# Exercises: Indices and Data Aggregation

This document defines the **exercise** assignments for the ["Databases Basics - MSSQL" course @ Software University](https://softuni.bg/modules/22/csharp-db-may-2024/1473).   
You can check your solutions in the [Judge system](https://judge.softuni.org/Contests/291/Indices-and-Data-Aggregation).

*Mr. Bodrog is a greedy small goblin who is in charge of* ***Gringotts*** *– the biggest wizard bank. His most precious possession is a small database of the deposits in the wizard's world. Taking money is his hobby. He wants your money as well, but unfortunately, you are not a wizard. The only magic you know is how to work with* ***databases****. That's how you got access to the precious data. Mr. Bodrog wants you to send him some reports, otherwise he will send a pack of hungry werewolves after you. You don't want to confront a pack of hungry werewolves, do you?*

Before going on the next task make sure to download the **Gringotts** database.

# Part I – Queries for Gringotts Database

## Records' Count

Import the database and send the **total count of records** from the one and only table to Mr. Bodrog. Make sure nothing gets lost.

### Example

|  |
| --- |
| **Count** |
| 162 |

## Longest Magic Wand

Select the **size** of the **longest magic wand**. Rename the new column appropriately.

### Example

|  |
| --- |
| **LongestMagicWand** |
| 31 |

## Longest Magic Wand Per Deposit Groups

For wizards in **each deposit group** show the **longest magic wand**. Rename the new column appropriately.

### Example

|  |  |
| --- | --- |
| **DepositGroup** | **LongestMagicWand** |
| Blue Phoenix | 31 |
| … | … |

## Smallest Deposit Group Per Magic Wand Size

Select the **two deposit groups** with the **lowest average wand size**.

### Example

|  |
| --- |
| **DepositGroup** |
| Troll Chest |
| Venomous Tongue |

## Deposits Sum

Select **all deposit groups** and their **total deposit sums**.

### Example

|  |  |
| --- | --- |
| **DepositGroup** | **TotalSum** |
| Blue Phoenix | 819598.73 |
| Human Pride | 1041291.52 |
| … | … |

## Deposits Sum for Ollivander Family

Select **all deposit groups** and their **total deposit sums,** but **only for the wizards,** who have their magic wands **crafted by the Ollivander family**.

### Example

|  |  |
| --- | --- |
| **DepositGroup** | **TotalSum** |
| Blue Phoenix | 52968.96 |
| Human Pride | 188366.86 |
| … | … |

## Deposits Filter

Select **all deposit groups** and their total deposit sums, but **only for the wizards,** who have their magic wands **crafted by the Ollivander family**. **Filter** total **deposit amounts lower than 150000**. Order by **total deposit amount** in **descending** order.

### Example

|  |  |
| --- | --- |
| **DepositGroup** | **TotalSum** |
| Troll Chest | 126585.18 |
| … | … |

## Deposit Charge

Create a query that selects:

* **Deposit group**
* **Magic wand creator**
* Minimum **deposit charge** for each group

Select the data in **ascending** order by **MagicWandCreator** and **DepositGroup**.

### Example

|  |  |  |
| --- | --- | --- |
| **DepositGroup** | **MagicWandCreator** | **MinDepositCharge** |
| Blue Phoenix | Antioch Peverell | 30.00 |
| … | … |  |

## Age Groups

Write down a query that creates 7 different groups based on their **age**.

**Age groups** should be as follows:

* [0-10]
* [11-20]
* [21-30]
* [31-40]
* [41-50]
* [51-60]
* [61+]

The query should return

* **Age groups**
* **Count** of wizards in it

### Example

|  |  |
| --- | --- |
| **AgeGroup** | **WizardCount** |
| [11-20] | 21 |
| … | … |

## First Letter

Create a query that returns **all the unique wizard first letters** of their **first names** only if they have **deposit of type Troll Chest**. Order them **alphabetically**. Use GROUP BY for uniqueness.

### Example

|  |
| --- |
| **FirstLetter** |
| A |
| … |

## Average Interest

Mr. Bodrog is highly interested in profitability. He wants to know the **average interest** of all **deposit groups,** split by whether the deposit has **expired or not**. But that's not all. He wants you to select deposits with **start date after** 01/01/1985. Order the data **descending** by **Deposit Group** and **ascending** by **Expiration Flag**.

The output should consist of the following columns:

### Example

|  |  |  |
| --- | --- | --- |
| **DepositGroup** | **IsDepositExpired** | **AverageInterest** |
| Venomous Tongue | 0 | 16.698947 |
| … | … |  |

## \*Rich Wizard, Poor Wizard

*Mr. Bodrog definitely likes his werewolves more than you. This is your last chance to survive! Give him some data to play his favorite game Rich Wizard, Poor Wizard. The rules are simple*:

You **compare** the **deposits of every wizard** with the **wizard after him**. If a wizard is the last one in the database, simply **ignore it**. In the end you have to **sum** the **difference between the deposits**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Host Wizard** | **Host Wizard Deposit** | **Guest Wizard** | **Guest Wizard Deposit** | **Difference** |
| Harry | 10 000 | Tom | 12 000 | -2000 |
| Tom | 12 000 | … | … | … |

At the end your query should return a **single value**: the SUM of all differences.

### Example

|  |
| --- |
| **SumDifference** |
| 44393.97 |

# Part II – Queries for SoftUni Database

*That's it! You no longer work for Mr. Bodrog. You have decided to find a proper job as an analyst in SoftUni.   
It's not a surprise that you will use the* ***SoftUni*** *database. Things get more exciting here!*

## Departments Total Salaries

Create a query that shows the **total sum** of **salaries** for **each department**. **Order** them by **DepartmentID**.  
Your query should return:

* **DepartmentID**

### Example

|  |  |
| --- | --- |
| **DepartmentID** | **TotalSalary** |
| 1 | 241000.00 |
| … | … |

## Employees Minimum Salaries

Select the **minimum salary** from the **employees** for **departments** with **ID** (2, 5, 7) but **only** for those, **hired** **after** 01/01/2000.  
Your query should return:

* **DepartmentID**

### Example

|  |  |
| --- | --- |
| **DepartmentID** | **MinimumSalary** |
| 2 | 25000.00 |
| 5 | 12800.00 |
| … | … |

## Employees Average Salaries

Select all **employees** who earn **more than** 30000 **into a new table.** Then **delete** all **employees** who have **ManagerID = 42** (in the new table). Then **increase the salaries** of all employees with **DepartmentID = 1** by 5000. Finally, select the **average salaries** in **each department**.

### Example

|  |  |
| --- | --- |
| **DepartmentID** | **AverageSalary** |
| 1 | 45166.6666 |
| … | … |

## Employees Maximum Salaries

Find the **max salary** for **each department**. **Filter** those, which have **max salaries** **NOT** in the **range** 30000 – 70000.

### Example

|  |  |
| --- | --- |
| **DepartmentID** | **MaxSalary** |
| 2 | 29800.00 |
| … | … |

## Employees Count Salaries

**Count** the salaries of all **employees,** who don’t have a **manager**.

### Example

|  |
| --- |
| **Count** |
| 4 |

## \*3rd Highest Salary

Find the **third highest salary** in **each department** if there is such.

### Example

|  |  |
| --- | --- |
| **DepartmentID** | **ThirdHighestSalary** |
| 1 | 36100.00 |
| … | … |

## \*\*Salary Challenge

Create a query that returns:

* **FirstName**
* **LastName**
* **DepartmentID**

Select all **employees** who have salary **higher than the average salary** of their respective **departments**. Select **only** the first **10 rows**. **Order** them by **DepartmentID**.

### Example

|  |  |  |
| --- | --- | --- |
| **FirstName** | **LastName** | **DepartmentID** |
| Roberto | Tamburello | 1 |
| … | … |  |