GROUP BY

SQL

- Welcome to this section on 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.

SQL

- 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

Let’s get started!

Aggregate Functions

SQL

- SQL provides a variety of aggregate functions.
- The main idea behind an aggregate function is to take multiple inputs and return a single output.
- https://www.postgresql.org/docs/current/functions-aggregate.html

SQL

- Most Common Aggregate Functions:
  - AVG() - returns average value
  - COUNT() - returns number of values
  - MAX() - returns maximum value
  - MIN() - returns minimum value
  - SUM() - returns the sum of all values

SQL

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

SQL

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

SQL

- Let’s see some examples in our database!

GROUP BY

PART ONE

SQL

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

SQL

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

SQL

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.

Keep in mind, they can still be numerical, such as cabin class categories on a ship (e.g. Class 1, Class 2, Class 3)

SQL

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

Let's now see what happens with a GROUP BY call.

SQL

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

PIERIAN DATA

SQL

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

Aggregate Function
SUM

Category	Result
A	15
B	6
C	18

PIERIAN DATA

SQL

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

Aggregate Function
AVG

Category	Result
A	7.5
B	3
C	9

PIERIAN DATA

SQL

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

Aggregate Function
COUNT

Category	Result
A	2
B	2
C	2

PIERIAN DATA

SQL

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

SQL

- 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.

SQL

- 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.

SQL

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

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

SQL

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

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

SQL

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

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

SQL

- SELECT company, division, SUM(sales)
  FROM finance_table
  GROUP BY company,division

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

SQL

- SELECT company, division, SUM(sales)
  FROM finance_table
  GROUP BY company,division

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

SQL

- SELECT company, division, SUM(sales)
  FROM finance_table
  GROUP BY company,division

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

SQL

- SELECT company, division, SUM(sales)
  FROM finance_table
  WHERE division IN (‘marketing’, ‘transport’)
  GROUP BY company,division

- WHERE statements should not refer to the aggregation result, later on we will learn to use HAVING to filter on those results.

SQL

- SELECT company, SUM(sales)
  FROM finance_table
  GROUP BY company
  ORDER BY SUM(sales)

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

SQL

- SELECT company, SUM(sales)
  FROM finance_table
  GROUP BY company
  ORDER BY SUM(sales)
  LIMIT 5
- If you want to sort results based on the aggregate, make sure to reference the entire function

GROUP BY

PART TWO

SQL

- Let’s jump to our database and work through some GROUP BY examples!

GROUP BY

CHALLENGE TASKS

SQL

- Challenge
- Expected Result
- Hints
- Solution

SQL

- We have two staff members, with Staff IDs 1 and 2. We want to give a bonus to the staff member that handled the most payments. (Most in terms of number of payments processed, not total dollar amount).
- How many payments did each staff member handle and who gets the bonus?

SQL

- Expected Result

	staff_id smallint	count bigint
1	1	7292
2	2	7304

SQL

- Hints
  - Use the payment table
  - Understand the difference between COUNT and SUM

SQL

- Solution
  - SELECT staff_id,COUNT(amount)

    FROM payment

    GROUP BY staff_id

SQL

- Solution
  - SELECT staff_id,COUNT(*)

    FROM payment

    GROUP BY staff_id

SQL

- Corporate HQ is conducting a study on the relationship between replacement cost and a movie MPAA rating (e.g. G, PG, R, etc...).
- What is the average replacement cost per MPAA rating?
  - Note: You may need to expand the AVG column to view correct results

SQL

- Expected Result

Data Output | Explain | Messages | Notifications

	rating mpaa_rating	avg numeric
1	NC-17	20.1376190476190476
2	G	20.1248314606741573
3	PG	18.9590721649484536
4	PG-13	20.4025560538116592
5	R	20.2310256410256410

SQL

- Hints
  - Use the film table
  - Recall that AVG returns back many significant digits, you can either stretch the column or use ROUND() to fix this issue.

SQL

- Solution
  - SELECT rating , AVG(replacement_cost)

    FROM film

    GROUP BY rating

SQL

- Solution
  - SELECT rating ,
    ROUND(AVG(replacement_cost),2)
    FROM film
    GROUP BY rating

SQL

- We are running a promotion to reward our top 5 customers with coupons.
- What are the customer ids of the top 5 customers by total spend?

SQL

- Expected Results

	customer_id smallint	sum numeric
1	148	211.55
2	526	208.58
3	178	194.61
4	137	191.62
5	144	189.60

SQL

- Hints
  - Use the payment table
  - Use ORDER BY
  - Recall you can order by the results of an aggregate function
  - You may want to use LIMIT to view just the top 5

SQL

- Solution
  - SELECT customer_id , SUM(amount)
    FROM payment
    GROUP BY customer_id
    ORDER BY SUM(amount) DESC
    LIMIT 5

HAVING

SQL

- The HAVING clause allows us to filter **after** an aggregation has already taken place.
- Let’s take a look back at one of our previous examples.

SQL

- SELECT company, SUM(sales)
  FROM finance_table
  GROUP BY company

SQL

- SELECT company, SUM(sales)
  FROM finance_table
  WHERE company != ‘Google’
  GROUP BY company

- We’ve already seen we can filter before executing the GROUP BY, but what if we want to filter based on SUM(sales)?

SQL

- SELECT company, SUM(sales)
  FROM finance_table
  WHERE company != ‘Google’
  GROUP BY company

- We can not use WHERE to filter based off of aggregate results, because those happen **after** a WHERE is executed.

SQL

- SELECT company, SUM(sales)
  FROM finance_table
  WHERE company != ‘Google’
  GROUP BY company
  HAVING SUM(sales) > 1000

- HAVING allows us to use the aggregate result as a filter along with a GROUP BY

SQL

- SELECT company, SUM(sales)
  FROM finance_table
  GROUP BY company
  HAVING SUM(sales) > 1000

SQL

- Let’s explore some examples of HAVING in our database!

HAVING

CHALLENGE TASKS

SQL

- These challenge tasks will all utilize the HAVING clause.
  - Challenge
  - Expected Result
  - Hints
  - Solution

SQL

- Challenge
  - We are launching a platinum service for our most loyal customers. We will assign platinum status to customers that have had 40 or more transaction payments.
  - What customer_ids are eligible for platinum status?

SQL

- Expected Result

	customer_id smallint	count bigint
1	144	40
2	526	42
3	148	45

SQL

- Hints
  - Use the payment table
  - Recall any column can be passed into a COUNT() call

SQL

- Solution
  - SELECT customer_id, COUNT(*)

    FROM payment

    GROUP BY customer_id

    HAVING COUNT(*) >= 40;

SQL

- Challenge
  - What are the customer ids of customers who have spent more than $100 in payment transactions with our staff_id member 2?

SQL

- Expected Result

	customer_id smallint	sum numeric
1	187	110.81
2	522	102.80
3	526	101.78
4	211	108.77
5	148	110.78

SQL

- Hints
  - Use the payment table
  - Remember to use WHERE to first filter based on the staff_id , then use the GROUP BY clause

SQL

- Solution
  - SELECT customer_id, SUM(amount)
    FROM payment
    WHERE staff_id = 2
    GROUP BY customer_id
    HAVING SUM(amount) > 100