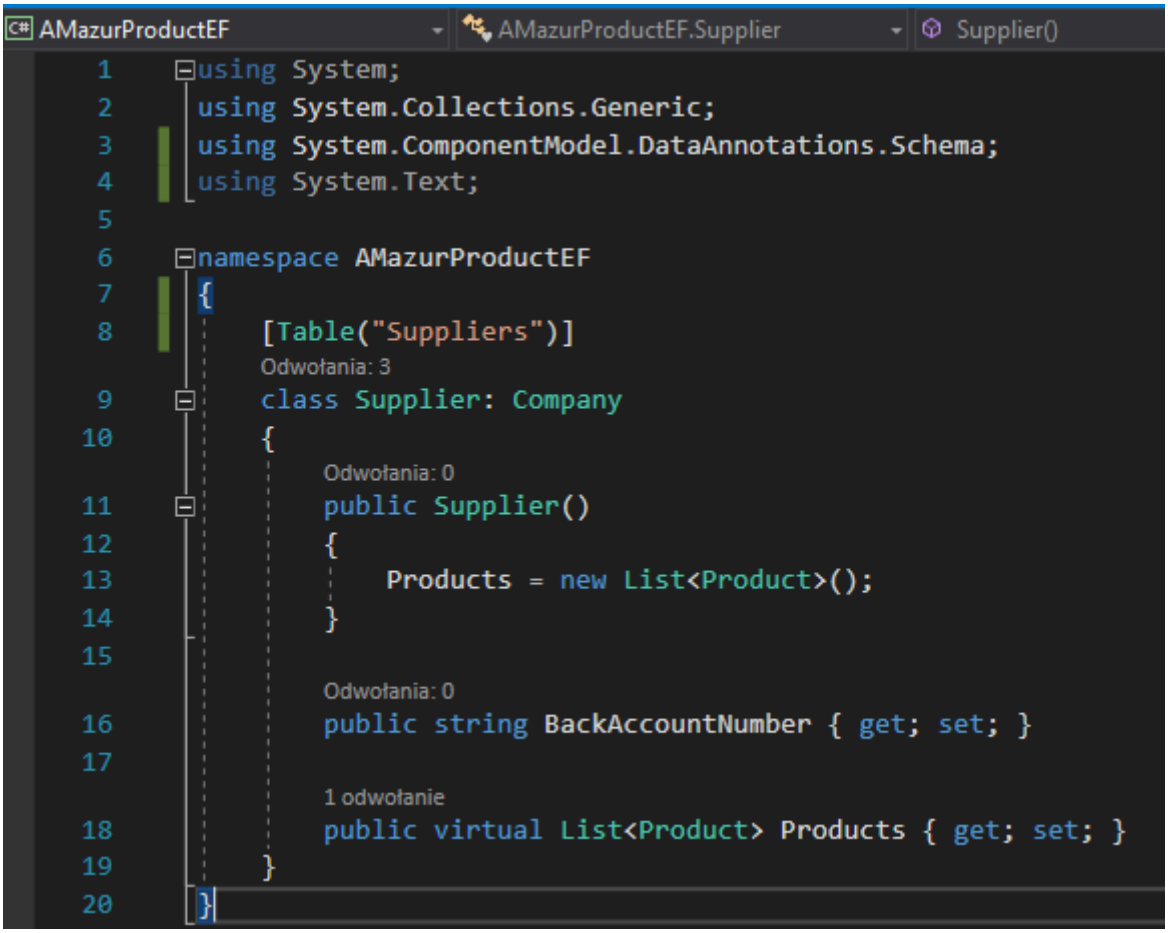
```
ProdContext context = new ProdContext();
var customers = context.Companies.OfType<Customer>().ToList();
var suppliers = context.Companies.OfType<Supplier>().ToList();
Console.WriteLine("Customers: ");
foreach (var c in customers)
{
    Console.WriteLine("- " + c.CompanyName + " with discount = " + c.Discount);
}
Console.WriteLine("Suppliers: ");
foreach (var s in suppliers)
{
    Console.WriteLine("- " + s.CompanyName + " with bank account number = " + s.BackAccountNumber); ;
}
```


Konsola debugowania programu Microsoft Visual Studio
Customers:
- Customer1 with discount = 0,8
- Customer1 with discount = 0,5
Suppliers:
- Supplier1 with bank account number = 123456789
- Supplier2 with bank account number = 987654321

**TablePerType**

Niestety strategia mapowania dziedziczenia *TablePerType* nie jest dostępna w wersjach od *3.0 Entity Framework*, więc nie da się wykonać tego podpunktu. Poniżej jednak zostały przedstawione próby wykonania zadania.

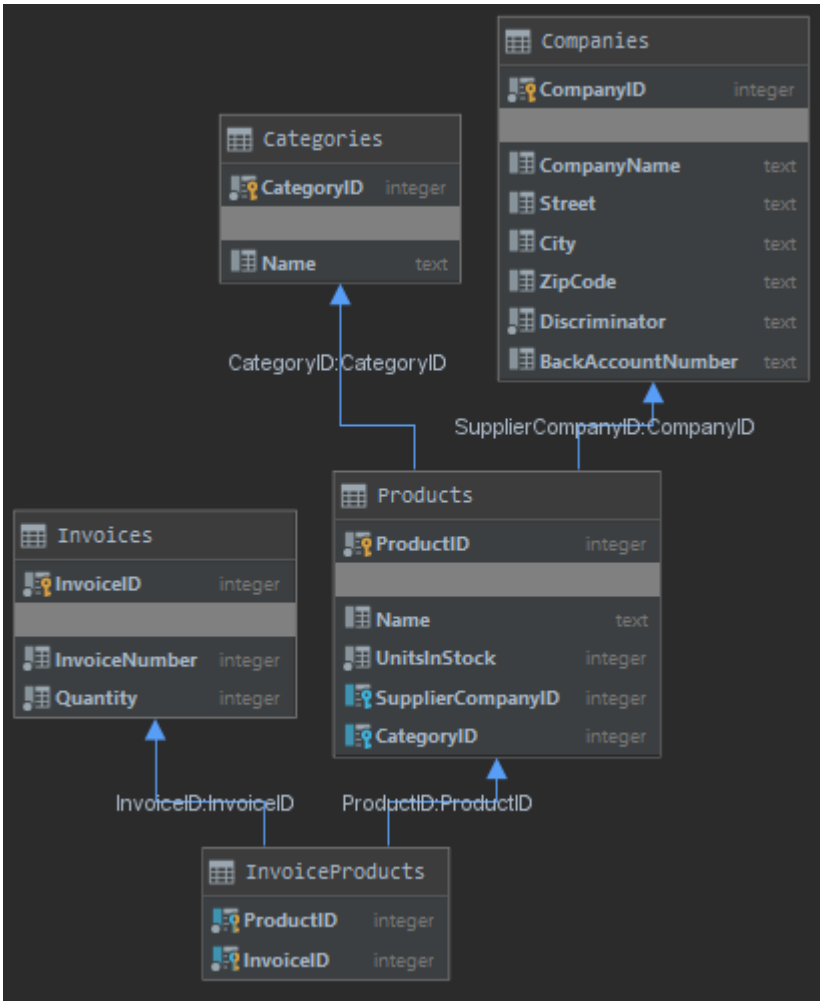


AMazurProductEF | AMazurProductEF.ProdContext | Products

```
1   using Microsoft.EntityFrameworkCore;
2   using System;
3   using System.Collections.Generic;
4   using System.Text;
5
6   namespace AMazurProductEF
7   {
        Odwołania: 4
8       class ProdContext : DbContext
9       {
            Odwołania: 0
10          public DbSet<Product> Products { get; set; }
11
            1 odwołanie
12          public DbSet<Company> Companies { get; set; }
13
            Odwołania: 0
14          public DbSet<Category> Categories { get; set; }
15
            Odwołania: 0
16          public DbSet<Invoice> Invoices { get; set; }
17
            Odwołania: 0
18          public DbSet<InvoiceProduct> InvoiceProducts { get; set; }
19
            Odwołania: 0
20          protected override void OnConfiguring(DbContextOptionsBuilder options) =>
21              options.UseSqlite("DataSource=Product.db");
22
            Odwołania: 0
23          protected override void OnModelCreating(ModelBuilder modelBuilder)
24          {
25              modelBuilder.Entity<InvoiceProduct>()
26                  .HasKey(ip => new { ip.ProductID, ip.InvoiceID });
27          }
28      }
29  }
30
```

Schemat bazy wygląda następująco:




```
-- auto-generated definition
create table Companies
(
    CompanyID         INTEGER not null
        constraint PK_Companies
            primary key autoincrement,
    CompanyName       TEXT,
    Street            TEXT,
    City              TEXT,
    ZipCode           TEXT,
    Discriminator     TEXT    not null,
    BackAccountNumber TEXT
);
```

Można zauważyć, że w tabeli *Companies* brakuje pola *Discount* należącego do klasy *Customer*.

Do bazy dodano poniższe dane:

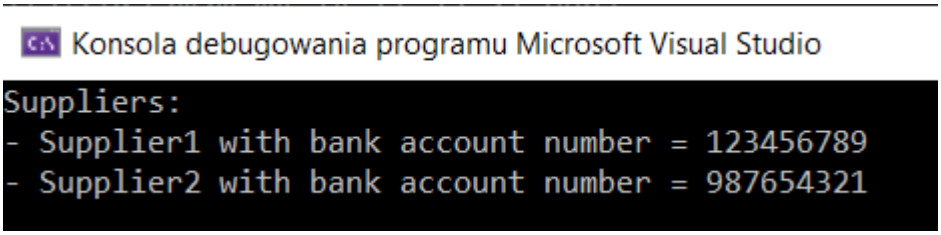
```
Supplier supplier1 = new Supplier
{
    CompanyName = "Supplier1",
    City = "Cracow",
    Street = "Kawiory",
    ZipCode = "30-551",
    BackAccountNumber = "123456789",
};
context.Add(supplier1);
Supplier supplier2 = new Supplier
{
    CompanyName = "Supplier2",
    City = "Warsaw",
    Street = "Aleja Pokoju",
    ZipCode = "33-330",
    BackAccountNumber = "987654321",
};
context.Add(supplier2);
Customer customer1 = new Customer
{
    CompanyName = "Customer1",
    City = "Paris",
    Street = "Czarnowiejska",
    ZipCode = "32-360",
    Discount = 0.8f,
};
context.Add(customer1);
Customer customer2 = new Customer
{
    CompanyName = "Customer1",
    City = "London",
    Street = "Downing Street",
    ZipCode = "31-330",
    Discount = 0.5f,
};
context.Add(customer2);
context.SaveChanges();
```

| | CompanyID | CompanyName | Street | City | ZipCode | Discriminator | BackAccountNumber |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Supplier1 | Kawiory | Cracow | 30-551 | Supplier | 123456789 |
| 2 | 2 | Supplier2 | Aleja Pokoju | Warsaw | 33-330 | Supplier | 987654321 |
| 3 | 3 | Customer1 | Czarnowiejska | Paris | 32-360 | Company | <null> |
| 4 | 4 | Customer1 | Downing Street | London | 31-330 | Company | <null> |

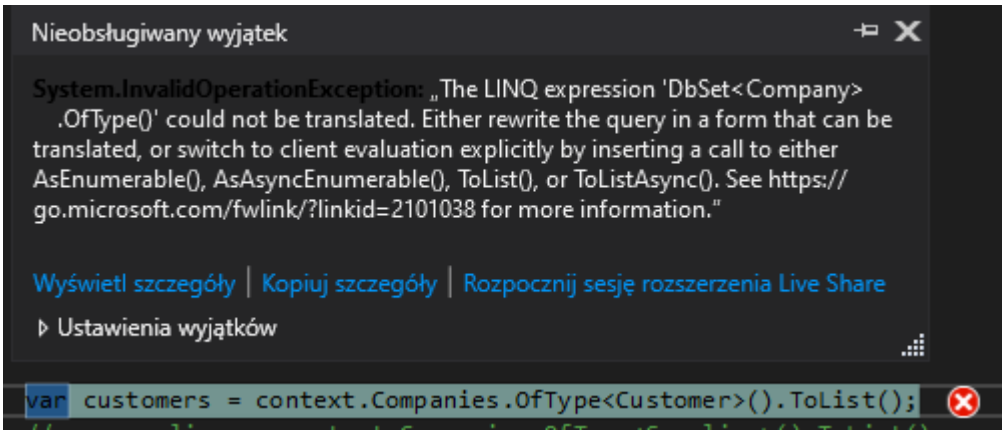
Spróbowano wypisać dodane wcześniej dane.

```
var customers = context.Companies.OfType<Customer>().ToList();
var suppliers = context.Companies.OfType<Supplier>().ToList();
Console.WriteLine("Customers: ");
foreach (var c in customers)
{
    Console.WriteLine("- " + c.CompanyName + " with discount = " + c.Discount);
}
Console.WriteLine("Suppliers: ");
foreach (var s in suppliers)
{
    Console.WriteLine("- " + s.CompanyName + " with bank account number = " + s.BackAccountNumber); ;
}
```

Dostawcy zostali wypisani poprawnie:




Z kolei przy próbie wypisania klientów, został wyświetlony poniższy wyjątek:




**TablePerClass**

Strategia *TablePerClass* również nie jest możliwa do wykonania, ponieważ metoda *ToTable()*, która jest niezbędna do wykonania tego zadania rzuca błąd od wersji *3.0 Entity Framework*.

> ⓘ Note
>
> The table-per-type (TPT) and table-per-concrete-type (TPC), which are supported by EF6, are not yet supported by EF Core. TPT is a major feature planned for EF Core 5.0.