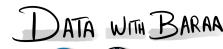




SQL 30 Performance Tips

CHEAT SHEET



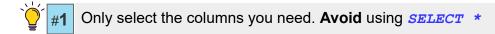


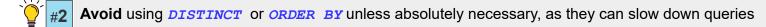






FETCHING DATA





For exploration, **limit** the number of rows using **TOP** to avoid fetching unnecessary data

FILTERING DATA

#4 Create non-clustered indexes on columns frequently used in the WHERE to speed up queries

#5 Avoid functions e.g., UPPER(), YEAR() to columns in the WHERE, as this prevents indexes from being used

#6 Avoid starting string searches with a wildcard (%example), as this disables index usage

Use *IN* instead of multiple *OR* conditions for better readability and performance

JOINING DATA

Understand the performance implications of different join types. Use INNER JOIN when possible for efficiency

Always use **explicit** (ANSI-style) joins (INNER JOIN, LEFT JOIN, etc.) instead of older implicit join syntax

 $^{\circ}$ #10 Ensure that the columns in the $^{\circ}N$ of your joins are **indexed** for optimal performance

#11 Filter before joining large tables to reduce the size of the dataset being joined

* Aggregate before joining large tables to reduce the size of the dataset being joined

Replace OR conditions in join logic with UNION where possible to improve query performance

* Be aware of **nested loops** in your query execution plan. Use **SQL Hints** if needed to optimize performance

#15 Use UNION ALL instead of UNION if duplicates are acceptable, as it is faster

#16 When duplicates are not acceptable, use UNION ALL + DISTINCT instead of UNION for better performance

AGGREGATING DATA

#17 Use columnstore indexes for queries involving heavy aggregations on large tables

#18 Pre-aggregate data and store the results in a separate table for faster reporting

SUBQUERIES

" Understand when to use JOIN, EXISTS, or IN. Avoid IN with large lists as it can be inefficient

#20 Simplify your queries by eliminating redundant logic and conditions by using CTE

DDL

#21 Avoid VARCHAR or TEXT types unnecessarily; choose precise data types to save storage and improve performance

#22 Avoid defining excessive lengths in your data types (e.g., VARCHAR (MAX)) unless truly needed

#23 Use NOT NULL constraints wherever possible to enforce data integrity

#24 Ensure all tables have a clustered primary key to provide structure and improve query performance

Add non-clustered indexes to foreign keys that are frequently queried to speed up lookups

INDEXING

#26 Avoid Over Indexing, as it can slow down insert, update, and delete operations

[] #27 Regularly review and drop unused indexes to save space and improve write performance

#28 Update table statistics weekly to ensure the query optimizer has the most up-to-date information

#29 Reorganize and rebuild fragmented indexes weekly to maintain query performance.

For large tables (e.g., fact tables), **partition the data** and then apply a **columnstore index** for best performance results