1. **Identify Frequently Used Columns in WHERE Clauses:** Analyze expected queries and pinpoint the columns that appear most often in WHERE clause conditions. These are prime candidates for indexing.
2. **Consider JOIN Conditions:** If queries frequently involve JOIN operations, creating indexes on the columns used in the JOIN conditions can significantly speed up the process.
3. **Unique vs. Non-Unique Indexes:** Decide between unique and non-unique indexes. Unique indexes enforce uniqueness on the chosen column(s), preventing duplicate values. They are ideal for scenarios where you need to ensure distinct entries (e.g., email addresses). Non-unique indexes simply improve search speed based on the column values.
4. **Composite Indexes (Optional):** For complex queries involving multiple conditions, consider creating composite indexes. These indexes combine multiple columns, allowing faster retrieval when multiple conditions are specified in the WHERE clause.
   example:
   CREATE INDEX [index_name] ON [table_name] ([column1], [column2], ...);