JEGYZŐKÖNYV

Modern adatbázis rendszerek MSc

2022. tavasz féléves feladat

LINQ programozása

Készítette: **Veres Marcell**

Neptunkód: **JEDU1N**

A feladat leírása:

Hozzunk létre az órán megjelentek névsorából egy string tömböt. Kérdezzük le a tömb első 3 elemét, majd írjuk ki az eredményt. Rendezzük a neveket csökkenő sorrendbe vezetéknév szerint, majd írjuk ki az eredményt. Adjuk vissza azon nevek darabszámát, amelyek tartalmaznak l vagy L karaktert. Írjuk ki azoknak a keresztnevét, akiknek a vezetékneve hosszabb mint 5.

Hozzunk létre egy int tömböt, amely tartalmazza a számokat 0-tól 10-ig. Írjuk ki a páros számok négyzetét. Írjuk ki a számok átlagát. Írjuk ki az összes olyan számot és az átlagtól számított eltérését, amelyik nagyobb, mint az átlag. Írjuk ki, hogy az egyes eltérésekhez mely értékek tartoznak.

Hozzunk létre egy Guest nevű típust a következő adatokkal:

- RoomId (3 számjegy: Emelet, Szárny, szobaszám)
- Name
- Age
- Address (ország, város).

Hozzunk létre egy legalább 4 elemű tömböt (Guest elemekkel). Írjuk ki azon országokat és az onnan érkezők számát, ahonnan legalább ketten jöttek. Írjuk ki azon vendégeket, akik az első vendéggel egy emeleten és egy szárnyban lettek elszállásolva.

Kérjük le az alábbi adatokat a Nutshell DB-ből:

- Összes vásárlót (Customers tábla)
- Kérjük le a vásárlók számát
- Kérjük le az 1-es id-val rendelkezőt

Kérjük le az alábbi adatokat a Demo database DB-ből:

- Összes vásárlót (Customers tábla)
- Kérjük le a vásárlók számát
- Kérjük le az 1-es id-val rendelkezőt

Adjunk hozzá a Customer táblához egy új rekordot. Kérdezzük le a tábla elemszámát a hozzáadás előtt és után. Töröljük a Customer táblából az új rekordot. Kérdezzük le a tábla elemszámát a törlés előtt és után. Módosítsuk a Customer tábla egy rekordját, majd írjuk ki az id-ja alapján a módosított értéket. Írjuk ki a vásárlókat és a hozzájuk tartozó vásárlások leírását és árát Join segítségével. Készítsük el az előző lekérdezést C# Expression segítségével is.

A feladat elkészítésének lépései:

```
//a
var names = new [] {"Vadon Enikő", "Bolyki Balázs", "Toronya Bertalan", "Polonkai Dávid",
"Fazekas Levente", "Német Viktor"};
//b
names.Select(x => x).Dump("Task B");
//c
names.OrderBy(x => x.Split()[0]).Dump("Task C");
//d
names.Where(x => x.Contains("L") || x.Contains("l")).Count().Dump("Task D");
//e
names.Where(x => x.Split(" ")[0].Length > 5).Select(x => x.Split(" ")[1]).Dump("Task E");

//a
List<int> numbers = new(){0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
//b
numbers.Where(x => x % 2 == 0).Select(x => x * x).Dump("Task B");
//c
numbers.Average().Dump("Task C");
//d
numbers.Where(x => x > numbers.Average()).Select(x => new {x, Diff = Math.Abs(x -
numbers.Average())}).Dump("Task D");
//e
numbers.GroupBy(x => Math.Abs(x - numbers.Average())).Dump("Task E");

var guests = new []
{
	new Guest(101, "Mariska", 18, new Address("Fokváros", "Kukutyin")),
	new Guest(103, "Gizi", 43, new Address("Újváros", "Kukutyin")),
	new Guest(135, "Ferenc", 56, new Address("Főváros", "Kamcsatka")),
	new Guest(400, "Aladár", 23, new Address("Óváros", "Románia")),
};
//c
guests.GroupBy(x => x.Address.Country).Where(x => x.Count() > 1).Select(x => new{x.Key,
count=x.Count()}).Dump("Task C");
//d
guests.Where(x => x.RoomId.ToString().Substring(0,2) ==
guests.First().RoomId.ToString().Substring(0,2)).Select(x => x.Name).Dump("Task D");
//a
class Guest
{
	public int RoomId {get; set;}
	public string Name {get; set;}
	public int Age {get; set;}
	public Address Address {get; set;}

	public Guest(int roomId, string name, int age, Address address)
	{
		RoomId = roomId;
		Name = name;
		Age = age;
		Address = address;
	}
}

class Address
{
	public string City {get; set;}
	public string Country {get; set;}

	public Address(string city, string country)
	{
		City = city;
		Country = country;
	}
}
```

```
var dbContext = this;

//a & b
Customers.Dump("Customers");
Customers.Count().Dump("Count");
Customers.Single(c => c.ID == 2).Dump("ID 2");
//Customers.Single(c => c.Customerd == 2).Dump();

//c & d
dbContext.Customers.Count().Dump("Count");

Customer customer = new Customer()
{
	ID = 10,
	Name = "Thomas"
};

//dbContext.Customers.Add(customer);
//dbContext.SaveChanges();

dbContext.Customers.Count().Dump("Count");

//dbContext.Customers.Remove(customer);
dbContext.SaveChanges();

dbContext.Customers.Count().Dump("Count");

//e
customer.Name = "Thomas";
dbContext.SaveChanges();

Customers.Single(x => x.ID == customer.ID).Dump("Task E");

//f
Customers.Join(Purchases, c => c.ID, p => p.CustomerID, (c, p) => new {c.Name,
p.Description, p.Price}).Dump("Task F");

//g
/*from c in Customers
join p in Purchases on c.ID equals p.CustomerID
select new
{
	c.Name,
	p.Description,
	p.Price
}*/
```

A futtatás eredménye: