Complete SQL Bootcamp

SECTION III: GROUP BY statements

[Video 28] Introduction to GROUP BY

- In this section we are going to be discussing GROUP BY and Aggregate functions.
- GROUP BY will allow us to aggregate data and apply functions to better understand how data is distributed per category.
  - ↳ Section overview
    - Aggregate Functions
    - GROUP BY - Part one - Theory
    - GROUP BY - Part two - Implementation
    - Challenge tasks for GROUP BY
    - HAVING - Filtering with a GROUP BY
    - Challenge tasks for HAVING

[Video 29] Aggregation Functions

- Before we discuss GROUP BY operations, we should understand aggregate functions in general.
- SQL provides a variety of aggregate functions.
  - ↳ The aggregate functions we see here are available in any SQL engine.

- Main idea behind aggregate functions

Take multiple inputs and return a single output.

- There are many aggregate functions!

⇓

We will explore the most common aggregate functions

↳ These are:

AVG() - Returns average value
COUNT() - Returns number of values
MAX() - Returns maximum value
MIN() - Returns minimum value
SUM() - Returns the sum of all values.

↳ Aggregate function calls happen only in the SELECT clause or the HAVING clause.

- Special notes
  - AVG() returns a floating point value with many decimal places (e.g. 2.342418...)
  - You can use ROUND() to specify precision after the decimal.
  - COUNT() - Returns the number of rows, which means by convention we just use COUNT(*)

[Video 30] GROUP BY - Theory

- GROUP BY allows us to aggregate columns per some category
- Let's explore this idea with a simple example:

Category	Data Value
A	10
A	5
B	2
B	4
C	12
C	6

- We need to choose a Categorical column to GROUP BY.
  - Categorical columns are non-continuous.

Can still be numerical!

↳ Class 1, Class 2, --- Class N

↳ Rental rates: $1.99, $2.99, --- etc.

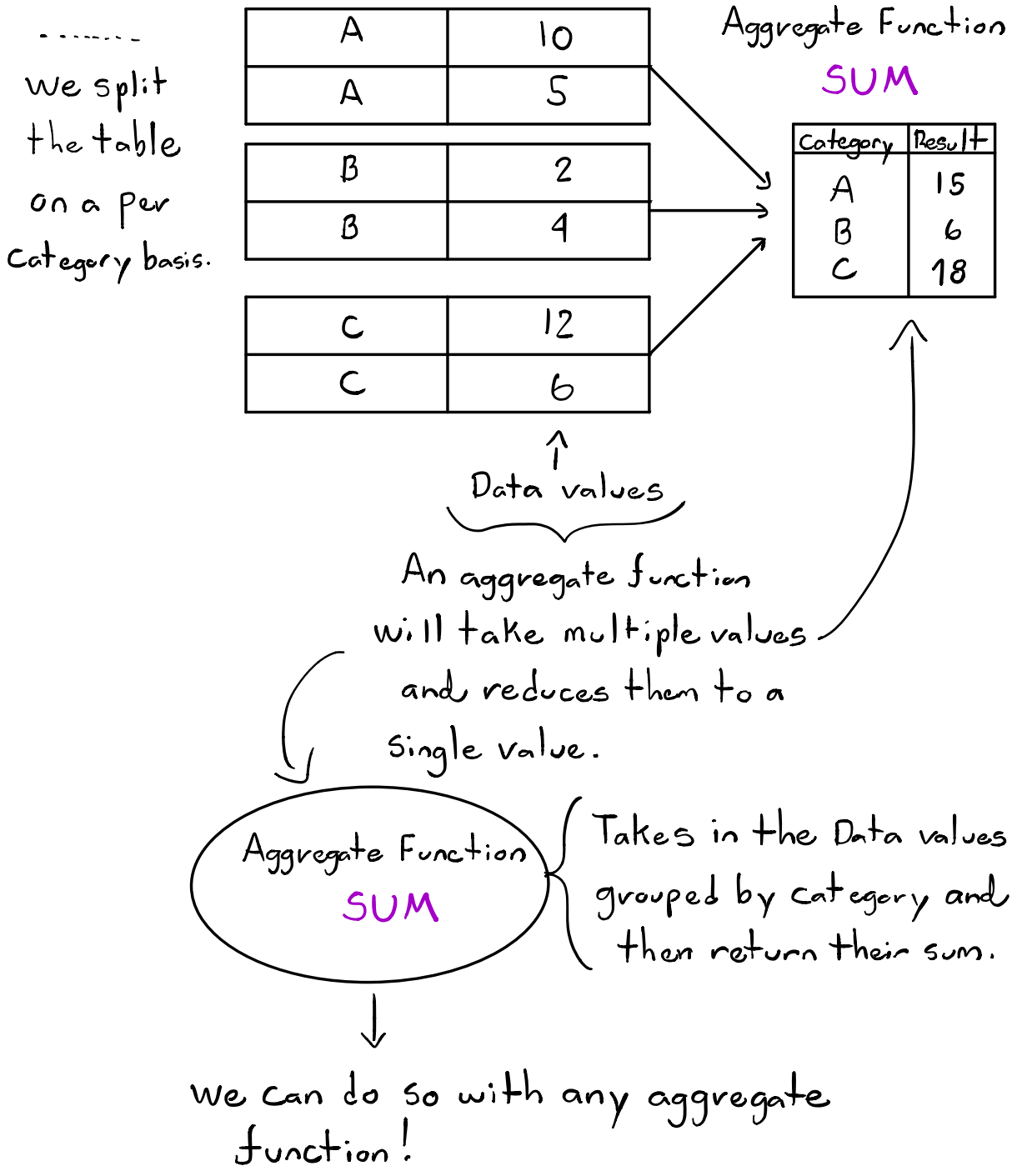
↳ Let's see what happens with a GROUP BY call:

Category	Data Value
A	10
A	5
B	2
B	4
C	12
C	6

A	10
A	5

B	2
B	4

C	12
C	6


we split the table on a per category basis.
A 10
A 5
B 2
B 4
C 12
C 6
Aggregate Function
SUM
Category Result
A 15
B 6
C 18
Data values
An aggregate function will take multiple values and reduces them to a single value.
Aggregate Function
SUM
Takes in the Data values grouped by category and then return their sum.
We can do so with any aggregate function!

- Now, let's explore some general syntax:
  - Most basic GROUP BY call

```
SELECT category_col, AGG(data_col)
FROM table
GROUP BY category_col
```

↳ The GROUP BY clause must appear right after a FROM or WHERE statement.

  - Using a WHERE statement to do some filtering.

```
SELECT category_col, AGG(data_col)
FROM table
WHERE category_col != 'A'
GROUP BY category_col
```

↳ The GROUP BY clause must appear right after a FROM or WHERE statement.

* -> Really important!
  - In the SELECT statement, columns must either have an aggregate function or be in the GROUP BY call.

↳ Consider the following example:

```
SELECT Company, division, SUM(sales)
FROM finance_table
GROUP BY Company, division
```

AGG

↳ In this table we have Companies, divisions per Company and after calling GROUP BY with SUM(sales) as an aggregate function we obtain Sum of sales per division per Company.

↳ Appear in both the SELECT and GROUP BY statements.

↳ ORDER is decided by predence of parameter.

↳ param1, param2, ... paramN

Similar to

ORDER BY

Another example:

```
SELECT Company, division, SUM(sales)
FROM finance_table
WHERE division IN ('marketing', 'transport')
GROUP BY Company, division
```

AGG

→ Filters by division

↳ WHERE statements should not refer to the aggregation result. Later, we will learn to use HAVING to filter those results.

Sorting Results based on aggregate:

```
SELECT Company, SUM(sales)
FROM finance_table
GROUP BY Company
ORDER BY SUM(sales)
```

(sales) → AGG

↳ If you want to sort results based on the aggregate, make sure to reference the entire function.

[Video 33] HAVING

- The HAVING clause allows us to filter after an aggregation has already taken place.
- Consider the following example:

↳ we've already seen we can filter before executing the GROUP BY, but what if we want to filter based on an aggregate function such as SUM?

Consider

```
SELECT Company, SUM(sales)
FROM finance_table
WHERE Company != 'Google'
GROUP BY Company
```

(sales) → AGG

→ ok To call! NO AGG in the WHERE Statement.

- SUM(Sales) -> Is executed until after the GROUP BY statement is executed.

  -> Cannot call WHERE on SUM()!

We must use the HAVING statement to filter off of aggregate results.

We cannot use WHERE to filter based off of aggregate results, because those happen after a WHERE is executed.

Can think of it as a WHERE statement aggregated by a GROUP BY.

```
SELECT Company, SUM(sales)
FROM finance_table
WHERE Company != 'Google'
GROUP BY Company
HAVING SUM(sales) > 100
```

(sales: AGG)