# Exercises: Data Aggregation

This document defines the **exercise assignments** for the ["Java DB - MySQL" course @ Software University](https://softuni.bg/trainings/2352/mysql-may-2019).

Submit your solutions here: <https://judge.softuni.bg/Contests/296/Data-Aggregation-Exercise>.

Mr. Bodrog is a greedy small goblin. His most precious possession is a small database of the deposits in the wizard’s world. Mr. Bodrog wants you to send him some reports.

Get familiar with the **gringotts** database. You will use it in the assignments below.

## 1. Records’ Count

Import the database and send the **total count of records** to Mr. Bodrog. Make sure nothing got lost.

### Example:

|  |
| --- |
| **count** |
| **162** |

## 2. Longest Magic Wand

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

### Example:

|  |
| --- |
| **longest\_magic\_wand** |
| **31** |

## 3. Longest Magic Wand per Deposit Groups

For wizards in each deposit group show the longest magic wand. **Sort result by longest magic** wand for each deposit group **in increasing order**, then by deposit\_group alphabetically. Rename the new column appropriately.

### Example:

|  |  |
| --- | --- |
| **deposit\_group** | **longest\_magic\_wand** |
| Human Pride | 30 |
| … | … |

## 4. Smallest Deposit Group per Magic Wand Size\*

Select the deposit group with the **lowest** average wand size.

### Example:

|  |
| --- |
| **deposit\_group** |
| Troll Chest |

## 5. Deposits Sum

Select all deposit groups and its **total deposit sum**. Sort result by total\_sum in **increasing order**.

### Example:

|  |  |
| --- | --- |
| **deposit\_group** | **total\_sum** |
| Blue Phoenix | 819598.73 |
| … | … |

## 6. Deposits Sum for Ollivander family

Select all deposit groups and its total deposit sum but **only for the wizards who has their magic wand crafted by Ollivander family**. Sort result by deposit\_group **alphabetically**.

### Example:

|  |  |
| --- | --- |
| **deposit\_group** | **total\_sum** |
| Blue Phoenix | 52968.96 |
| Human Pride | 188366.86 |
| … | … |

## 7. Deposits Filter

Select all deposit groups and its total deposit sum but **only for the wizards who has their magic wand crafted by Ollivander family**. After this, **filter** total deposit sums **lower than 150000**. Order by total deposit sum in **descending order**.

### Example:

|  |  |
| --- | --- |
| **deposit\_group** | **total\_sum** |
| Troll Chest | 126585.18 |
| … | … |

## 8. Deposit charge

Create a query that selects:

• **Deposit group**

• **Magic wand creator**

• **Minimum deposit charge for each group**

Group by deposit\_group and magic\_wand\_creator.

Select the data in **ascending** order by magic\_wand\_creator and deposit\_group.

### Example:

|  |  |  |
| --- | --- | --- |
| **deposit\_group** | **magic\_wand\_creator** | **min\_deposit\_charge** |
| Blue Phoenix | Antioch Peverell | 30.00 |
| … | … |  |

## 9. 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**

Sort result by **increasing size** of age groups.

### Example:

|  |  |
| --- | --- |
| **age\_group** | **wizard\_count** |
| [11-20] | 21 |
| … | … |

## 10. First Letter

Write a query that returns all **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:

|  |
| --- |
| **first\_letter** |
| A |
| … |

## 11. Average Interest

Mr. Bodrog is highly interested in profitability. He wants to know the average interest of all deposits 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.

### Example:

|  |  |  |
| --- | --- | --- |
| **deposit\_group** | **is\_deposit\_expired** | **average\_interest** |
| Venomous Tongue | 0 | 16.698947 |
| … | … |  |

## 12. Rich Wizard, Poor Wizard\*

Give Mr. Bodrog 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. At 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:

|  |
| --- |
| **sum\_difference** |
| 44393.97 |

## 13. Employees Minimum Salaries

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 **soft\_uni** database.

Select the minimum salary from the employees for departments with **ID (2,5,7)** but only for those who are **hired after 01/01/2000**. Sort result by department\_id in **ascending** order.

Your query should return:

• department\_id

### Example:

|  |  |
| --- | --- |
| **department\_id** | **minimum\_salary** |
| 2 | 25000.00 |
| … | … |

## 14. Employees Average Salaries

Select all high paid employees who earn **more than 30000** into a new table. Then **delete** all high paid employees who have **manager\_id = 42** from the new table; Then **increase** the salaries of all high paid employees with **department\_id =1** **with 5000** in the new table. Finally, select the **average** salaries in each department from the new table. Sort result by department\_id in **increasing** order.

### Example:

|  |  |
| --- | --- |
| **department\_id** | **avg\_salary** |
| 1 | 45166.6666 |
| … | … |

## 15. Employees Maximum Salaries

Find the **max** salary for each department. Filter those which have max salaries **not in the range 30000 and 70000**. Sort result by department\_id in **increasing** order.

### Example:

|  |  |
| --- | --- |
| **department\_id** | **max\_salary** |
| 2 | 29800.00 |
| … | … |

## 16. Employees Count Salaries

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

## 17. 3rd Highest Salary\*

Find the **third highest salary** in each department if there is such. Sort result by department\_id in **increasing** order.

### Example:

|  |  |
| --- | --- |
| **department\_id** | **third\_highest\_salary** |
| 1 | 36100.00 |
| 2 | 25000.00 |
| … | … |

## 18. Salary Challenge\*\*

Write a query that returns:

first\_name

last\_name

department\_id

for all employees who have salary **higher than the average salary** of their respective departments. Select only the **first 10 rows**. Order by department\_id.

### Example:

|  |  |  |
| --- | --- | --- |
| **first\_name** | **last\_name** | **department\_id** |
| Roberto | Tamburello | 1 |
| Terri | Duffy | 1 |
| Rob | Walters | 2 |
| … | … | ... |

## 19. Departments Total Salaries

Create a query which shows the **total sum of salaries** for each department. Order by department\_id.

Your query should return:

• department\_id

### Example:

|  |  |
| --- | --- |
| **department\_id** | **total\_salary** |
| 1 | 241000.00 |
| … | … |