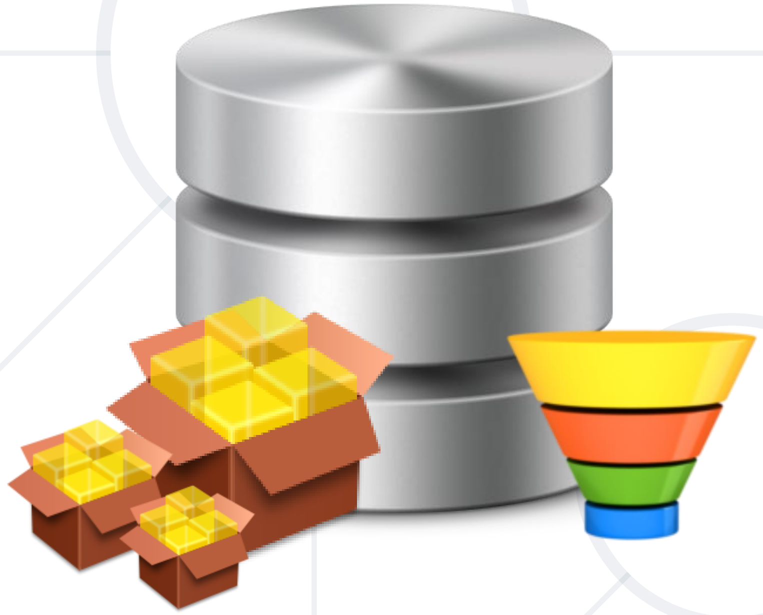
Data Aggregation

How to Get Data Insights?



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

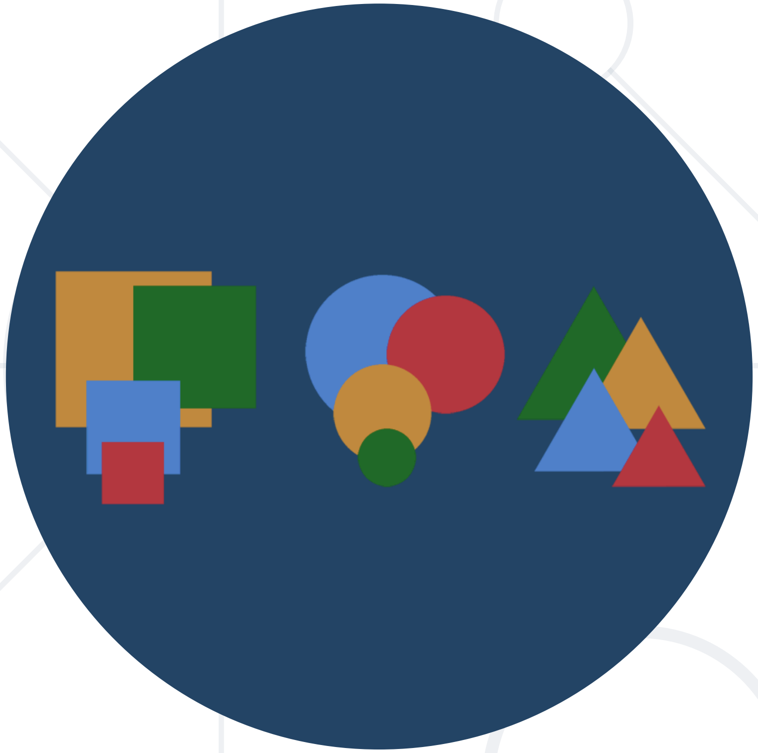
Questions

Software University

sli.do

#python-db

Table of Contents



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

Example: Grouping

- Get the total salaries of all employees **grouped by department**

employee	department_name	salary
Adam	Database Support	5,000
John	Database Support	15,000
Jane	Application Support	10,000
George	Application Support	15,000
Lia	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

Grouping Syntax

- Get each separate group to use an "aggregate" function over it

```
SELECT column_one,
       column_two
FROM table_name

GROUP BY column_one,
         column_two;
```

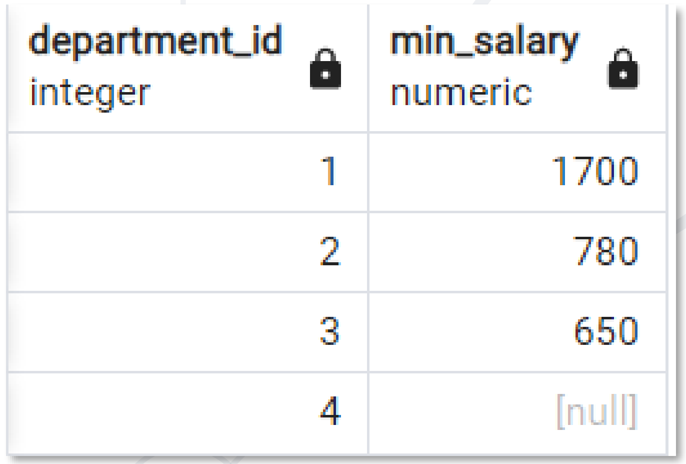
Grouping Columns

Aggregate Functions

COUNT, SUM, MAX, MIN, AVG…

Aggregate Functions (1)

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



department_id integer	min_salary numeric
1	1700
2	780
3	650
4	[null]

Aggregate Functions (2)

- Use an "aggregate" function over each separate group

```
SELECT column_one,
       aggregate_function(column_two)
FROM table_name
GROUP BY column_one;
```

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	employee_count
Database Support	2
Application Support	3
Software Support	1

COUNT Syntax

- Problem & Solution: Departments Info
  - **Count** the number of **employees** per **department**

```
SELECT "department_id",
  COUNT("id") AS "employee_count"
FROM "employees"
GROUP BY "department_id"
ORDER BY "department_id";
```

New Column Alias

Grouping Columns

COUNT and NULL values

- Problem & Solution: Departments Info by Salary Count
- Note, **COUNT** will ignore every employee with a **NULL** value for salary

```
SELECT "department_id",
  COUNT("salary") AS "employee_count"
FROM "employees"
GROUP BY "department_id"
ORDER BY "department_id";
```

Column with NULL values

SUM

- **SUM** - sums the values in a column 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	total_salary
Database Support	20,000
Application Support	30,000
Software Support	15,000

SUM Syntax

- Problem & Solution: Sum Salaries per Department
- If all employees in a department have no salaries, **NULL** will be displayed

```
SELECT "department_id",
  SUM("salary") AS "total_salaries"
FROM "employees"
GROUP BY "department_id"
ORDER BY "department_id";
```

Grouping Column

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	20,000
Lila	Application Support	5,000
Fred	Software Support	15,000

department_name	max_salary
Database Support	15,000
Application Support	20,000
Software Support	15,000

MAX Syntax

- Problem & Solution: Maximum Salary per Department

```sql
SELECT "department_id",
  MAX("salary") AS "max_salary"
FROM "employees"
GROUP BY "department_id"
ORDER BY "department_id";
```

Grouping Column

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

- Problem & Solution: Minimum Salary per Department

```
SELECT "department_id",
  MIN("salary") AS "min_salary"
FROM "employees"
GROUP BY "department_id"
ORDER BY "department_id";
```

Grouping Column

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

AVG Syntax

- Problem & Solution: Average Salary per Department

```
SELECT "department_id",
  AVG("salary") AS "avg_salary"
FROM "employees"
GROUP BY "department_id"
ORDER BY "department_id";
```

Grouping Column

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 that have a **total** salary **less than** 25,000.

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

Aggregated value

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

HAVING Syntax

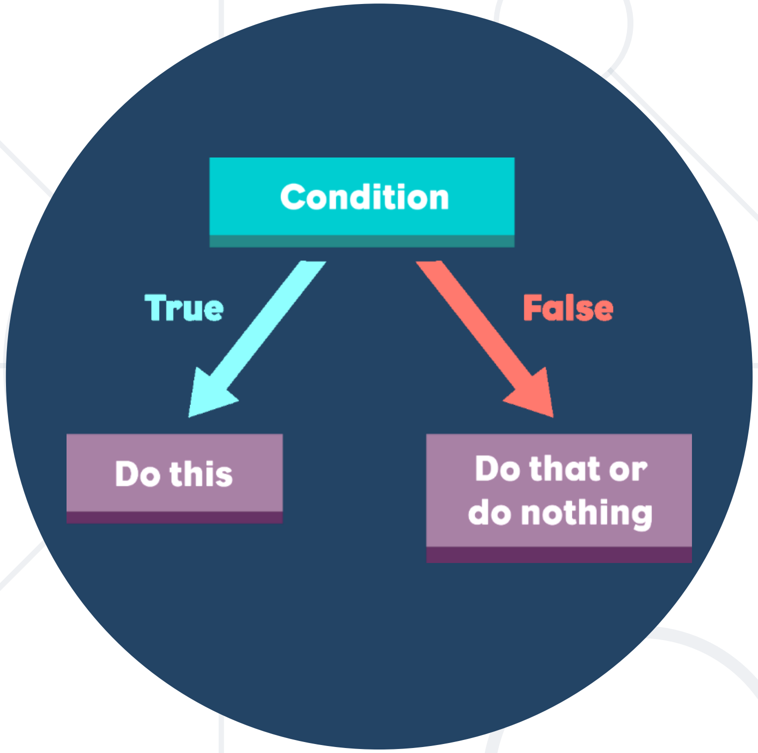
- Problem & Solution: Filter Total Salaries per Department
  - Where total salary is less than 4200

```
SELECT "department_id",
    SUM("salary") AS "Total Salary"
FROM "employees"
GROUP BY "department_id"
HAVING SUM("salary") < 4200
ORDER BY "department_id";
```

Aggregate Function

Grouping Column

Having Predicate

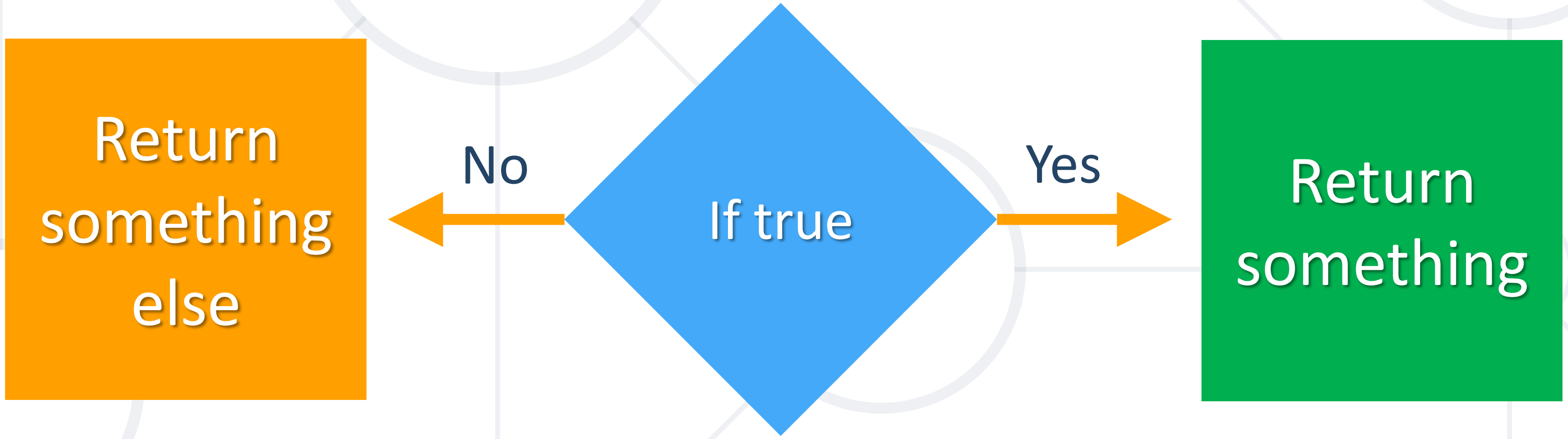


Conditions

Creating Conditional Queries

Conditional Statement

- We can **check** if a condition (**case**) is true or false
- Then, we can proceed, depending on the result



Conditional Statement: Example

- Find the **number of employees** who have a salary **less than `7,500`**; the ones who have salaries **between `7,500` and `12,500`**; and the employees whose salaries are **higher than `12,500`**

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

salary_range	count
Low (< 7,500)	2
Medium (7,500-12,500)	1
High (> 12,500)	3

CASE Expression

- The PostgreSQL **CASE** expression is the same as IF/ELSE statement in other programming languages
- It allows you to add if-else logic to form a powerful query
- The **CASE** expression has two forms: general and simple form
- Can be used in **SELECT, WHERE, GROUP BY** clauses

CASE Expression General Syntax

```
CASE
      WHEN condition_1  THEN result_1
      WHEN condition_2  THEN result_2
      ...
[ELSE else_result]
END AS column_name
```

Boolean Expression

Returns result if True

If none of the above is True

CASE Expression and SELECT

```sql
SELECT id, first_name, last_name, salary,
    CASE
        WHEN department_id = 1 THEN 'Management'
        WHEN department_id = 2 THEN 'Kitchen Staff'
        WHEN department_id = 3 THEN 'Service Staff'
    ELSE 'Other'
    END AS department_name
FROM employees;
```

CASE Expression Simple Syntax

```
CASE expression
      WHEN value_1  THEN result_1
      WHEN value_2  THEN result_2
      ...
ELSE result_n
END AS column_name
```

First evaluated

Compares evaluated expression with value

Returns the result if comparison is True

If none of the above is True

Problem: Department Names

- Write a query to retrieve information from table **`employees`**
  - about the **department names,** according to **department `id`**
  - use **Simple CASE Expression**
    - `1 – "Management"`
    - `2 – "Kitchen Staff"`
    - `3 – "Service Staff"`
    - any other number – `"Other"`

Solution: Department Names

```sql
SELECT id, first_name, last_name,
    TRUNC(salary, 2) AS salary,
department_id,
    CASE department_id
        WHEN 1 THEN 'Management'
        WHEN 2 THEN 'Kitchen Staff'
        WHEN 3 THEN 'Service Staff'
    ELSE 'Other'
    END AS department_name
FROM employees ORDER BY id;
```

CASE Expression in Aggregate Functions

```
SELECT SUM(salary) AS total_salaries,
    SUM(CASE department_id
            WHEN 1 THEN salary*1.15
            WHEN 2 THEN salary*1.10
        ELSE salary*1.05
        END) AS total_increased_salaries
FROM employees;
```

Increasing salary, according to department_id

Computes the future salary expenses if an increase is planned

CASE Expression and GROUP BY

Retrieve products count grouped by new categories

```
SELECT
    CASE
     WHEN category_id IN (1, 2, 3) THEN 'Starters'
     WHEN category_id = 4 THEN 'Mains'
    ELSE 'Desserts'
    END AS "new product category",
  COUNT(id)
FROM products
GROUP BY "new product category";
```

CASE Expression in HAVING

Skip salary ranges that have zero salary count

```
SELECT
     CASE
       WHEN salary < 1000 THEN 'Low (< 1000)'
       WHEN salary <= 3000 THEN 'Middle (1000-3000)'
     ELSE 'High (> 3000)'
     END AS "salary_range",
     COUNT(salary) AS "salary_count"
FROM employees
GROUP BY "salary_range"
HAVING CASE COUNT(salary)
           WHEN 0 THEN  'false'::boolean
       ELSE 'true'::boolean
       END;
```

HAVING requires a boolean value

Summary

- Grouping
- Aggregate Functions
- Having
- Conditional Statements
  - CASE Expression

Questions?

Software University

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter International

INDEAVR Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

Trainings @ Software University (SoftUni)

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

License