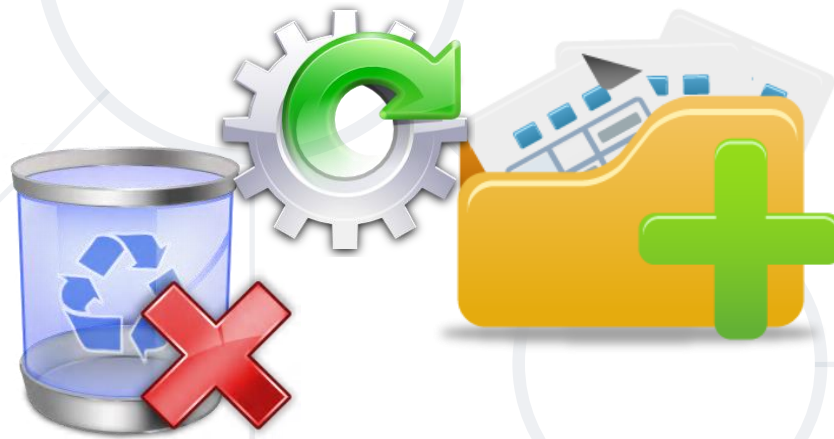


Basic CRUD in SQL Server

Create, Read, Update, Delete
using SQL Queries



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

sli.do

#csharp-db

1. Query Basics
2. Retrieving Data
 - SELECT
 - Views
3. Writing Data
 - INSERT
4. Modifying Existing Records
 - UPDATE and DELETE





Query Basics

SQL and T-SQL Introduction

What Are SQL and T-SQL?

- **Structured Query Language**

- Declarative language
- Close to regular English

```
SELECT FirstName, LastName, JobTitle FROM Employees
```

- Supports definition, manipulation and access control of records
- **Transact-SQL (T-SQL)** – SQL Server's version of SQL
 - Supports control flow (**if**-statements, **loops**)
 - Designed for writing **logic** inside the database

```
SELECT FirstName, LastName, JobTitle FROM Employees
```

```
SELECT * FROM Projects WHERE StartDate = '1/1/2006'
```

```
INSERT INTO Projects(Name, StartDate)  
VALUES ('Introduction to SQL Course', '1/1/2006')
```

```
UPDATE Projects  
SET EndDate = '8/31/2006'  
WHERE StartDate = '1/1/2006'
```

```
DELETE FROM Projects  
WHERE StartDate = '1/1/2006'
```



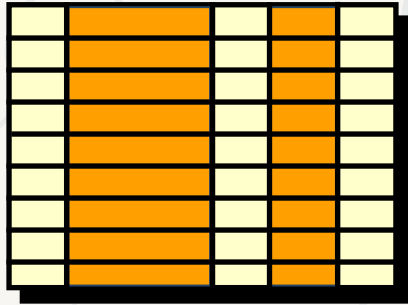
Retrieving Data

Using SQL SELECT

Capabilities of SQL SELECT

Projection

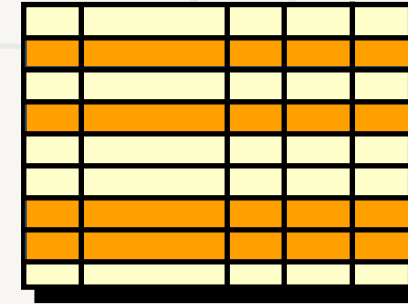
Take a subset of the columns



	Orange	Orange	Orange	Orange	
	Orange	Orange	Orange	Orange	
	Orange	Orange	Orange	Orange	
	Orange	Orange	Orange	Orange	
	Orange	Orange	Orange	Orange	
	Orange	Orange	Orange	Orange	
	Orange	Orange	Orange	Orange	
	Orange	Orange	Orange	Orange	
	Orange	Orange	Orange	Orange	
	Orange	Orange	Orange	Orange	

Selection

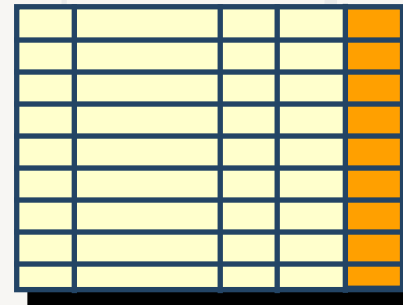
Take a subset of the rows



Orange					
Orange					
Orange					
Orange					
Orange					
Orange					
Orange					

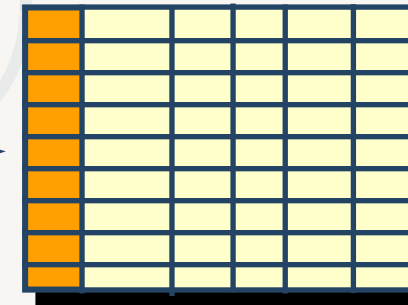
Join

Combine tables by
some column



					Orange
					Orange
					Orange
					Orange
					Orange
					Orange
					Orange
					Orange
					Orange
					Orange

Table 1



Orange					
Orange					
Orange					
Orange					
Orange					
Orange					
Orange					
Orange					
Orange					
Orange					

Table 2

SELECT – Example

- Selecting **all** columns from the "Departments" table

```
SELECT * FROM Departments
```

DepartmentID	Name	ManagerID
1	Engineering	12
2	Tool design	4
3	Sales	273
...

- Selecting **specific** columns

```
SELECT DepartmentId, Name  
FROM Departments
```



DepartmentID	Name
1	Engineering
2	Tool design
3	Sales
...	...

- **Aliases** rename a table or a column heading

Display Name

```
SELECT EmployeeID AS ID,  
       FirstName,  
       LastName  
FROM Employees
```



ID	FirstName	LastName
1	Guy	Gilbert
2	Kevin	Brown
...

- You can shorten fields or clarify abbreviations

```
SELECT c.Duration,  
       c.ACG AS 'Access Control Gateway'  
FROM Calls AS c
```

- You can **concatenate** column names using the **+** operator
 - **String literals** are enclosed in **single quotes**
 - Column names containing **special symbols** use **brackets**

```
SELECT FirstName + ' ' + LastName AS [Full Name],  
       EmployeeID AS [No.]  
FROM Employees
```

Full Name	No.
Guy Gilbert	1
Kevin Brown	2
...	...

Problem: Employee Summary

- Find information about all employees, listing their **full name**, **job title** and **salary**
 - Use **concatenation** to display first and last names as **one field**

	Full Name	JobTitle	Salary
1	Guy Gilbert	Production Technician	12500.00
2	Kevin Brown	Marketing Assistant	13500.00
3	Roberto Tamburello	Engineering Manager	43300.00
4	Rob Walters	Senior Tool Designer	29800.00
5	Thierry D'Hers	Tool Designer	25000.00
6	David Bradley	Marketing Manager	37500.00
7	JoLynn Dobney	Production Supervisor	25000.00
8	Ruth Ellerbrock	Production Technician	13500.00
9	Gail Erickson	Design Engineer	32700.00

- Note: Query **SoftUni** database

Solution: Employee Summary

Concatenation

```
SELECT FirstName + ' ' + LastName  
AS [Full Name],  
JobTitle,  
Salary  
FROM Employees
```

Column Alias

Filtering the Selected Rows

- Use **DISTINCT** to eliminate **duplicate** results

```
SELECT DISTINCT DepartmentID  
FROM Employees
```

- Filter rows by specific **conditions** using the **WHERE** clause

```
SELECT LastName, DepartmentID  
FROM Employees  
WHERE DepartmentID = 1
```

- Other **logical operators** can be used for greater control

```
SELECT LastName, Salary FROM Employees  
WHERE Salary <= 20000
```

Other Comparison Conditions

- Combine conditions using **NOT**, **OR**, **AND** and **brackets**

```
SELECT LastName FROM Employees  
WHERE NOT (ManagerID = 3 OR ManagerID = 4)
```

- Using **BETWEEN** operator to **specify a range**

```
SELECT LastName, Salary FROM Employees  
WHERE Salary BETWEEN 20000 AND 22000
```

- Using **IN** / **NOT IN** to specify **a set of values**

```
SELECT FirstName, LastName, ManagerID  
FROM Employees  
WHERE ManagerID IN (109, 3, 16)
```

Comparing with NULL

- **NULL** is a special value that means missing value
 - Not the same as **0** or a **blank space**
- Checking for **NULL** values



```
SELECT LastName, ManagerId FROM Employees  
WHERE ManagerId = NULL
```

This is always **false**!

```
SELECT LastName, ManagerId FROM Employees  
WHERE ManagerId IS NULL
```

```
SELECT LastName, ManagerId FROM Employees  
WHERE ManagerId IS NOT NULL
```


Sorting Result Sets

- Sort rows with the **ORDER BY** clause
 - ASC**: ascending order, default
 - DESC**: descending order

```
SELECT LastName, HireDate
FROM Employees
ORDER BY HireDate
```

```
SELECT LastName, HireDate
FROM Employees
ORDER BY HireDate DESC
```



LastName	HireDate
Gilbert	1998-07-31
Brown	1999-02-26
Tamburello	1999-12-12
...	...

LastName	HireDate
Valdez	2005-07-01
Tsoflias	2005-07-01
Abbas	2005-04-15
...	...

- Views are **named (saved) queries**
 - Simplify** complex queries
 - Limit access** to data for certain users
- Example: Get employee **names** and **salaries**, by department



```
CREATE VIEW v_EmployeesByDepartment AS
```

```
SELECT FirstName + ' ' + LastName AS [Full Name],  
       Salary  
FROM Employees
```

Executes query

```
SELECT * FROM v_EmployeesByDepartment
```

Problem: Highest Peak

- Create a **view** that selects all information about the **highest peak**
 - Name the view **v_HighestPeak**

```
SELECT * FROM v_HighestPeak
```



	Id	PeakName	Elevation	MountainId
1	68	Everest	8848	9

- Note: Query **Geography** database

Solution: Highest Peak

- **TOP(x)** selects the first x values

```
CREATE VIEW v_HighestPeak  
AS  
SELECT TOP (1) *  
FROM Peaks  
ORDER BY Elevation DESC
```



Sorting column

Greatest value first



Writing Data in Tables

Using SQL INSERT

Inserting Data

- The SQL **INSERT** command

```
INSERT INTO Towns VALUES (33, 'Paris')
```

```
INSERT INTO Projects (Name, StartDate)  
VALUES ('Reflective Jacket', GETDATE())
```

- **Bulk data** can be recorded in a single query, separated by comma

```
INSERT INTO EmployeesProjects  
VALUES (229, 1),  
       (229, 2),  
       (229, 3), ...
```



- Inserting rows into existing table:

List of columns

```
INSERT INTO Projects (Name, StartDate)
SELECT Name + ' Restructuring', GETDATE()
FROM Departments
```

- Using existing records to create a **new table**:

```
SELECT CustomerID, FirstName, Email, Phone
INTO CustomerContacts
FROM Customers
```

New table name

Existing source

- **Sequences** are **special object** in SQL Server
 - Similar to **IDENTITY** fields
- Returns an **incrementing value** every time it's used

```
CREATE SEQUENCE seq_Customers_CustomerID  
            AS INT  
            START WITH 1  
            INCREMENT BY 1
```

```
SELECT NEXT VALUE FOR seq_Customers_CustomerID
```




Modifying Existing Records

Using SQL UPDATE and DELETE

Deleting Data

- Deleting specific rows from a table

```
DELETE FROM Employees WHERE EmployeeID = 1
```

- Note: Don't forget the **WHERE** clause!

Condition


- Delete all rows from a table (works faster than **DELETE**):

```
TRUNCATE TABLE Users
```



Updating Data


- The SQL **UPDATE** command



```
UPDATE Employees
SET LastName = 'Brown'
WHERE EmployeeID = 1
```

New values

```
UPDATE Employees
SET Salary = Salary * 1.10,
      JobTitle = 'Senior' + JobTitle
WHERE DepartmentID = 3
```



- Note: Don't forget the **WHERE** clause!

Problem: Update Projects

- Mark **all unfinished** Projects as being **completed today**
 - Hint: Unfinished projects have their **EndDate** set to **NULL**

Name	EndDate
Classic Vest	NULL
HL Touring Frame	NULL
LL Touring Frame	NULL
...	...



Name	EndDate
Classic Vest	2017-01-23
HL Touring Frame	2017-01-23
LL Touring Frame	2017-01-23
...	...

- Note: Query **SoftUni** database

Solution: Update Projects



```
UPDATE Projects  
SET EndDate = GETDATE()  
WHERE EndDate IS NULL
```

Filter only records
with no value

- **T-SQL** is the language of **SQL Server**

```
SELECT *  
FROM Projects  
WHERE StartDate = '1/1/2006'
```

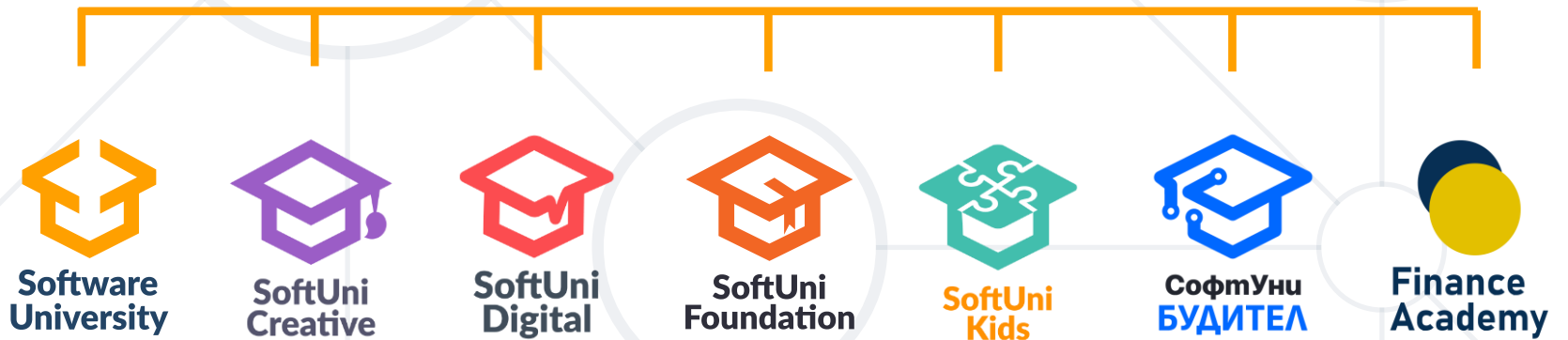
- Queries provide a **flexible** and **powerful method** to **manipulate records**
- **Views** allow us to **store queries** for easier use



Questions?



SoftUni



SoftUni Diamond Partners



- 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



- 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>

