# Exercises: Introduction to DB Apps

This document defines the **exercise assignments** for the ["Databases Advanced – EF Core" course @ Software University](https://softuni.bg/trainings/1741/databases-advanced-entity-framework-october-2017).

## Initial Setup

Write a program that connects to your **localhost** server. Create **new database** called **MinionsDB** where we will keep information about our minions and villains.

For each **minion** we should keep information about its **name**, **age** and **town**. Each **town** has information about **the** **country** where it’s located. **Villains** have **name** and **evilness** **factor** (**super good**, **good**, **bad**, **evil**, **super** **evil**). Each **minion** can **serve** **several** **villains** and **each** **villain** can **have** **several** **minions** **serving** **him**. Fill all tables with at least 5 records each.

In the end you shoud have the following tables:

* Countries
* Towns
* Minions
* EvilnessFactors
* Villains
* MinionsVillains



**Solution :**

using System;

using System.Data.SqlClient;

namespace Problem01

{

public class StartUp

{

private static string SERVER\_PATH = "Server=DESKTOP-1SHT5A0\\SQLEXPRESS; Database={0}; Integrated Security = true";

static void Main(string[] args)

{

var connection = new SqlConnection(string.Format(SERVER\_PATH, "master"));

connection.Open();

using (connection)

{

string command = "CREATE DATABASE MinionsDB";

try

{

SqlCommand createDataBase = new SqlCommand(command, connection);

createDataBase.ExecuteNonQuery();

Console.WriteLine("Database created successfully!");

}

catch (Exception e)

{

Console.WriteLine("Database was not created!");

Console.WriteLine(e.Message);

}

}

connection = new SqlConnection(string.Format(SERVER\_PATH, "MinionsDB"));

connection.Open();

using (connection)

{

string command = "CREATE TABLE Countries (Id INT PRIMARY KEY IDENTITY,Name VARCHAR(50)) " +

"CREATE TABLE Towns(Id INT PRIMARY KEY IDENTITY, Name VARCHAR(50), CountryCode INT FOREIGN KEY REFERENCES Countries(Id))" +

"CREATE TABLE Minions(Id INT PRIMARY KEY IDENTITY, Name VARCHAR(30), Age INT, TownId INT FOREIGN KEY REFERENCES Towns(Id))" +

"CREATE TABLE EvilnessFactors(Id INT PRIMARY KEY IDENTITY, Name VARCHAR(50))" +

"CREATE TABLE Villains(Id INT PRIMARY KEY IDENTITY, Name VARCHAR(50), EvilnessFactorId INT FOREIGN KEY REFERENCES EvilnessFactors(Id))" +

"CREATE TABLE MinionsVillains(MinionId INT FOREIGN KEY REFERENCES Minions(Id), VillainId INT FOREIGN KEY REFERENCES Villains(Id), CONSTRAINT PK\_MinionsVillains PRIMARY KEY(MinionId, VillainId))";

try

{

SqlCommand createTables = new SqlCommand(command, connection);

createTables.ExecuteNonQuery();

Console.WriteLine("Tables created successfully!");

}

catch (Exception e)

{

Console.WriteLine("A able was not created!");

Console.WriteLine(e.Message);

}

}

connection = new SqlConnection(string.Format(SERVER\_PATH, "MinionsDB"));

connection.Open();

using (connection)

{

string command = "INSERT INTO Countries ([Name]) VALUES ('Bulgaria'),('England'),('Cyprus'),('Germany'),('Norway') " +

"INSERT INTO Towns([Name], CountryCode) VALUES('Plovdiv', 1),('Varna', 1),('Burgas', 1),('Sofia', 1),('London', 2),('Southampton', 2),('Bath', 2),('Liverpool', 2),('Berlin', 3),('Frankfurt', 3),('Oslo', 4) " +

"INSERT INTO Minions(Name, Age, TownId) VALUES('Bob', 42, 3),('Kevin', 1, 1),('Bob ', 32, 6),('Simon', 45, 3),('Cathleen', 11, 2),('Carry ', 50, 10),('Becky', 125, 5),('Mars', 21, 1),('Misho', 5, 10),('Zoe', 125, 5),('Json', 21, 1) " +

"INSERT INTO EvilnessFactors(Name) VALUES('Super good'),('Good'),('Bad'), ('Evil'),('Super evil') " +

"INSERT INTO Villains(Name, EvilnessFactorId) VALUES('Gru', 2),('Victor', 1),('Jilly', 3),('Miro', 4),('Rosen', 5),('Dimityr', 1),('Dobromir', 2) " +

"INSERT INTO MinionsVillains(MinionId, VillainId) VALUES(4, 2),(1, 1),(5, 7),(3, 5),(2, 6),(11, 5),(8, 4),(9, 7),(7, 1),(1, 3),(7, 3),(5, 3),(4, 3),(1, 2),(2, 1),(2, 7)";

try

{

SqlCommand feedtables = new SqlCommand(command, connection);

feedtables.ExecuteNonQuery();

Console.WriteLine("Tables filled in successfully!");

}

catch (Exception e)

{

Console.WriteLine("A able was not filled in correctly!");

Console.WriteLine(e.Message);

}

}

}

}

}

## Villain Names

Write a program that prints on the console **all villains’ names** and their **number of minions** of those who have more than 3 minions **ordered descending** by number of minions.

**Solution:**

using System;

using System.Data.SqlClient;

namespace Problem02\_Villain\_Names

{

public class StartUp

{

private static string SERVER\_PATH = "Server=DESKTOP-1SHT5A0\\SQLEXPRESS; Database=MinionsDB; Integrated Security = true";

static void Main(string[] args)

{

var connection = new SqlConnection(SERVER\_PATH);

connection.Open();

using (connection)

{

string command = " SELECT v.Name, COUNT(mv.VillainId) AS MinionsCount " +

" FROM Villains AS v " +

" JOIN MinionsVillains AS mv ON v.Id = mv.VillainId " +

" GROUP BY v.Id, v.Name " +

" HAVING COUNT(mv.VillainId) > 3 " +

" ORDER BY COUNT(mv.VillainId)";

try

{

SqlCommand getData = new SqlCommand(command, connection);

SqlDataReader reader = getData.ExecuteReader();

using (reader)

{

while (reader.Read())

{

string villainName = (string)reader["Name"];

int minionsCount = (int)reader["MinionsCount"];

Console.WriteLine($"{villainName} - {minionsCount}");

}

}

}

catch (Exception e )

{

Console.WriteLine(e.Message);

}

}

}

}

}

### Example

|  |
| --- |
| **Output** |
| Gru - 6  Victor - 4  Jilly – 4 |

## Minion Names

Write a program that prints on the console **all minion names** and age for a given **villain id**, ordered by **name** **alphabetically.**

If there is no villain with the given ID, print "No villain with ID <**VillainId**> exists in the database.".  
If the selected villain has no minions, print "(no minions)" on the second row.

**Solution:**

using System;

using System.Data;

using System.Data.SqlClient;

namespace Problem03.MinionNames

{

public class StartUp

{

private static string SERVER\_PATH = "Server=DESKTOP-1SHT5A0\\SQLEXPRESS; Database=MinionsDB; Integrated Security = true";

static void Main(string[] args)

{

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

string minionName = string.Empty;

var connection = new SqlConnection(SERVER\_PATH);

connection.Open();

using (connection)

{

string command = $"SELECT Name FROM Villains WHERE Id = {villainId}";

try

{

SqlCommand getData = new SqlCommand(command, connection);

string villainName = (string)getData.ExecuteScalar();

if (villainName == null)

{

Console.WriteLine($"No villain with ID {villainId} exists in the database.");

return;

}

else

{

Console.WriteLine($"Villain: {villainName}");

string getMinionsNames = $"SELECT ROW\_NUMBER() OVER (ORDER BY m.Name) as RowNum, " +

$" m.Name, " +

$" m.Age " +

$" FROM MinionsVillains AS mv " +

$" JOIN Minions As m ON mv.MinionId = m.Id " +

$" WHERE mv.VillainId = {villainId} " +

$" ORDER BY m.Name";

try

{

SqlCommand getMinions = new SqlCommand(getMinionsNames, connection);

SqlDataReader reader = getMinions.ExecuteReader();

using (reader)

{

int counter = 0;

while (reader.Read())

{

minionName = (string)reader[1];

int minionAge = (int)reader[2];

Console.WriteLine($"{++counter}. {minionName} {minionAge}");

}

if (counter == 0)

{

Console.WriteLine("(no minions)");

return;

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

}

}

### Example

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 1 | Villain: Gru  1. Bob 13  2. Kevin 14  3. Steward 19 |  | 3 | Villain: Victor  1. Bob 13  2. Simon 22 |  | 2 | Villain: Victor Jr.  (no minions) |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10 | No villain with ID 10 exists in the database. |

## Add Minion

Write a program that **reads** **information** about a minion and its villain and **adds it to the database**.

* In case the town of the minion is not in the database, **insert** it as well.
* In case the villain is not present in the database, add him too with a default **evilness** **factor** of "evil" (4).
* Finally set the new minion to be a servant of the villain. Print appropriate messages after each operation.

### Input

The input comes in two lines:

* On the first line, you will receive the **minion** **information** in the format "Minion: <**Name**> <**Age**> <**TownName**>"
* On the second – the **villain** **information** in the format "Villain: <**Name**>"

### Output

After completing an operation, you must print one of the following messages:

* After adding a new **town** to the database: "Town <**TownName**> was added to the database."
* After adding a new **villain** to the database: "Villain <**VillainName**> was added to the database."
* Finally, after successfully adding the **minion** to the database and making it a **servant** of a villain: "Successfully added <**MinionName**> to be minion of <**VillainName**>."

**\*Bonus task:** Make sure all operations are executed successfully. In case of an error do not change the database.

**Solution:**

using System;

using System.Data.SqlClient;

using System.Linq;

namespace Problem04AddMinion

{

public class StartUp

{

private static string SERVER\_PATH = "Server=DESKTOP-1SHT5A0\\SQLEXPRESS; Database=MinionsDB; Integrated Security = true";

static void Main(string[] args)

{

string[] minionInput = Console.ReadLine()

.Split(": ")

.ToArray();

string[] villainInput = Console.ReadLine()

.Split(": ")

.ToArray();

string minionName = minionInput[1].Split()[0];

int minionAge = int.Parse(minionInput[1].Split()[1]);

string minionTown = minionInput[1].Split()[2];

string villainName = villainInput[1];

string findVillain = $"SELECT Id FROM Villains WHERE Name = '{villainName}'"; // create a method getting the Id

string findMinion = $"SELECT Id FROM Minions WHERE Name = '{minionName}'"; // create a method getting the Id

string findTown = $"SELECT Id FROM Towns WHERE Name = '{minionTown}'";

string addTown = $"INSERT INTO Towns (Name) VALUES ('{minionTown}')"; // create the town

string createVallain = $"INSERT INTO Villains (Name, EvilnessFactorId) VALUES ('{villainName}', 4)"; // if villain is not in the table

var connection = new SqlConnection(SERVER\_PATH);

connection.Open();

using (connection)

{

SqlTransaction sqlTran = connection.BeginTransaction();

SqlCommand getTown = new SqlCommand(findTown, connection); // find the town

getTown.Transaction = sqlTran;

string townId = getTown.ExecuteScalar() as string;

try

{

if (townId == null)

{

SqlCommand createTown = new SqlCommand(addTown, connection);

createTown.Transaction = sqlTran;

createTown.ExecuteNonQuery();

Console.WriteLine($"Town {minionTown} was added to the database.");

sqlTran.Commit();

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

sqlTran.Rollback();

}

SqlCommand getVillain = new SqlCommand(findVillain, connection);

getVillain.Transaction = sqlTran;

object villainId = getVillain.ExecuteScalar();

getVillain.Transaction = sqlTran;

try

{

if (villainId == null)

{

SqlCommand createVillain = new SqlCommand(createVallain, connection);

createVillain.Transaction = sqlTran;

createVillain.ExecuteNonQuery();

Console.WriteLine($"Villain {villainName} was added to the database.");

sqlTran.Commit();

}

int villiansIdToInt = (int)villainId; // error??? as int ????

try

{

SqlCommand getMinionId = new SqlCommand(findMinion, connection);

getMinionId.Transaction = sqlTran;

int minionId = (int)getMinionId.ExecuteScalar();

try

{

string addMinionVallain = $"INSERT INTO MinionsVillains (MinionId, VillainId) VALUES ({minionId}, {villiansIdToInt})";

SqlCommand addMinionVillain = new SqlCommand(addMinionVallain, connection);

addMinionVillain.Transaction = sqlTran;

addMinionVillain.ExecuteNonQuery();

sqlTran.Commit();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

sqlTran.Rollback();

}

sqlTran.Commit();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

sqlTran.Rollback();

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

sqlTran.Rollback();

}

getTown = new SqlCommand(findTown, connection);

getTown.Transaction = sqlTran;

int townIdInt = (int)getTown.ExecuteScalar();

string addMinion = $"INSERT INTO Minions (Name, Age, TownId) VALUES ('{minionName}', {minionAge}, {townIdInt})";

try

{

SqlCommand createMinion = new SqlCommand(addMinion, connection);

createMinion.Transaction = sqlTran;

createMinion.ExecuteNonQuery();

Console.WriteLine($"Successfully added {minionName} to be minion of {villainName}.");

sqlTran.Commit();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

sqlTran.Rollback();

}

}

}

}

}

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| Minion: Bob 14 Berlin  Villain: Gru | Successfully added Robert to be minion of Gru. |
| Minion: Cathleen 20 Liverpool  Villain: Gru | Town Liverpool was added to the database.  Successfully added Cathleen to be minion of Gru. |
| Minion: Mars 23 Sofia  Villain: Poppy | Villain Poppy was added to the database.  Successfully added Mars to be minion of Poppy. |
| Minion: Carry 20 Eindhoven  Villain: Jimmy | Town Eindhoven was added to the database.  Villain Jimmy was added to the database.  Successfully added Carry to be minion of Jimmy. |

## Change Town Names Casing

Write a program that **changes all town names to uppercase** for a given country.

You will receive one line of input with the **name** of the country.

**Print the** **number of towns that were changed** in the format "<**ChangedTownsCount**> town names were affected.". On a second line, **print** the **names that were changed**, separated with a comma and a space.

If **no** **towns** were affected (the country does not exist in the database or has no cities connected to it), **print** "**No town names were affected.**".

**Solution:**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

namespace Problem05ChangeTownNamesCasing

{

public class StartUp

{

private static string SERVER\_PATH = "Server=DESKTOP-1SHT5A0\\SQLEXPRESS; Database=MinionsDB; Integrated Security = true";

static void Main(string[] args)

{

string inputCountryName = Console.ReadLine();

string command = $"UPDATE Towns" +

$" SET Name = UPPER(Name) " +

$" WHERE CountryCode = (" +

$" SELECT c.Id FROM Countries AS c " +

$" WHERE c.Name = '{inputCountryName}')";

string checkCommand = $" SELECT COUNT(t.Name) FROM Towns as t " +

$" JOIN Countries AS c ON c.Id = t.CountryCode " +

$" WHERE c.Name = '{inputCountryName}'";

string printTowns = $" SELECT t.Name FROM Towns as t " +

$" JOIN Countries AS c ON c.Id = t.CountryCode " +

$" WHERE c.Name = '{inputCountryName}'";

List<string> output = new List<string>();

SqlConnection connection = new SqlConnection(SERVER\_PATH);

connection.Open();

using (connection)

{

SqlCommand checkTowns = new SqlCommand(checkCommand, connection);

int townCount = (int)checkTowns.ExecuteScalar();

if (townCount == 0)

{

Console.WriteLine("No town names were affected.");

return;

}

else

{

SqlCommand sqlCommand = new SqlCommand(command, connection);

sqlCommand.ExecuteNonQuery();

Console.WriteLine($"{townCount} town names were affected.");

SqlCommand printTownsUpper = new SqlCommand(printTowns, connection);

SqlDataReader reader = printTownsUpper.ExecuteReader();

while (reader.Read())

{

output.Add((string)reader[0]);

}

Console.WriteLine($"[{String.Join((", "), output)}]");

}

}

}

}

}

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| Bulgaria | 3 town names were affected.  [SOFIA, VARNA, BURGAS] |
| Germany | No town names were affected. |

## Remove Villain

Write a program that receives the **ID** of a villain, **deletes him from the database** and **releases his minions** from serving to him. Print on **two** **lines** the name of the deleted villain in format "<**Name**> **was** **deleted**." and the number of minions released in format "<**MinionCount**> **minions** **were** **released**.". Make sure all operations go as planned, otherwise do not make any changes in the database.

If there is no villain in the database with the given ID, print "**No such villain was found**.".

**Solution:**

using System;

using System.Data.SqlClient;

namespace Problem06RemoveVillain

{

public class StartUp

{

private static string SERVER\_PATH = "Server=DESKTOP-1SHT5A0\\SQLEXPRESS; Database=MinionsDB; Integrated Security = true";

static void Main(string[] args)

{

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

string checkVillainsId = $"SELECT Name FROM Villains WHERE Id = {villainId}"; // we either have him or not

string minionsCount = $" SELECT COUNT(\*) FROM MinionsVillains WHERE VillainId = {villainId}"; // get the number of minions under the villain

string deletefromMV = $"DELETE FROM MinionsVillains WHERE VillainId = {villainId}"; // first deletion

string deleteFromVillains = $"DELETE FROM Villains WHERE Id = {villainId}";

SqlConnection connection = new SqlConnection(SERVER\_PATH);

connection.Open();

using (connection)

{

try

{

SqlCommand deleteCommand = new SqlCommand(checkVillainsId, connection);

string count = (string)deleteCommand.ExecuteScalar();

if (count == null) // no villain found

{

Console.WriteLine("No such villain was found.");

}

else // villain found

{

SqlCommand getMinionsNumber = new SqlCommand(minionsCount, connection);

int minionCount = (int)getMinionsNumber.ExecuteScalar(); // how many of the idiots are there

//delete now

try

{

SqlCommand deleteVillainfromMV = new SqlCommand(deletefromMV, connection);

deleteVillainfromMV.ExecuteNonQuery();

try

{

SqlCommand deleteVillainfromVillains = new SqlCommand(deleteFromVillains, connection);

deleteVillainfromVillains.ExecuteScalar();

Console.WriteLine($"{count} was deleted.");

Console.WriteLine($"{minionCount} minions were released.");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

}

}

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 | Gru was deleted.  6 minions were released. |
| 3 | Victor was deleted.  0 minions were released. |
| 101 | No such villain was found. |

## Print All Minion Names

Write a program that **prints all minion names** from the minions table **in the following order:** first record, last record, first + 1, last - 1, first + 2, last - 2 … first + n, last - n.

**Solution:**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

namespace Problem07PrintAllMinionNames

{

public class StartUp

{

private static string SERVER\_PATH = "Server=DESKTOP-1SHT5A0\\SQLEXPRESS; Database=MinionsDB; Integrated Security = true";

static void Main(string[] args)

{

List<string> minionNames = new List<string>();

List<string> resultList = new List<string>();

string getMinionNames = "SELECT Name FROM Minions";

SqlConnection connection = new SqlConnection(SERVER\_PATH);

connection.Open();

using (connection)

{

SqlCommand getNames = new SqlCommand(getMinionNames, connection);

SqlDataReader reader = getNames.ExecuteReader();

while (reader.Read())

{

minionNames.Add((string)reader[0]);

}

}

Console.WriteLine(String.Join(", ", minionNames));// OK

for (int i = 0; i <= (minionNames.Count-1)/2; i++)

{

resultList.Add(minionNames[i]);

for (int j = minionNames.Count - 1 - i ; j > (minionNames.Count - 1) / 2; j--)

{

resultList.Add(minionNames[j]);

break;

}

continue;

}

Console.WriteLine(String.Join(", ", resultList));

}

}

}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 10 | 2 | 9 | 3 | 8 | 4 | 7 | 5 | 6 |

### Example

|  |  |
| --- | --- |
| **Original Order** | **Output** |
| Bob  Kevin  Steward  Jimmy  Vicky  Becky  Jully | Bob  Jully  Kevin  Becky  Steward  Vicky  Jimmy |

## Increase Minion Age

Read from the console minion IDs separated by space. **Increment the age** of those minions **by 1** and make their **names title case**. Finally, **print the name and the age of all minions** in the database, each on a new row in format **"<Name> <Age>**".

**Solution:**

using System;

using System.Data.SqlClient;

using System.Linq;

namespace Problem08IncreaseMinionAge

{

public class StartUp

{

private static string SERVER\_PATH = "Server=DESKTOP-1SHT5A0\\SQLEXPRESS; Database=MinionsDB; Integrated Security = true";

static void Main(string[] args)

{

int[] minionsIds = Console.ReadLine()

.Split()

.Select(int.Parse)

.ToArray();

string printMinions = "SELECT Name, Age FROM Minions";

var connection = new SqlConnection(SERVER\_PATH);

connection.Open();

using (connection)

{

try

{

for (int i = 0; i < minionsIds.Length; i++)

{

string updateMinions = $" UPDATE Minions" +

$" SET Name = UPPER(LEFT(Name, 1)) + SUBSTRING(Name, 2, LEN(Name)), Age += 1 " +

$"WHERE Id = {minionsIds[i]}";

SqlCommand changeData = new SqlCommand(updateMinions, connection);

changeData.ExecuteNonQuery();

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

try

{

SqlCommand printData = new SqlCommand(printMinions, connection);

SqlDataReader reader = printData.ExecuteReader();

while (reader.Read())

{

Console.WriteLine($"{reader[0]} {reader[1]}");

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

}

}

### Example

|  |  |  |
| --- | --- | --- |
| **Minions** | | |
| **Id** | **Name** | **Age** |
| 1 | bob | 14 |
| 2 | stuart | 22 |
| 3 | kevin | 13 |
| 4 | jimmy | 49 |
| 5 | vicky jackson | 26 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 2 1 4 | Bob 15  Stuart 23  kevin 13  Jimmy 50  vicky jackson 26 |  | 5 | bob 14  stuart 22  kevin 13  jimmy 49  Vicky Jackson 27 |

## Increase Age Stored Procedure

Create stored procedure **usp\_GetOlder** (**directly in the database** using **Management Studio** or any other similar tool) that receives **MinionId** and **increases that minion’s age by 1**. Write a program that **uses that stored procedure to increase the age** of a minion whose id will be given as input from the console. After that **print the name and the age** of that minion.

**Solution:**

using System;

using System.Data.SqlClient;

namespace Problem09IncreaseAgeStoredProcedure

{

public class StartUp

{

private static string SERVER\_PATH = "Server=DESKTOP-1SHT5A0\\SQLEXPRESS; Database=MinionsDB; Integrated Security = true";

static void Main(string[] args)

{

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

string command = $"Exec usp\_GetOlder {minionId}";

string displayMinion = $"SELECT Name, Age FROM Minions WHERE Id = {minionId}";

SqlConnection connection = new SqlConnection(SERVER\_PATH);

connection.Open();

using (connection)

{

SqlCommand increaseMinionsAge = new SqlCommand(command, connection);

increaseMinionsAge.ExecuteNonQuery(); //ExecuteScalar();

try

{

SqlCommand displayMinionsDetails = new SqlCommand(displayMinion, connection);

SqlDataReader reader = displayMinionsDetails.ExecuteReader();

using (reader)

{

while (reader.Read())

{

Console.WriteLine($"{reader[0]} – {reader[1]} years old");

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

}

}

### Example

|  |  |  |
| --- | --- | --- |
| **Minions** | | |
| **Id** | **Name** | **Age** |
| 1 | bob | 14 |
| 2 | steward | 22 |
| 3 | kevin | 13 |
| 4 | jimmy | 49 |
| 5 | vicky jackson | 26 |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 | bob – 15 years old |
| 3 | kevin – 14 years old |
| 5 | vicky jackson – 27 years old |