**I-                Troubleshooting incidents or common issues in the database environment**

1-     **Reporting a problem or incident**:

**Important Note:** before running anyquery, the context database should be changed to the corresponding SafeWatch database, the name may differ from one environment to another

a.      Reporting SQL timeouts, locks, blocks, or slowness:

If any problem of this type happens, there are specific steps that the team should follow in order to report it to the DBA with sufficient information, I’ll include the scripts needed to perform each task below:

* + - 1. First, we need to make sure the following settings are enabled in **every query window** so that the report will appear properly to the DBA:

a. Right-click inside the query window and choose “**Query Options**”

b.      Choose “**Grid >> Results**” from the left pane and check the two options “**Include column headers when copying or saving the results”** and “**Retain CR/LF on copy or save**” and then click “**OK”**

* + - 1. Capture current running sessions using the script **“1-Analyze\_Sessions”** and then export the result files to CSV files by right-clicking on the result, choosing “**Save Results As” and**then save the file in CSV format with **UTF-8** encoding, this step should be done for all queries below.
      2. Export server settings by running the script “**2-SQL\_Settings**” and then exporting the result to CSV as done in the previous step.
      3. Export index statistics by running the script “**3-Analyze\_Indexes**” and then exporting the result to CSV.
      4. Export database table sizes by running “**4-Database\_Size**” and then exporting the result to CSV.
      5. Export the recent execution plan cache by running “**5-Cache\_Summary**” and then exporting the result to CSV.

After following the previous steps, files should be attached either to an email or to the corresponding TFS ticket.

b.      Reporting SQL-related exceptions, failures, or errors:

                                                            1-     Perform all steps in **“1-a”**

                                                            2-     Include SQL error logs by running script “**6-Search\_ErrorLog**” and including the results in the attachments

\*\*All scripts are included in the attached file

**II-              SQL Server best practices for enhancing database performance and reducing common problems**

1. **Maintaining data and log file size:**

* 1. Data files (mdf, ndf): data files should be monitored regularly for data growth, data growth settings should also be in MBs only and meet the average size of transactions, sometimes a specific table has a noticeably higher growth rate, in that case, we should consider two options in order to increase the performance of queries using this table:

                                                                        i.     Option one: use table partitioning on that table and use different disks for the different partitions so that main data disk performance will not be affected.

                                                                       ii.     Option two: archiving the historical data that is **rarely used** and inserting them in a new table.

                                                                      iii.     Option three: horizontal partitioning, which means choosing columns that have a large size data type (like**nvarchar** or **ntext**) and inserting them in a separate table, the last step is to create a view to link all tables.

* 1. Log files (**ldf**): the purpose of log files is to store running transactions that will be either committed or rolled back, this file increases in size in environments with heavy transactions, which requires shrinking the log file from time to time, DBAs should create a maintenance job that performs this task regularly (every weekend) in idle times and before any kind of backup is taken (DB backup, VM backup or Disk backup)

1. **Maintaining indexes and statistics:**

* 1. Indexes: fragmentation in indexes increase on a regular basis whenever data is inserted, updated, or deleted from tables, which impacts their performance, so in order to defragment the indexes they need to be rebuilt or reorganized on a regular basis if their fragmentation percentages approach **30%**, it is suggested to perform this task every day outside working hours and before performing backups
  2. Statistics: those are mainly used by the query optimizer to determine the optimal execution plan for a given query, statistics get outdated also after performing CRUD operations, which requires updating, it’s recommended to update statistics using full-scan **after** performing index rebuild/reorganize