To optimize the performance on the database and also on the queries, there are some tips:

- Don’t use the * in your queries. A SELECT * does an overload on the table, I/O and network bandwidth.
- All columns involved in indexes should appear on WHERE and JOIN clauses on the same sequence they appear on index.
- Use VIEWs only when are benefits of doing them.
- Verify if a critical query gains performance by turning it in a stored procedure.
- Avoid too much JOINs on your query: use only what is necessary!
- Avoid cursors at all costs!
- Always restrict the number of rows and columns of your result. It saves disk, memory and network of the database server. Always verify the WHERE clause and use TOP if necessary.
- Verify if the server is not suffering from not-enough-disk-space illness. Reserve at least 30% of available space on your disc.
- SQL Server is *case insensitive*: It does not care about A or a. Save time and don‘t use functions like LOWER and UPPER when comparing VARCHARs.
- The decreasing performance order of operators is: = (faster)>, >=, <, <=, LIKE, <> (slower)
- If a query is slow and the index is not being used by it (check in execution plan), it can be forced by using WITH (INDEX=index_name), right after the table declaration on the FROM clause.
- Use EXISTS or NOT EXISTS instead of IN or NOT IN. IN operators creates a overload on database.
- Try to use BETWEEN instead of IN, too.
- When using LIKE operator, try to leave the wildcards on the right side of the VARCHAR.
- Avoid to use functions on the queries. SUBSTRING is the enemy. Try to use LIKE instead.
- Queries with all operations on the WHERE clause connected by ANDs are processed from the left to right. So, if a operation returns false, all other operations in the right side of it are ignored, because they can‘t change the AND result anyway. It is better then to start your WHERE clause with the operations that returns false most of the time.
- Sometimes is better to make various queries with UNION ALL than a unique query with too much OR operations on WHERE clause.
- When there is a HAVING clause, it is better to filter most results on the WHERE clause and use HAVING only for what it is necessary.
- If there is a need of returning some data fast, even if it is not the whole result, use the FAST option.
- Use, if possible, UNION ALL instead of UNION. The second eliminates all redundant rows and requires more server‘s resources.
- Use less subqueries, or try to nest all of them on a unique block.
- Avoid to do much operations on your WHERE clause. If you are searching for a + 2 > 7, use a > 5 instead.
- Use more variable tables and less temporary tables.
- Use functions to reuse code. But don’t exaggerate on using them!
- To delete all rows on a table, use TRUNCATE TABLE statement instead of DELETE.
- For IDENTITY on a primary key, when are done dozens of simultaneous insertions on, make it a non-clusterized primary key index to avoid bottlenecks.