Different Uses of Aggregation

1. Difference between GROUP BY and ORDER BY

- GROUP BY is used to aggregate rows based on one or more columns (e.g., to compute totals or averages per group).
- ORDER BY sorts the final output of a query either in ascending (ASC) or descending (DESC) order.
- While GROUP BY changes the row structure to represent summaries, ORDER BY only affects the display order of the result.

2. Why use HAVING instead of WHERE for filtering aggregates

- WHERE filters rows *before* aggregation happens.
- HAVING filters groups *after* aggregation is complete.
- Example: To get only those courses with an average rating above 4, you must use HAVING AVG(Rating) > 4, not WHERE.

3. Common beginner mistakes with aggregation

- Using SELECT columns that are not included in the GROUP BY clause (without an aggregate function).
- Trying to filter aggregated results using WHERE instead of HAVING.
- Forgetting to alias aggregated results, making queries harder to read.
- Misunderstanding the difference between COUNT(*), COUNT(column), and COUNT(DISTINCT column).

4. When to use COUNT(DISTINCT ...), AVG(...), and SUM(...) together

- Useful in performance and sales analysis, e.g.:
    - COUNT(DISTINCT CustomerID) to get number of unique customers.
    - AVG(OrderValue) to see average order value.
    - SUM(OrderValue) to calculate total sales.
- Often used in dashboards and reports summarizing key metrics.

5. How GROUP BY affects query performance and role of indexes

- GROUP BY can be resource-intensive on large datasets as it requires sorting or hashing internally.
- Indexes on grouped columns can significantly improve performance by reducing the need for full table scans.
- Composite indexes (multi-column) are especially useful when grouping by multiple columns.