# Exercises: Data Aggregation

This document defines the **exercise assignments** for the [MySQL course @ Software University](https://softuni.bg/opencourses/databases-basics-mysql).

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.

## Records' Count

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

**select count(id) as total from wizzard\_deposits;**

### Example:

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

## Longest Magic Wand

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

**select max(magic\_wand\_size) as longest\_magic\_wand from wizzard\_deposits;**

### 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.

**select deposit\_group, max(magic\_wand\_size) as longest\_magic\_wand from wizzard\_deposits**

**group by deposit\_group**

**order by longest\_magic\_wand,deposit\_group;**

### 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.

### SELECT deposit\_group

### FROM wizzard\_deposits

### GROUP BY deposit\_group

### order by avg(magic\_wand\_size)

### limit 1;

### 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**.

**SELECT deposit\_group,sum(deposit\_amount) as total\_sum**

**FROM wizzard\_deposits**

**GROUP BY deposit\_group**

**order by total\_sum;**

### 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**.

**SELECT deposit\_group,sum(deposit\_amount) as total\_sum**

**FROM wizzard\_deposits where magic\_wand\_creator='Ollivander family'**

**GROUP BY deposit\_group**

**order by deposit\_group;**

### 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**.

**SELECT deposit\_group,sum(deposit\_amount) as total\_sum**

**FROM wizzard\_deposits where magic\_wand\_creator='Ollivander family'**

**GROUP BY deposit\_group**

**having total\_sum<150000**

**order by total\_sum DESC;**

### 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.

**SELECT deposit\_group,magic\_wand\_creator, min(deposit\_charge) as min\_deposit\_charge**

**FROM wizzard\_deposits**

**GROUP BY deposit\_group,magic\_wand\_creator**

**order by magic\_wand\_creator ASC,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.

**SELECT**

**case**

**when age<11 then '[0-10]'**

**when age>=11 and age<21 then '[11-20]'**

**when age>=21 and age<31 then '[21-30]'**

**when age>=31 and age<41 then '[31-40]'**

**when age>=41 and age<51 then '[41-50]'**

**when age>=51 and age<61 then '[51-60]'**

**when age>60 then '[61+]'**

**end as age\_group, count(id) as wizard\_count from wizzard\_deposits**

**group by age\_group**

**order by age\_group;**

### 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.

**select substring(first\_name,1,1) as first\_letter from wizzard\_deposits where deposit\_group='Troll Chest'**

**group by first\_letter**

**order by first\_letter;**

### 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.

**select deposit\_group, is\_deposit\_expired,avg(deposit\_interest) as average\_interest**

**from wizzard\_deposits where `deposit\_start\_date` > '1985-01-01'**

**group by deposit\_group, is\_deposit\_expired**

**order by deposit\_group desc, is\_deposit\_expired asc;**

### Example:

|  |  |  |
| --- | --- | --- |
| **deposit\_group** | **is\_deposit\_expired** | **average\_interest** |
| Venomous Tongue | 0 | 16.698947 |
| Venomous Tongue | 1 | 13.147500 |
| Troll Chest | 0 | 21.623571 |

## 12. 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

**select department\_id, min(salary) as minimum\_salary from employees**

**where department\_id in (2,5,7) and hire\_date>'2000-01-01'**

**group by department\_id**

**order by department\_id;**

### Example:

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

## 13. 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.

**CREATE TABLE avg\_salaries AS**

**SELECT \***

**FROM employees AS e**

**WHERE e.salary > 30000;**

**DELETE FROM avg\_salaries**

**WHERE manager\_id = 42;**

**UPDATE avg\_salaries**

**SET salary = salary + 5000**

**WHERE department\_id = 1;**

**SELECT a.department\_id, AVG(a.salary)**

**FROM avg\_salaries AS a**

**GROUP BY a.department\_id**

**order by a.department\_id;**

### Example:

|  |  |
| --- | --- |
| **department\_id** | **avg\_salary** |
| 1 | 45166.66666667 |
| … | … |

## 14. 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.

**select department\_id,max(salary) as max\_salary from employees**

**group by department\_id**

**having max\_salary>70000 or max\_salary < 30000**

**order by department\_id;**

### Example:

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

## 15. Employees Count Salaries

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

**select count(salary) from employees where manager\_id is null;**

|  |
| --- |
|  |
| 4 |

## 16. 3rd Highest Salary\*

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

**SELECT**

**`department\_id`,**

**(SELECT DISTINCT**

**`e2`.`salary`**

**FROM**

**`employees` AS `e2`**

**WHERE**

**`e2`.`department\_id` = `e1`.`department\_id`**

**ORDER BY `e2`.`salary` DESC**

**LIMIT 2 , 1) AS `third\_highest\_salary`**

**FROM**

**`employees` AS `e1`**

**GROUP BY `department\_id`**

**HAVING `third\_highest\_salary` IS NOT NULL**

**ORDER BY `department\_id`;**

### Example:

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

## 17. 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, employee\_id.

**SELECT**

**first\_name, last\_name, department\_id**

**FROM**

**employees AS e**

**WHERE**

**e.salary > (SELECT**

**AVG(salary)**

**FROM**

**employees**

**WHERE**

**department\_id = e.department\_id**

**GROUP BY department\_id)**

**ORDER BY e.department\_id,e.employee\_id**

**LIMIT 10;**

### Example:

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

## 18. 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

**select department\_id ,sum(salary) from employees**

**group by department\_id**

**order by department\_id;**

### Example:

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