# Exercises: Joins, Subqueries, CTE and Indices

This document defines the **exercise assignments** for the ["Databases Basics - MSSQL" course @ Software University.](https://softuni.bg/courses/databases-basics-ms-sql-server) For problems from 1 to 11 (inclusively) use "**SoftUni**" database and for the other problems – "**Geography**".

## Employee Address Solution:

Write a query that selects:

* **EmployeeId**
* **JobTitle**
* **AddressId**
* **AddressText**

Return the **first 5** rows **sorted** by **AddressId** in **ascending** order.

### Solution :

SELECT TOP (5) e.EmployeeID, e.JobTitle, a.AddressID, a.AddressText

FROM Employees AS e

JOIN Addresses AS a

ON e.AddressID = a.AddressID

### ORDER BY a.AddressID

### Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **EmployeeId** | **JobTitle** | **AddressId** | **AddressText** |
| 142 | Production Technician | 1 | 108 Lakeside Court |
| 30 | Human Resources Manager | 2 | 1341 Prospect St |
| … | … | … | … |

## Addresses with Towns

Write a query that selects:

* **FirstName**
* **LastName**
* **Town**
* **AddressText**

**Sorted** by **FirstName** in **ascending** order then by **LastName**. Select **first 50** employees.

### Solution :

SELECT TOP (50) e.FirstName, e.LastName, t.[Name] AS Town, a.AddressText

FROM Employees AS e

JOIN Addresses AS a

ON e.AddressID = a.AddressID

JOIN Towns AS t

ON t.TownID = a.TownID

ORDER BY e.FirstName, e.LastName

### Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **FirstName** | **LastName** | **Town** | **AddressText** |
| A.Scott | Wright | Newport Hills | 1400 Gate Drive |
| Alan | Brewer | Kenmore | 8192 Seagull Court |
| … | … | … | … |

## Sales Employee

Write a query that selects:

* **EmployeeID**
* **FirstName**
* **LastName**
* **DepartmentName**

**Sorted** by **EmployeeID** in **ascending** order. Select only **employees** from “**Sales**” department.

### Solution :

SELECT e.EmployeeID, e.FirstName, e.LastName, d.[Name] AS DepartmentName

FROM Employees AS e

JOIN Departments AS d

ON e.DepartmentID = d.DepartmentID

WHERE d.[Name] = 'Sales'

ORDER BY e.EmployeeID

### Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **EmployeeID** | **FirstName** | **LastName** | **DepartmentName** |
| 268 | Stephen | Jiang | Sales |
| 273 | Brian | Welcker | Sales |
| … | … | … | … |

## Employee Departments

Write a query that selects:

* **EmployeeID**
* **FirstName**
* **Salary**
* **DepartmentName**

Filter only **employees** with **salary higher than 15000**. Return the **first 5** rows **sorted** by **DepartmentID** in **ascending** order.

### Solution :

SELECT TOP (5) e.EmployeeID, e.FirstName, e.Salary, d.[Name] AS DepartmentName

FROM Employees AS e

JOIN Departments AS d

ON e.DepartmentID = d.DepartmentID

WHERE e.Salary > 15000

ORDER BY e.DepartmentID

### Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **EmployeeID** | **FirstName** | **Salary** | **DepartmentName** |
| 3 | Roberto | 43300.00 | Engineering |
| 9 | Gail | 32700.00 | Engineering |
| … | … | … | … |

## Employees Without Project

Write a query that selects:

* **EmployeeID**
* **FirstName**

Filter only **employees** **without** a **project**. Return the **first 3** rows **sorted** by **EmployeeID** in **ascending** order.

### Solution :

SELECT TOP(3) a.EmployeeID, a.FirstName FROM EmployeesProjects AS ep

FULL JOIN Employees AS a

ON a.EmployeeID = ep.EmployeeID

WHERE ep.EmployeeID IS NULL

ORDER BY a.EmployeeID

### Example:

|  |  |
| --- | --- |
| **EmployeeID** | **FirstName** |
| 2 | Kevin |
| 6 | David |
| … | … |

## Employees Hired After

Write a query that selects:

* **FirstName**
* **LastName**
* **HireDate**
* **DeptName**

Filter only **employees** **hired after 1.1.1999** and are from either **"Sales"** or **"Finance"** departments, s**orted** by **HireDate** (**ascending**).

### Solution :

SELECT e.FirstName, e.LastName, e.HireDate, d.[Name] AS DeptName

FROM Employees AS e

JOIN Departments AS d

ON e.DepartmentID = d.DepartmentID

WHERE e.HireDate > '1999-01-01 00:00:00' AND d.[Name] = 'Finance' OR d.[Name] = 'Sales'

ORDER BY e.HireDate

### Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **FirstName** | **LastName** | **HireDate** | **DeptName** |
| Debora | Poe | 2001-01-19 00:00:00 | Finance |
| Wendy | Kahn | 2001-01-26 00:00:00 | Finance |
| … | … | … | … |

## Employees with Project

Write a query that selects:

* **EmployeeID**
* **FirstName**
* **ProjectName**

Filter only **employees** **with** a **project** which has **started after 13.08.2002** and it is still **ongoing** (no end date). Return the **first 5** rows **sorted** by **EmployeeID** in **ascending** order.

### Solution :

SELECT TOP (5) e.EmployeeID, e.FirstName , p.[Name] AS ProjectName FROM Employees AS e

JOIN EmployeesProjects AS ep

ON e.EmployeeID = ep.EmployeeID

JOIN Projects AS p

ON ep.ProjectID = p.ProjectID

WHERE p.StartDate > '2002-08-13 00:00:00' AND p.EndDate IS NULL

ORDER BY e.EmployeeID

### Example

|  |  |  |
| --- | --- | --- |
| **EmployeeID** | **FirstName** | **ProjectName** |
| 1 | Guy | Racing Socks |
| 1 | Guy | Road Bottle Cage |
| … | … | … |

## Employee 24

Write a query that selects:

* **EmployeeID**
* **FirstName**
* **ProjectName**

Filter all the **projects** of **employee** with **Id 24**. If the project has **started during or** **after** **2005** the **returned** value should be **NULL**.

### Solution :

SELECT e.EmployeeID, e.FirstName,

CASE

WHEN DATEPART(YEAR, p.StartDate) > 2004 THEN NULL

ELSE p.[Name]

END AS ProjectName

FROM Employees AS e

JOIN EmployeesProjects AS ep

ON e.EmployeeID = ep.EmployeeID

JOIN Projects AS p

ON ep.ProjectID = p.ProjectID

WHERE e.EmployeeID = 24

### Example

|  |  |  |
| --- | --- | --- |
| **EmployeeID** | **FirstName** | **ProjectName** |
| 24 | David | NULL |
| 24 | David | Road-650 |
| … | … | … |

## Employee Manager

Write a query that selects:

* **EmployeeID**
* **FirstName**
* **ManagerID**
* **ManagerName**

Filter all **employees** with a **manager** who has **ID** equals to **3 or 7**. Return all the rows, **sorted** by **EmployeeID** in **ascending** order.

### Solution :

SELECT e.EmployeeID, e.FirstName, e.ManagerID, em.FirstName AS ManagerName

FROM Employees AS e

JOIN Employees AS em

ON e.ManagerID = em.EmployeeID

WHERE e.ManagerID IN (3,7)

ORDER BY e.EmployeeID

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **EmployeeID** | **FirstName** | **ManagerID** | **ManagerName** |
| 4 | Rob | 3 | Roberto |
| 9 | Gail | 3 | Roberto |
| … | … | … | … |

## Employee Summary

Write a query that selects:

* **EmployeeID**
* **EmployeeName**
* **ManagerName**
* **DepartmentName**

Show **first 50 employees** with their **managers** and the **departments** they are in (show the departments of the employees). **Order** by **EmployeeID**.

### Solution :

SELECT TOP (50) e.EmployeeID,

CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,

CONCAT(em.FirstName, ' ', em.LastName) AS ManagerName,

d.[Name] AS DepartmentName

FROM Employees AS e

JOIN Employees AS em

ON e.ManagerID = em.EmployeeID

JOIN Departments AS d

ON e.DepartmentID = d.DepartmentID

ORDER BY e.EmployeeID

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **EmployeeID** | **EmployeeName** | **ManagerName** | **DepartmentName** |
| 1 | Guy Gilbert | Jo Brown | Production |
| 2 | Kevin Brown | David Bradley | Marketing |
| 3 | Roberto Tamburello | Terri Duffy | Engineering |
| … | … | … | … |

## Min Average Salary

Write a query that **returns** the value of the **lowest** **average** **salary** of all **departments**.

### Solution :

SELECT TOP(1) AVG(Salary) AS MinAverageSalary FROM Employees

GROUP BY DepartmentID

ORDER BY AVG(Salary)

### Example:

|  |
| --- |
| **MinAverageSalary** |
| 10866.6666 |

## Highest Peaks in Bulgaria

Write a query that selects:

* **CountryCode**
* **MountainRange**
* **PeakName**
* **Elevation**

Filter all **peaks** in **Bulgaria** with **elevation** **over** **2835**. **Return** all the rows **sorted** by **elevation** in **descending** order.

### Solution :

SELECT c.CountryCode,m.MountainRange, p.PeakName, p.Elevation FROM Countries AS c

JOIN MountainsCountries AS mc

ON c.CountryCode = mc.CountryCode

JOIN Mountains AS m

ON mc.MountainId = m.Id

JOIN Peaks AS p

ON m.Id= p.MountainId

WHERE c.CountryCode = 'BG' AND p.Elevation > 2835

ORDER BY p.Elevation DESC

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **CountryCode** | **MountainRange** | **PeakName** | **Elevation** |
| BG | Rila | Musala | 2925 |
| BG | Pirin | Vihren | 2914 |
| … | … | … | … |

## Count Mountain Ranges

Write a query that selects:

* **CountryCode**
* **MountainRanges**

Filter the **count** of the **mountain** **ranges** in the **United** **States**, **Russia** and **Bulgaria**.

### Solution :

SELECT c.CountryCode, COUNT(M.MountainRange) AS MountainRanges FROM Countries AS c

JOIN MountainsCountries AS mc

ON c.CountryCode = mc.CountryCode

JOIN Mountains AS m

ON mc.MountainId = m.Id

WHERE c.CountryCode IN ('US', 'RU', 'BG')

GROUP BY c.CountryCode

### Example

|  |  |
| --- | --- |
| **CountryCode** | **MountainRanges** |
| BG | 6 |
| RU | 1 |
| … | … |

## Countries with Rivers

Write a query that selects:

* **CountryName**
* **RiverName**

Find the **first** **5** **countries** with or without **rivers** in **Africa**. **Sort** them by **CountryName** in **ascending** order.

### Solution :

SELECT TOP (5) c.CountryName, r.RiverName FROM Countries AS c

LEFT JOIN CountriesRivers AS cr

ON c.CountryCode = cr.CountryCode

LEFT JOIN Rivers AS r

ON cr.RiverId = r.Id

WHERE ContinentCode = 'AF'

ORDER BY c.CountryName

### Example

|  |  |
| --- | --- |
| **CountryName** | **RiverName** |
| Algeria | Niger |
| Angola | Congo |
| Benin | Niger |
| Botswana | NULL |
| Burkina Faso | Niger |

## \*Continents and Currencies

Write a query that selects:

* **ContinentCode**
* **CurrencyCode**
* **CurrencyUsage**

Find all **continents** and their **most** **used** **currency**. Filter any **currency** that is used in **only** **one** **country**. **Sort** your results by **ContinentCode**.

### Solution :

SELECT k.ContinentCode,

k.CurrencyCode,

k.CurrencyUsage FROM

(SELECT c.ContinentCode,

c.CurrencyCode ,

COUNT(c.CurrencyCode) AS CurrencyUsage,

DENSE\_RANK() OVER (PARTITION BY c.ContinentCode

ORDER BY COUNT(c.CurrencyCode) DESC) AS Ranking

FROM Countries AS c

GROUP BY c.ContinentCode, c.CurrencyCode

HAVING COUNT(c.CurrencyCode) > 1

) AS k

WHERE k.Ranking = 1

ORDER BY k.ContinentCode

### Example

|  |  |  |
| --- | --- | --- |
| **ContinentCode** | **CurrencyCode** | **CurrencyUsage** |
| AF | XOF | 8 |
| AS | AUD | 2 |
| AS | ILS | 2 |
| EU | EUR | 26 |
| NA | XCD | 8 |
| OC | USD | 8 |

## Countries without any Mountains

Find all the **count** of all **countries,** which **don’t** **have** a **mountain**.

### Solution :

SELECT COUNT(\*) AS Count FROM Countries AS c

LEFT JOIN MountainsCountries AS mc

ON c.CountryCode = mc.CountryCode

LEFT JOIN Mountains AS m

ON mc.MountainId = m.Id

WHERE m.MountainRange IS NULL

### Example

|  |
| --- |
| **Count** |
| 231 |

## Highest Peak and Longest River by Country

For each country, find the elevation of **the highest peak** and **the length of the longest river**, **sorted** by the **highest peak elevation** (from highest to lowest), then by the **longest river length** (from longest to smallest), then by **country name** (alphabetically). Display **NULL** when no data is available in some of the columns. Limit only the **first 5** rows.

### Solution :

SELECT TOP (5) c.CountryName ,

MAX(p.Elevation) AS HighestPeakElevation,

MAX (r.[Length]) AS LongestRiverLength

FROM Countries AS c

LEFT JOIN MountainsCountries AS mc ON c.CountryCode = mc.CountryCode

LEFT JOIN Mountains AS m ON mc.MountainId = m.Id

LEFT JOIN Peaks AS p ON m.Id = p.MountainId

LEFT JOIN CountriesRivers AS cr ON c.CountryCode = cr.CountryCode

LEFT JOIN Rivers AS r ON cr.RiverId = r.Id

GROUP BY c.CountryName

ORDER BY HighestPeakElevation DESC , LongestRiverLength DESC , c.CountryName

### Example

|  |  |  |
| --- | --- | --- |
| **CountryName** | **HighestPeakElevation** | **LongestRiverLength** |
| China | 8848 | 6300 |
| India | 8848 | 3180 |
| Nepal | 8848 | 2948 |
| Pakistan | 8611 | 3180 |
| Argentina | 6962 | 4880 |

## \* Highest Peak Name and Elevation by Country

For each country, find the **name** and **elevation** of **the highest peak**, along with its **mountain**. When no peaks are available in some country, display elevation **0**, "**(no highest peak)**" as **peak name** and "**(no mountain)**" as **mountain name**. When **multiple peaks** in some country have the **same elevation**, display **all of them**. **Sort** the results by **country name alphabetically**, then by **highest peak name alphabetically**. Limit only the **first 5** rows.

### Solution :

SELECT TOP (5) k.CountryName,

k.[Highest Peak Name],

k.[Highest Peak Elevation],

k.Mountain

FROM (

SELECT c.CountryName,

ISNULL(p.PeakName, '(no highest peak)') AS [Highest Peak Name],

ISNULL(p.Elevation, 0) AS [Highest Peak Elevation],

ISNULL(m.MountainRange, '(no mountain)' ) AS Mountain,

DENSE\_RANK() OVER ( PARTITION BY c.CountryName ORDER BY p.Elevation DESC) AS Ranking

FROM Countries AS c

LEFT JOIN MountainsCountries AS mc ON c.CountryCode = mc.CountryCode

LEFT JOIN Mountains AS m ON mc.MountainId = m.Id

LEFT JOIN Peaks AS p ON p.MountainId = m.Id ) AS k

WHERE k.Ranking = 1

### ORDER BY k.CountryName, k.[Highest Peak Name]

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **Country** | **Highest Peak Name** | **Highest Peak Elevation** | **Mountain** |
| Afghanistan | (no highest peak) | 0 | (no mountain) |
| … | … | … | … |
| Argentina | Aconcagua | 6962 | Andes |
| … | … | … | … |
| Bulgaria | Musala | 2925 | Rila |
| Burkina Faso | (no highest peak) | 0 | (no mountain) |
| … | … | … | … |
| United States | Mount McKinley | 6194 | Alaska Range |