The purpose of this lab is to familiarize the participant with the Quality Check customization features that Liquibase Pro offers.

You can run the steps in this lab in your own pre-installed and configured Liquibase Pro enabled environment. It assumes you have a liquibase.properties file or environmental variables setup and referenceable in the git repo directory so that you can connect to a target database.

Clone the https://github.com/lbjenn/vigilant-umbrella/ repo to your system where you will be running Liquibase Pro.

Anywhere you see reference to Visual Studio Code you can also substitute the editor of your choice.

Applications needed for lab:

Visual Studio code

Command prompt session or shell session on mac or linux

Reference → https://docs.liquibase.com/commands/quality-checks/subcommands/home.html

Or run `liquibase checks -help`

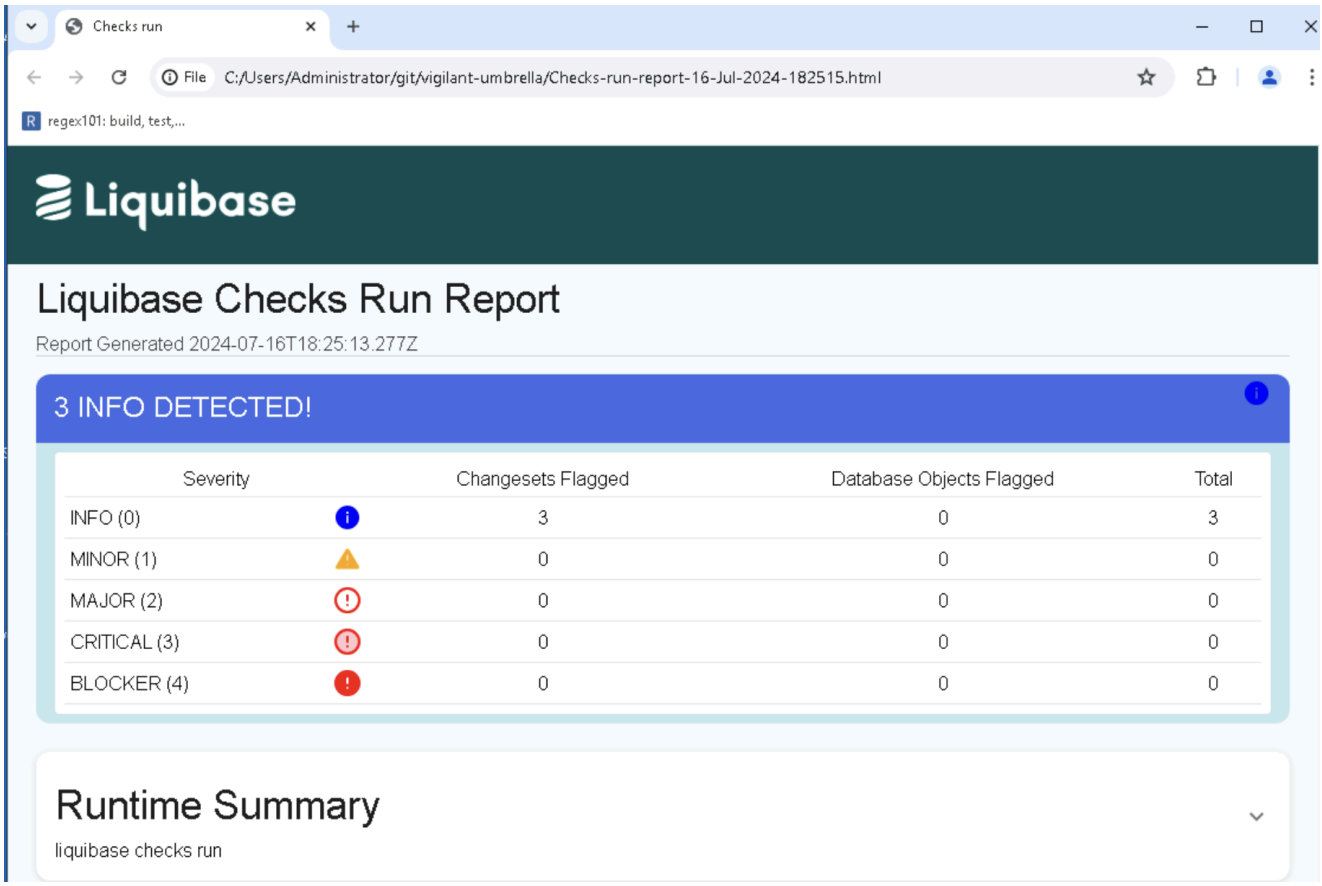
1. On the command line in the vigilant-umbrella directory, run `liquibase checks show` and review output to make note of which checks are enabled.

2. Now execute `liquibase checks run` in your CMD session

   Review this output and identify:
   - What checks were run
   - Which checks were triggered
   - Identify what needs to be changed to correct the offending changeset(s)

3. The execution of the command in the step above will open up a Chrome browser with the 'Liquibase Checks Run Report'

   The top of the report should look something like this:



Review the report and make note of which policy checks were triggered and why.

4. To remove the 2 warnings for the ChangesetContextCheck we need to disable it. In your command session type the following command:
   `liquibase checks disable --check-name=ChangesetContextCheck`
   If the command was successful, you should see output similar to the following:
   `Liquibase command 'checks disable' executed successfully.`

5. Re-run `liquibase checks run`
   Review the Checks Run Report that opens up in the browser and confirm the offending changesets from prior run are not triggering the ChangesetContextCheck QC now. There will still be one warning for SQLGrantWarn quality check.

6. Enable a quality check - check all changesets for Rollbacks by running
   `liquibase checks enable --check-name=RollbackRequired`
   If the command was successful, you should see output similar to the following:
   `Liquibase command 'checks enable' executed successfully.`

7. Now rerun quality checks to see if the output has changed → `liquibase checks run`

8. Notice the Return Code on each triggered quality check. Would any of these keep a deployment pipeline from running further? Why? Or why not? We will discuss at the end of the lab

9. To change the return code for a specific check use the '`liquibase checks customize`' Command as follows

   `liquibase checks customize --check-name=`<short name of check>', for example 'RollbackRequired' - this is the name in first column on left in `liquibase checks show` output → `liquibase checks customize --check-name=RollbackRequired`

   → Enter Severity Code between 1 and 4 → 2 (for example)

10. Run `liquibase checks run` to observe any changes that result in the Checks Run Report for this execution.

END OF LAB