ENY

osztály

**1. CSV fájl előkészítése**

**uid,paintingid,startdate,enddate,daily\_price,artist,title**

**2. Berles osztály létrehozása**

class Berles

{

public int Uid { get; set; }

public int PaintingId { get; set; }

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

public int DailyPrice { get; set; }

public string Artist { get; set; }

public string Title { get; set; }

public int Duration => (EndDate - StartDate).Days;

public int TotalPrice => Duration \* DailyPrice;

}

3. **Adatok beolvasása CSV fájlból**List<Berles> berlesek = File.ReadAllLines("festmeny\_berlesek\_2024.csv")

.Skip(1)

.Select(sor =>

{

var t = sor.Split(',');

return new Berles

{

Uid = int.Parse(t[0]),

PaintingId = int.Parse(t[1]),

StartDate = DateTime.Parse(t[2]),

EndDate = DateTime.Parse(t[3]),

DailyPrice = int.Parse(t[4]),

Artist = t[5],

Title = t[6]

};

})

.ToList();

4. **Lekérdezések és jelentések**

**1. Adott hónap bevétele**

Console.Write("Adjon meg egy hónapot (1-12): ");

int honap = int.Parse(Console.ReadLine());

int haviBevetel = berlesek

.Where(b => b.StartDate.Month == honap || b.EndDate.Month == honap)

.Sum(b =>

{

var start = b.StartDate.Month < honap ? new DateTime(b.StartDate.Year, honap, 1) : b.StartDate;

var end = b.EndDate.Month > honap ? new DateTime(b.EndDate.Year, honap, DateTime.DaysInMonth(b.EndDate.Year, honap)) : b.EndDate;

return (end - start).Days \* b.DailyPrice;

});

Console.WriteLine($"{honap}. havi bevétel: {haviBevetel:N0} Ft");

**2. Teljes éves bevétel**

int evesBevetel = berlesek.Sum(b => b.TotalPrice);

Console.WriteLine($"Teljes éves bevétel: {evesBevetel:N0} Ft");

**3. Legdrágább bérlés**

var legdragabb = berlesek.OrderByDescending(b => b.TotalPrice).First();

Console.WriteLine($"Legdrágább bérlés: {legdragabb.Title} ({legdragabb.TotalPrice:N0} Ft)");

**4. Különböző festmények száma**

int kulonbozoFestmenyek = berlesek.Select(b => b.PaintingId).Distinct().Count();

Console.WriteLine($"Különböző festmények száma: {kulonbozoFestmenyek}");

**5. Legtöbbször bérelt festmény**

var legtobbszorBerelt = berlesek

.GroupBy(b => b.Title)

.OrderByDescending(g => g.Count())

.First();

Console.WriteLine($"Legtöbbször bérelt festmény: {legtobbszorBerelt.Key} ({legtobbszorBerelt.Count()} alkalom)");

**6. Bérlések száma művészenként**

var muveszenkent = berlesek

.GroupBy(b => b.Artist)

.OrderBy(g => g.Key);

Console.WriteLine("Bérlések száma festőnként:");

foreach (var group in muveszenkent)

{

Console.WriteLine($"{group.Key}: {group.Count()} db");

}

**7. Átlagos bérlési időtartam (napban)**

double atlagosIdotartam = berlesek.Average(b => b.Duration);

Console.WriteLine($"Átlagos bérlési időtartam: {atlagosIdotartam:F1} nap");

TELJES

using System;

using System.Collections.Generic;

using System.Globalization;

using System.IO;

using System.Linq;

class Berles

{

public int Uid { get; set; }

public int PaintingId { get; set; }

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

public int DailyPrice { get; set; }

public string Artist { get; set; }

public string Title { get; set; }

public int Days => (EndDate - StartDate).Days + 1;

public int TotalPrice => Days \* DailyPrice;

}

class Program

{

static void Main()

{

var berlesek = new List<Berles>();

var lines = File.ReadAllLines("festmeny\_berlesek\_2024.csv", System.Text.Encoding.UTF8).Skip(1);

foreach (var line in lines)

{

var parts = line.Split(',');

berlesek.Add(new Berles

{

Uid = int.Parse(parts[0]),

PaintingId = int.Parse(parts[1]),

StartDate = DateTime.ParseExact(parts[2], "yyyy-MM-dd", CultureInfo.InvariantCulture),

EndDate = DateTime.ParseExact(parts[3], "yyyy-MM-dd", CultureInfo.InvariantCulture),

DailyPrice = int.Parse(parts[4]),

Artist = parts[5],

Title = parts[6]

});

}

Console.Write("Adjon meg egy hónapot (1-12): ");

int month = int.Parse(Console.ReadLine());

int haviBevetel = berlesek

.Where(b => b.StartDate.Month == month || b.EndDate.Month == month ||

(b.StartDate.Month < month && b.EndDate.Month > month))

.Sum(b => b.TotalPrice);

Console.WriteLine($"A(z) {month}. havi bevétel: {haviBevetel:N0} Ft");

int evesBevetel = berlesek.Sum(b => b.TotalPrice);

Console.WriteLine($"Teljes éves bevétel: {evesBevetel:N0} Ft");

var legdragabb = berlesek.OrderByDescending(b => b.TotalPrice).First();

Console.WriteLine($"Legdrágább bérlés: {legdragabb.Title} ({legdragabb.TotalPrice:N0} Ft)");

int kulonbozoFestmenyek = berlesek.Select(b => b.PaintingId).Distinct().Count();

Console.WriteLine($"Különböző festmények száma: {kulonbozoFestmenyek}");

var legtobbszorBerelt = berlesek

.GroupBy(b => b.Title)

.OrderByDescending(g => g.Count())

.First();

Console.WriteLine($"Legtöbbször bérelt festmény: {legtobbszorBerelt.Key} ({legtobbszorBerelt.Count()} alkalom)");

Console.WriteLine("Bérlések száma festőnként:");

var muveszCsoport = berlesek

.GroupBy(b => b.Artist)

.OrderByDescending(g => g.Count());

foreach (var group in muveszCsoport)

{

Console.WriteLine($"{group.Key}: {group.Count()} db");

}

double atlagNap = berlesek.Average(b => b.Days);

Console.WriteLine($"Átlagos bérlési időtartam: {atlagNap:F1} nap");

}

}