Steps for Tuning a SQL Query (SQL_ID)

Step 1 – Collect Basic Details

Ask the user for essential information such as:

Hostname
Database details
SQL_ID (if available).

Step 2 – Identify SQL_ID
If the SQL_ID is not provided, retrieve it using one of these methods:

Query v$sql using part of the sql_text to find the SQL_ID.
Use v$session_longops to identify long-running SQL operations, especially if the job has been running for an extended period.

Step 3 – Check OEM (Oracle Enterprise Manager)
Check OEM for Top SQL queries that are consuming significant resources or running slow. Confirm the details of the slow-running query with the user.

Step 4 – Get Active Session Details
Obtain the session details for the user by querying gv$session or gv$active_session_history.

Step 5 – Monitor Resource-Intensive Transactions/Jobs
Check for long-running transactions or jobs that are consuming excessive resources. These could be causing the SQL or job to run slowly. Use v$session, v$sysstat, and v$session_wait to track resource consumption.

Step 6 – Check for Locks or Blocking Sessions
Investigate if any locks are affecting the performance:

Use v$lock or dba_locks to find locking sessions.
If there are blocking sessions, gather session details and consult the application team to decide whether to kill the blocking session(s).

Step 7 – Examine SQL History and Plan Changes
Review the SQL's execution history:

Query dba_hist_snapshot and wrh$_sqlstat to check for any plan changes (PHV - Plan Hash Value) that occurred recently.
If a better execution plan existed in the past, consider creating a SQL Profile for optimizer to use that efficient plan.

Step 8 – Check and Update Statistics
Ensure that the table and index statistics are current:
If statistics are out dated/stale, collect them using DBMS_STATS. Check for INDEX status and INDEX stats as well.
Set up regular collection of statistics (weekly or more frequently based on load) to keep the optimizer's decisions accurate.

Step 9 – Analyze Execution Plan
Use DBMS_XPLAN.DISPLAY_CURSOR to get the query's execution plan:

Look for inefficient operations such as Full Table Scans and INDEX FAST FULL SCAN and also cost(%CPU)
Consider creating indexes on columns used in the WHERE clause or joins to improve performance.

Step 10 – Run SQL Tuning Advisor
Execute the SQL Tuning Advisor for the affected SQL_ID and follow the recommendations

Step 11 – Executions/Redo Generated:
Check no of executions and was there increase and also checj the redo size shows the total number of bytes of redo log generated by a particular session. High redo generation can indicate heavy DML operations (INSERT, UPDATE, and DELETE).

Step 12 – AWR Report
If the database is consistently performing slowly, generate an **AWR report** and analyze it to identify performance bottlenecks.