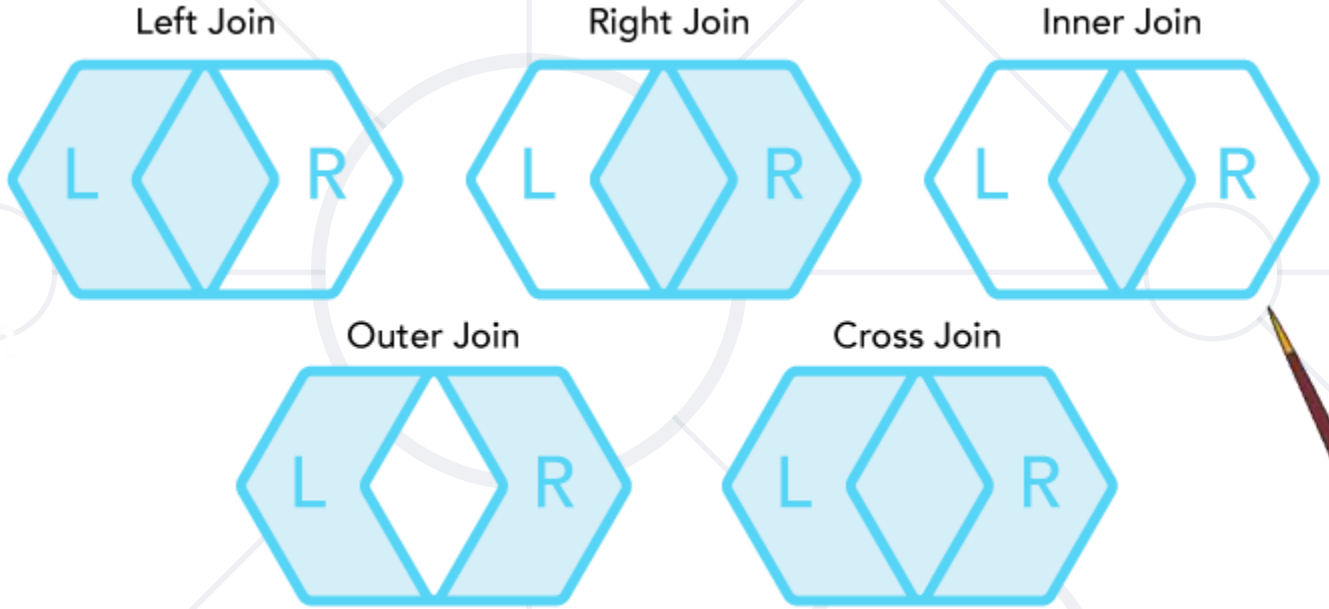
Joins, Subqueries, CTEs



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Table of Contents

Questions

Software University

sli.do

#csharp-db

JOINS

Gathering Data from Multiple Tables

Data from Multiple Tables

- Sometimes you need data from **several tables**

Employees

EmployeeName	DepartmentID
Edward	3
John	NULL

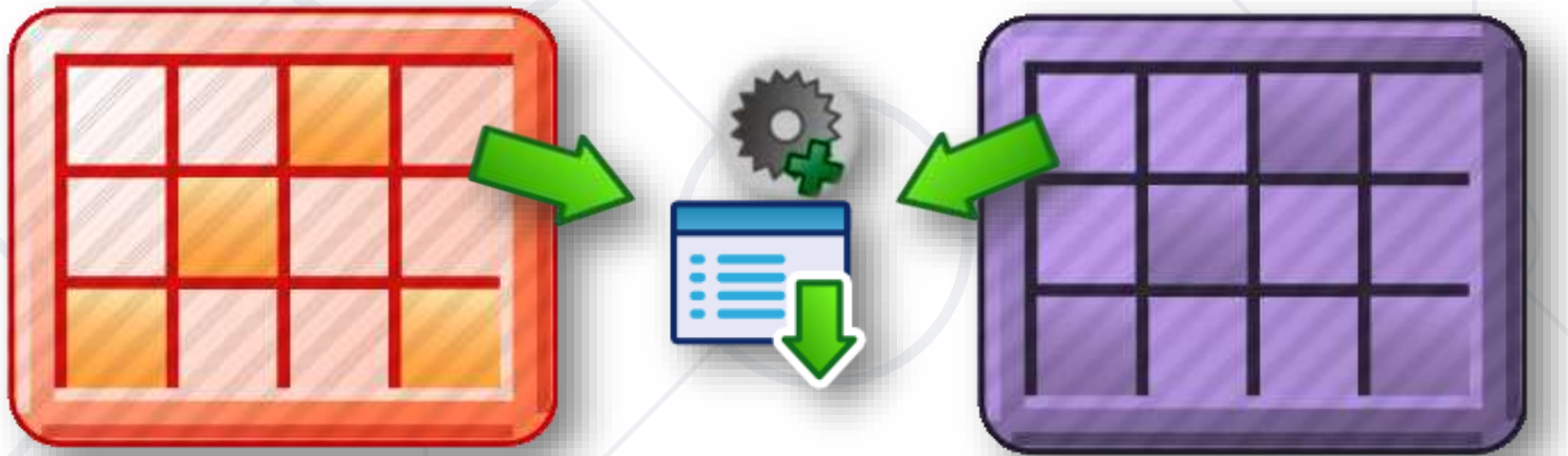
Departments

DepartmentID	DepartmentName
3	Sales
4	Marketing
5	Purchasing

EmployeeName	DepartmentID	DepartmentName
Edward	3	Sales

Types of Joins

- **Inner** joins
- **Left**, **right** and **full outer** joins
- **Cross** joins



INNER vs. OUTER Joins

- **Inner** join
  - Join of two tables returning **only rows matching** the join **condition**
- **Left** (or **right**) **outer** join
  - Returns the results of the inner join as well as unmatched rows from the left (or right) table
- **Full outer** join
  - Returns the results of an **inner join** along with all **unmatched rows**

Inner Join

Employees

EmployeeID	DepartmentID
263	3
270	NULL

Departments

DepartmentID	DepartmentName
3	Sales
4	Marketing
5	Purchasing

Result

EmployeeID	DepartmentID	DepartmentID	DepartmentName
263	3	3	Sales

Inner Join Syntax

```
SELECT * FROM Employees AS e
  INNER JOIN Departments AS d
  ON e.DepartmentID = d.DepartmentID
```

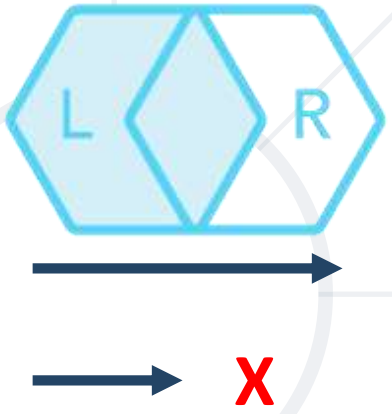
Depatments Table

Join Condition

Left Outer Join

Employees

EmployeeID	DepartmentID
263	3
270	NULL



Departments

DepartmentID	DepartmentName
3	Sales
4	Marketing
5	Purchasing

Result

EmployeeID	DepartmentID	DepartmentID	DepartmentName
263	3	3	Sales
270	NULL	NULL	NULL

Left Outer Join Syntax

```
SELECT * FROM Employees AS e
  LEFT OUTER JOIN Departments AS d
  ON e.DepartmentID = d.DepartmentID
```

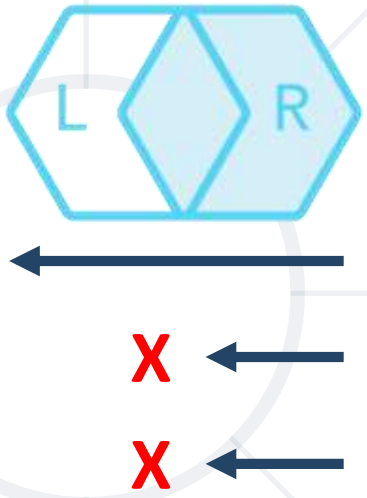
Table Depatments

Join Condition

Right Outer Join

Employees

EmployeeID	DepartmentID
263	3
270	NULL



Departments

DepartmentID	DepartmentName
3	Sales
4	Marketing
5	Purchasing

Result

EmployeeID	DepartmentID	DepartmentID	DepartmentName
263	3	3	Sales
NULL	NULL	4	Marketing
NULL	NULL	5	Purchasing

Right Outer Join Syntax

```
SELECT * FROM Employees AS e
 RIGHT OUTER JOIN Departments AS d
 ON e.DepartmentID = d.DepartmentID
```

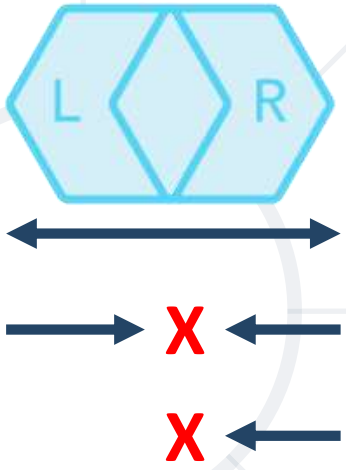
Depatments Table

Join Condition

Full Join

Employees

EmployeeID	DepartmentID
263	3
270	NULL



Departments

DepartmentID	DepartmentName
3	Sales
4	Marketing
5	Purchasing

Result

EmployeeID	DepartmentID	DepartmentID	DepartmentName
263	3	3	Sales
270	NULL	NULL	NULL
NULL	NULL	4	Marketing
NULL	NULL	5	Purchasing

Full Join Syntax

```
SELECT * FROM Employees AS e
  FULL JOIN Departments AS d
  ON e.DepartmentID = d.DepartmentID
```

Depatments Table

Join Condition

Cartesian Product (1)

- This will produce a Cartesian product:

```
SELECT LastName, Name AS
DepartmentName
FROM Employees, Departments
```

- The result:

LastName	DepartmentName
Gilbert	Engineering
Brown	Engineering
…	…
Gilbert	Sales
Brown	Sales

Cross Join

Employees

EmployeeID	DepartmentID
263	3
270	NULL

Departments

DepartmentID	DepartmentName
3	Sales
4	Marketing
5	Purchasing

Result

EmployeeID	DepartmentID	DepartmentID	DepartmentName
263	3	3	Sales
263	3	4	Marketing
263	3	5	Purchasing
270	NULL	3	Sales
270	NULL	4	Marketing
270	NULL	5	Purchasing

Cross Join Syntax

```
SELECT * FROM Employees AS e
 CROSS JOIN Departments AS d
```

Depatments Table

No Join Conditions

Join Overview

Sally	13
John	10
Michael	22
Bob	11
Robin	7
Jessica	15

18	Accounting
10	Marketing
12	HR
22	Engineering
8	Sales
7	Executive

Relation

Join Overview (2)

Software University

- Inner Join

Sally	13
John	10
Michael	22
Bob	11
Robin	7
Jessica	15

18	Accounting
10	Marketing
12	HR
22	Engineering
8	Sales
7	Executive

Join Overview (3)

Software University

- Left Outer Join

		⟷	18	Accounting
Sally	13	⟷	NULL	NULL
John	10	⟷	10	Marketing
			12	HR
Michael	22	⟷	22	Engineering
			8	Sales
Bob	11	⟷	NULL	NULL
Robin	7	⟷	7	Executive
Jessica	15	⟷	NULL	NULL

Join Overview (4)

Software University

- Right Outer Join

NULL	NULL	⟷	18	Accounting
Sally	13			
John	10	⟷	10	Marketing
NULL	NULL	⟷	12	HR
Michael	22	⟷	22	Engineering
NULL	NULL	⟷	8	Sales
Bob	11			
Robin	7	⟷	7	Executive
Jessica	15			

Join Overview (5)

Software University

- Full Outer Join

NULL	NULL	⟷	18	Accounting
Sally	13	⟷	NULL	NULL
John	10	⟷	10	Marketing
NULL	NULL	⟷	12	HR
Michael	22	⟷	22	Engineering
NULL	NULL	⟷	8	Sales
Bob	11	⟷	NULL	NULL
Robin	7	⟷	7	Executive
Jessica	15	⟷	NULL	NULL

Problem: Addresses with Towns

- Display **address information** of all employees in "**SoftUni**" **database**. Select **first 50 employees**.
  - The exact format of data is shown below
  - Order them by FirstName, then by LastName (ascending)
    - Hint: **Use three-way join**

	FirstName	LastName	Town	AddressText
1	A. Scott	Wright	Newport Hills	1400 Gate Drive
2	Alan	Brewer	Kenmore	8192 Seagull Court
3	Alejandro	McGuel	Seattle	7842 Ygnacio Valley Road
4	Alex	Nayberg	Newport Hills	4350 Minute Dr.

Check your solution here: https://judge.softuni.org/Contests/Compete/Index/393#0

Solution: Addresses with Towns

```
SELECT TOP 50 e.FirstName, e.LastName,
  t.Name as Town, a.AddressText
FROM Employees e
  JOIN Addresses a ON e.AddressID = a.AddressID
  JOIN Towns t ON a.TownID = t.TownID
ORDER BY e.FirstName, e.LastName
```

Check your solution here: https://judge.softuni.org/Contests/Compete/Index/393#0

Problem: Sales Employees

- Find **all employees** that are in the "**Sales**" **department**. Use "**SoftUni**" **database**.
  - Follow the specified format:

	EmployeeID	FirstName	LastName	DepartmentName
1	268	Stephen	Jiang	Sales
2	273	Brian	Welcker	Sales
3	275	Michael	Blythe	Sales
4	276	Linda	Mitchell	Sales
5	277	Jillian	Carson	Sales

  - Order them by **EmployeeID**

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/393#2

Solution: Sales Employees

```
SELECT e.EmployeeID, e.FirstName, e.LastName,
   d.Name AS DepartmentName
FROM Employees e
   INNER JOIN Departments d
      ON e.DepartmentID = d.DepartmentID
WHERE d.Name = 'Sales'
ORDER BY e.EmployeeID
```

Departments Table

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/393#2

Problem: Employees Hired After

- Show **all employees** that:
  - Are **hired after** `1/1/1999`
  - Are either **in** "`Sales`" or "`Finance`" department

	FirstName	LastName	HireDate	DeptName
1	Deborah	Poe	2001-01-19 00:00:00	Finance
2	Wendy	Kahn	2001-01-26 00:00:00	Finance
3	Candy	Spoon	2001-02-07 00:00:00	Finance
4	David	Barber	2001-02-13 00:00:00	Finance

  - Sorted by `HireDate` (**ascending**)

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/393#5

Solution: Employees Hired After

```
SELECT e.FirstName, e.LastName, e.HireDate,
  d.Name as DeptName
FROM Employees e
  INNER JOIN Departments d
  ON (e.DepartmentId = d.DepartmentId
  AND e.HireDate > '1/1/1999'
  AND d.Name IN ('Sales', 'Finance'))
ORDER BY e.HireDate ASC
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/393#5

Problem: Employee Summary

- Display information about **employee's manager** and **employee's department**
  - Show only **the first 50** employees
  - The exact format is shown below:

	EmployeeID	EmployeeName	ManagerName	DepartmentName
1	1	Guy Gilbert	Jo Brown	Production
2	2	Kevin Brown	David Bradley	Marketing
3	3	Roberto Tamburello	Terri Duffy	Engineering
4	4	Rob Walters	Roberto Tamburello	Tool Design
5	5	Thierry D'Hers	Ovidiu Cracium	Tool Design

  - Sort by **`EmployeeID`** (**ascending**)

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/393#9

Solution: Employee Summary

```
SELECT TOP 50
  e.EmployeeID,
  e.FirstName + ' ' + e.LastName AS EmployeeName,
  m.FirstName + ' ' + m. LastName AS ManagerName,
  d.Name AS DepartmentName
FROM Employees AS e
  LEFT JOIN Employees AS m ON m.EmployeeID =
e.ManagerID
  LEFT JOIN Departments AS d ON d.DepartmentID =
    e.DepartmentID
  ORDER BY e.EmployeeID ASC
```

Cross Table Selection

Self-join

Table Departments

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/393#9

Subqueries

Query Manipulation On Multiple Levels

Subqueries

- Use a query's result as data for another query

Employees

EmployeeID	Salary
59	19,000
71	43,300
…	…

Query

```
WHERE DepartmentID IN
```

Subquery

DepartmentID	Name
10	Finance

Subquery Syntax

```
SELECT FROM Employees AS e
 WHERE e.DepartmentID IN
  (
   SELECT d.DepartmentID
     FROM Deparments AS d
    WHERE d.Name = 'Finance'
  )
```

Table Depatments

Subquery

Problem: Min Average Salary

- Display **lowest average salary** of **all departments**.
  - Calculate average salary for each department.
  - Then show the value of smallest one.

	MinAverageSalary
1	10866.6666

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/393#10

Solution: Min Average Salary

```
SELECT
  MIN(a.AverageSalary) AS MinAverageSalary
  FROM
  (
    SELECT e.DepartmentID,
           AVG(e.Salary) AS AverageSalary
      FROM Employees AS e
   GROUP BY e.DepartmentID
  ) AS a
```

Subquery

Table Employees

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/393#10

Common Table Expressions

Reusable Subqueries

Common Table Expressions

- **Common Table Expressions** (**CTE**) can be considered as "**named subqueries**"
- They could be used to improve code **readability** and code **reuse**
- Usually, they are positioned in the beginning of the query

```
WITH CTE_Name (ColumnA, ColumnB…)
AS
(
-- Insert subquery here.
)
```

Common Table Expressions Syntax

```
WITH Employees_CTE
   (FirstName, LastName, DepartmentName)
AS
(
   SELECT e.FirstName, e.LastName, d.Name
   FROM Employees AS e
   LEFT JOIN Departments AS d ON
     d.DepartmentID = e.DepartmentID
)

SELECT FirstName, LastName, DepartmentName
   FROM Employees_CTE
```

Temporary Tables

Temporary Tables

- **Temporary tables** are stored in **tempdb**
- Automatically deleted when they are **no longer used**

```
CREATE TABLE #TempTable
(
    -- Add columns here.
)

SELECT * FROM #TempTable
```

Temporary Table Syntax

```
CREATE TABLE #Employees
(
      Id INT PRIMARY KEY,
      FirstName VARCHAR(50) NOT NULL,
      LastName VARCHAR(50),
      Address VARCHAR(50)
)

SELECT * FROM #Employees
```

Types of Temporary Tables

- **Table variables** (`DECLARE @t TABLE`)
  - Visible only to the connection that creates it
- **Local temporary tables** (`CREATE TABLE #t`)
  - Visible only to the connection that creates it
- **Global temporary tables** (`CREATE TABLE ##t`)
  - Visible to everyone
  - Deleted when all connections that have referenced them, have closed
- **Tempdb permanent tables** (`USE tempdb CREATE TABLE t`)
  - Visible to everyone. Deleted when the server is restarted

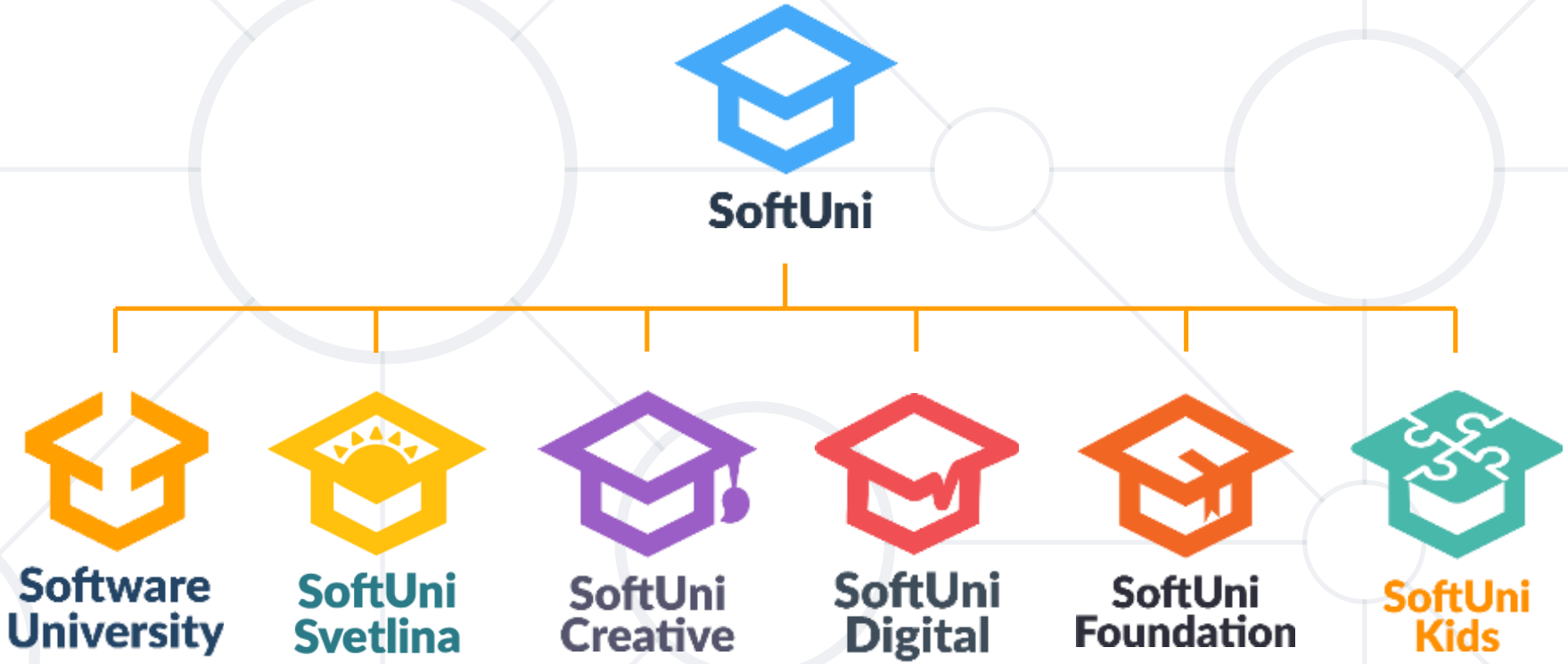
Summary

- **Joins**

```
SELECT * FROM Employees AS e
  JOIN Departments AS d ON
d.DepartmentId = e.DepartmentID
```

- **Subqueries** are used to nest queries
- **CTE's** improve code reuse and readability
- **Indices** improve SQL search performance if used properly

Questions?



SoftUni Diamond Partners

Software University

SCHWARZ

LIDL

Kaufland

SCHWARZ Produktion

prezero

Coca-Cola HBC Bulgaria

POKERSTARS | CAREERS

Postbank

Решения за твоето утре

AMBITIONED

DXC TECHNOLOGY

SOFTWARE GROUP

Bosch.IO

INDEAVR

Serving the high achievers

DRAFT KINGS

PHAR VISION

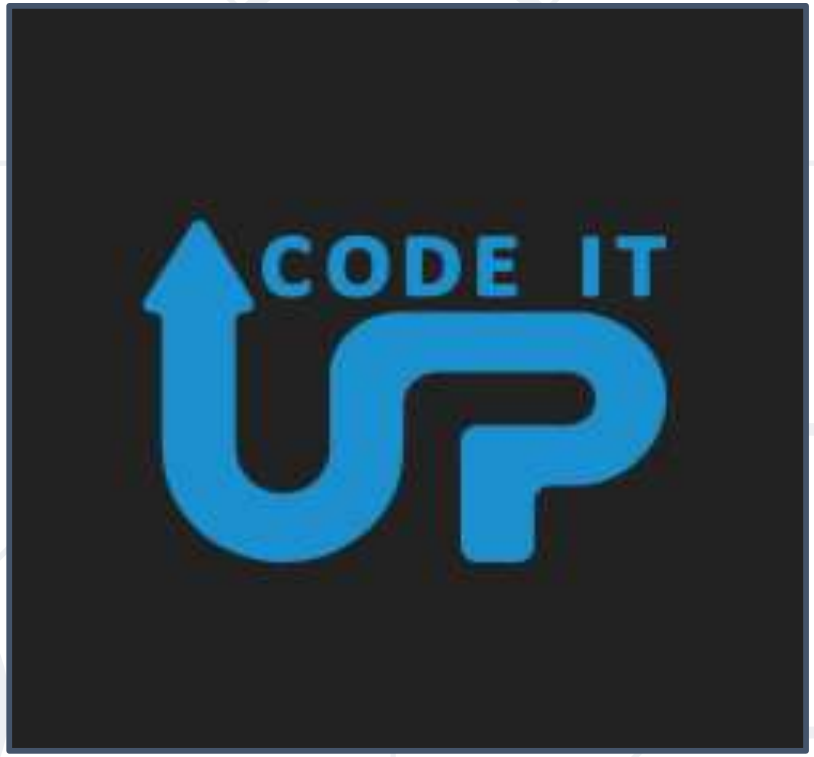
SmartIT

createX

SUPER HOSTING .BG

Educational Partners

Software University



Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg
- © Software University – https://softuni.bg