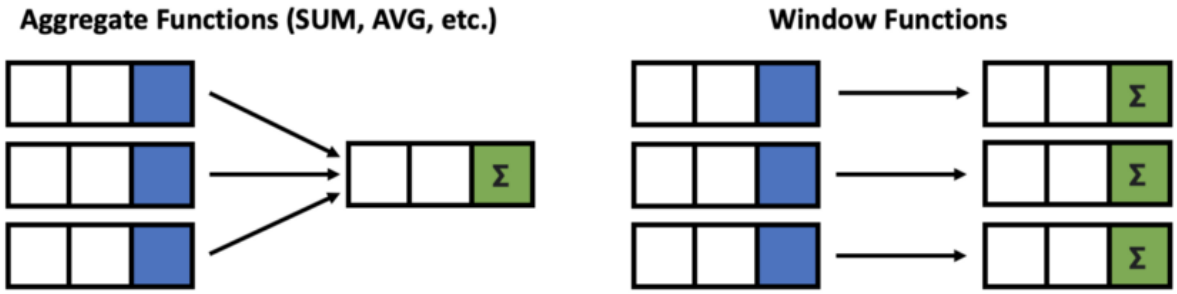
Question: How window functions are different from Aggregate functions? Any similarities between them?

Window functions are similar to the aggregation done in the GROUP BY clause. However, rows are not grouped into a single row, each row retains their separate identity. That is, a window function may return a single value for each row. Here's a good visualization of what I mean by that.



WINDOW VS GROUP BY

Let's say we have some salary data and we want to find to create a column that gives us the average salary for each job title.

JOB_TITLE	SALARY
ANALYST	3100
ANALYST	2900
ANALYST	3250
SALES	1700
SALES	2500
SALES	4100
SALES	1600
SALES	2200
ENGINEER	3500
ENGINEER	3100
ENGINEER	4100

GROUP BY

JOB_TITLE	AVG_SALARY
ANALYST	3083.33333
ENGINEER	3566.66667
SALES	2420

Window Function

JOB_TITLE	SALARY	AVG_SALARY
ANALYST	3100	3083.333333
ANALYST	2900	3083.333333
ANALYST	3250	3083.333333
SALES	1700	2420
SALES	2500	2420
SALES	4100	2420
SALES	1600	2420
SALES	2200	2420
ENGINEER	3500	3566.666667
ENGINEER	3100	3566.666667
ENGINEER	4100	3566.666667

Why use Window Functions?

1. **One major advantage of window functions is that it allows you to work with both aggregate and non-aggregate values all at once because the rows are not collapsed together.**

2. Window functions are also simple to use and read. That is, they can reduce the complexity of your queries, which makes it easier to maintain down the road.

3. In addition, they can help with performance issues. For example, you can use a window function instead of having to do a self-join or cross-join.

This is important because based off of this logical order, window functions are allowed in `SELECT` and `ORDER BY`, but they are not allowed in `FROM`, `WHERE`, `GROUP BY`, or `HAVING` clauses.

ORDER OF OPERATION: number 6

1. FROM, JOIN
2. WHERE
3. GROUP BY
4. aggregate functions
5. HAVING
6. **Window functions**
7. SELECT
8. DISTINCT
9. UNION/INTERSECT/EXCEPT
10. ORDER BY
11. OFFSET
12. LIMIT/FETCH/TOP

SYNTAX

Window Function Syntax

Here's what the generic syntax looks like for a window function in the `SELECT` clause.

```
SELECT <column_1>, <column_2>,
        <window_function> (expression) OVER
        (PARTITION BY <partition_list> ORDER BY <order_list> ROWS frame_clause)
FROM <table_name>
```

Image by Author

There's a lot of words here, so let's look at some definitions:

- **window_function** is the name of the window function we want to use; for example, sum, avg, or row_number (we'll learn more about these later)
- **expression** is the name of the column that we want the window function operated on. This may not be necessary depending on what window_function is used
- **OVER** is just to signify that this is a window function
- **PARTITION BY** divides the rows into partitions so we can specify which rows to use to compute the window function
- **partition_list** is the name of the column(s) we want to partition by

- **ORDER BY** is used so that we can order the rows within each partition. This is optional and does not have to be specified
- **order_list** is the name of the column(s) we want to order by
- **ROWS** can be used if we want to further limit the rows within our partition. This is optional and usually not used
- **frame_clause** defines how much to offset from our current row

LIST OF WINDOW FUNCTIONS