Data Aggregation

How to Get Data Insights?

SoftUni Team

Technical Trainers

Software University

Software University

https://softuni.bg

Table of Contents

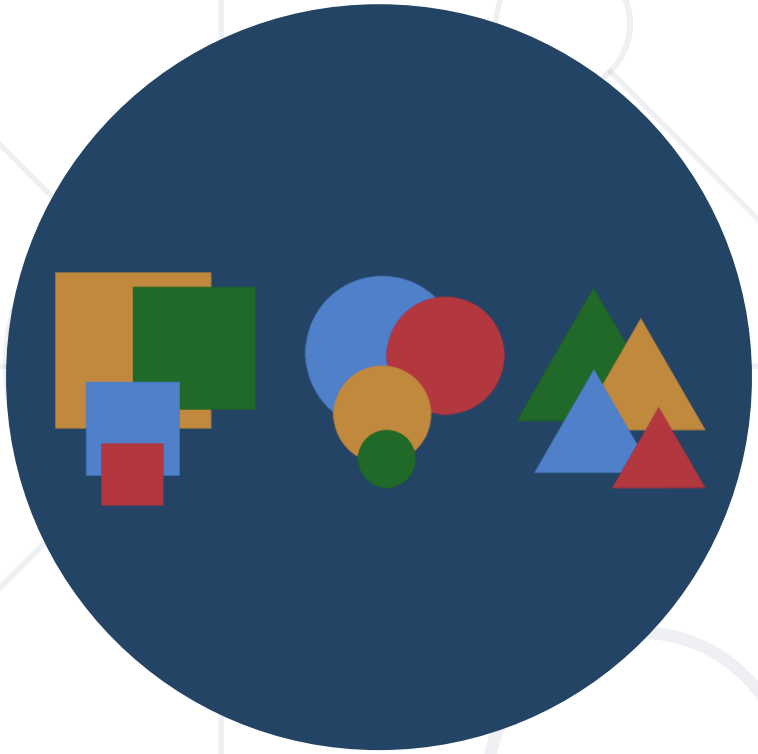
- **Grouping** – consolidating data based on criteria
- **Aggregate Function** – COUNT, SUM, MAX, MIN, AVG …
- **Having** – using predicates while grouping

Questions

Software University

sli.do

#Java-DB



Grouping

Consolidating Data Based On Criteria

Grouping

- Grouping allows taking data into **separate groups** based on a **common property**

Grouping column

employee	department_name	salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

Can be aggregated

GROUP BY

- With **GROUP BY** you can get each separate group and use an "aggregate" function over it (like Average, Min or Max):

```
SELECT e.`job_title`, count(employee_id)
  FROM `employees` AS e
GROUP BY e.`job_title`;
```

Grouping Columns

Problem: Departments Total Salaries

- Write a query which prints the total **sum** of salaries for each **department** in the soft_uni database
  - Order them by DepartmentID (ascending)

employee	department_name	salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

department_id	total_salary
1	20,000
2	30,000
3	15,000

Solution: Departments Total Salaries

Grouping Column

New Column Alias

```
SELECT e.`department_id`,
  SUM(e.`salary`) AS 'Total Salary'
FROM `employees` AS e
GROUP BY e.`department_id`
ORDER BY e.`department_id`;
```

Table Alias

Grouping Columns

Aggregate Functions

COUNT, SUM, MAX, MIN, AVG…

Aggregate Functions

- Used to operate over **one** or **more** groups performing **data analysis** on every one
  - MIN, MAX, AVG, COUNT etc.
- They usually **ignore NULL** values

```
SELECT e.`department_id`,
 MIN(e.`salary`) AS 'MinSalary'
FROM `employees` AS e
GROUP BY e.`department_id`;
```

department_id	MinSalary
1	32700.0000
2	25000.0000
3	23100.0000
4	13500.0000
5	12800.0000
6	40900.0000
7	9500.0000

COUNT

- **COUNT** - counts the values (not nulls) in one or more columns based on grouping criteria

employee	department_name	salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

department_name	SalaryCount
Database Support	2
Application Support	3
Software Support	1

COUNT Syntax

- Note that when we use **COUNT** we will ignore any employee with **NULL** salary.

Grouping Column

New Column Alias

```
SELECT e.`department_id`,
  COUNT(e.`salary`) AS 'Salary Count'
FROM `employees` AS e
GROUP BY e.`department_id`;
```

Grouping Columns

SUM

- **SUM** - sums the values in a column

employee	department_name	salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

department_name	total_salary
Database Support	20,000
Application Support	30,000
Software Support	15,000

SUM Syntax

- If any department has no salaries **NULL** will be displayed.

Grouping Column

```
SELECT e.`department_id`,
  SUM(e.`salary`) AS 'TotalSalary'
FROM `employees` AS e
GROUP BY e.`department_id`;
```

New Column Alias

Table Alias

Grouping Columns

MAX

- **MAX** - takes the maximum value in a column.

employee	department_name	salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

department_name	max_salary
Database Support	15,000
Application Support	15,000
Software Support	15,000

MAX Syntax

Grouping Column

New Column Alias

```
SELECT e.`department_id`,
  MAX(e.`salary`) AS 'MaxSalary'
FROM `employees` AS e
 GROUP BY e.`department_id`;
```

Table Alias

Grouping Columns

MIN

- **MIN** takes the minimum value in a column.

employee	department_name	salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

department_name	min_salary
Database Support	5,000
Application Support	5,000
Software Support	15,000

MIN Syntax

Grouping Column

New Column Alias

```
SELECT e.`department_id`,
  MIN(e.`salary`) AS 'MinSalary'
FROM `employees` AS e
GROUP BY e.`department_id`;
```

Table Alias

Grouping Columns

AVG

- **AVG** calculates the average value in a column.

employee	department_name	salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lila	Application Support	5,000
Fred	Software Support	15,000

department_name	average_salary
Database Support	10,000
Application Support	10,000
Software Support	15,000

Demo: AVG Syntax

Grouping Column

New Column Alias

```
SELECT e.`department_id`,
  AVG(e.`salary`) AS 'AvgSalary'
FROM `employees` AS e
GROUP BY e.`department_id`;
```

Table Alias

Grouping Columns

Having

Using Predicates While Grouping

Having Clause

- The **HAVING** clause is used to filter data based on **aggregate** values.
  - We cannot use it **without** grouping **before** that
- Any Aggregate functions in the "**HAVING**" clause and in the "**SELECT**" statement are executed one time only
- Unlike **HAVING**, the **WHERE** clause filters rows **before** the aggregation

Having Clause: Example

- Filter departments which have **total** salary **less than** 25,000.

Aggregated value

employee	department_name	salary	Total Salary
Adam	Database Support	5,000	20,000
John	Database Support	15,000	
Jane	Application Support	10,000	30,000
George	Application Support	15,000	
Lila	Application Support	5,000	
Fred	Software Support	15,000	15,000

department_name	total_salary
Database Support	20,000
Software Support	15,000

HAVING Syntax

Aggregate Function

Grouping Column

```
SELECT e.`department_id`,
  SUM(e.salary) AS 'TotalSalary'
FROM `employees` AS e
GROUP BY e.`department_id`
HAVING `TotalSalary` < 25000;
```

New Column Alias

Grouping Columns

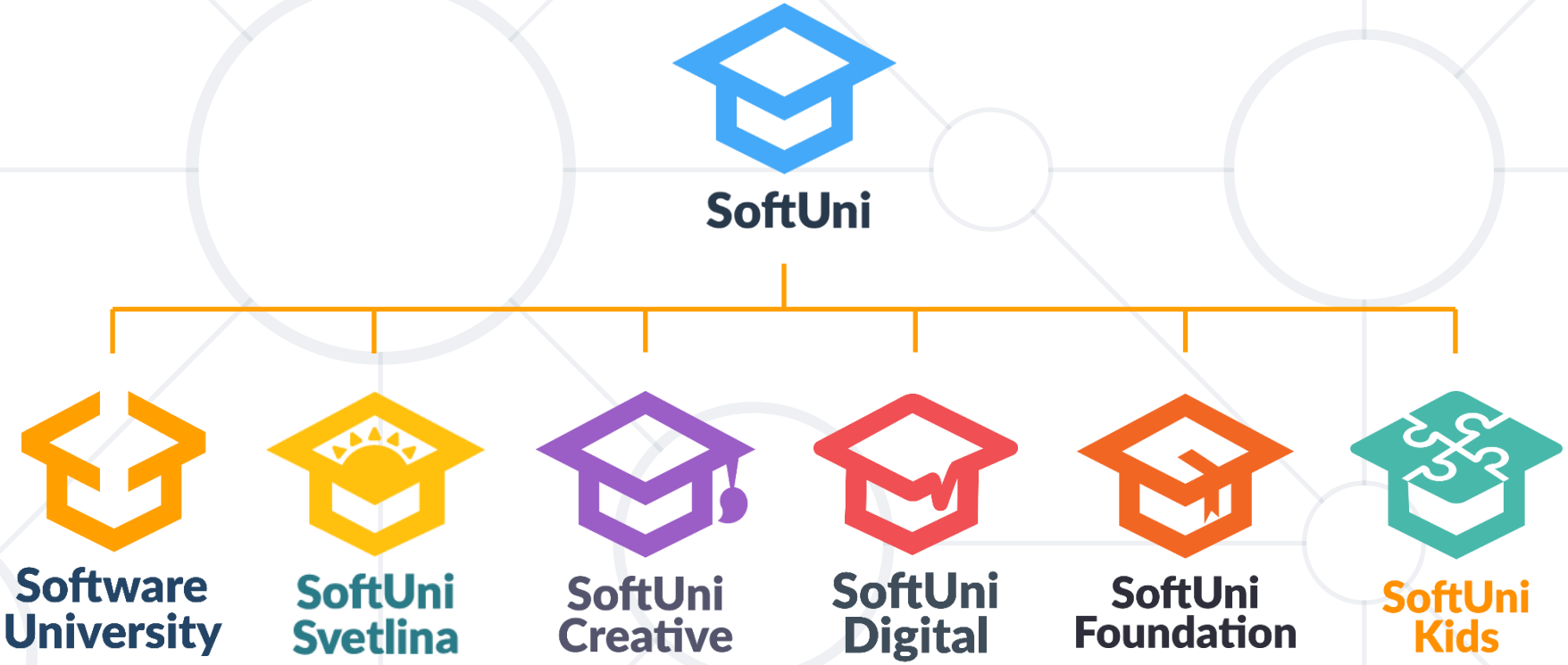
Having Predicate

Summary

- Grouping
- Aggregate Functions
- Having

```
SELECT
  SUM(e.`salary) AS 'TotalSalary'
FROM `employees` AS e
GROUP BY e.`department_id`
HAVING SUM(e.`salary`) < 25000;
```

Questions?



SoftUni Diamond Partners

Software University

XSsoftware

SBTech
we know sports

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

Postbank
Решения за твоето утре

SUPER HOSTING .BG

INDEAVR
Serving the high achievers

INFRAGISTICS®

STEMO®
Computer Systems & Software

SoftUni Organizational Partners

ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://softuni.org
- © Software University – https://softuni.bg