# 1. Problem statement

<table on misconfigurations>

## 1.1 Verifier Misconfiguration

Verifiers are easily misconfigured, and mistakes are not realized until LMK alerts are created. These LMK alerts are often false positives, such as:

* alerting on non-trading days when data isn’t even available,
* and more so for new joiners – deployment failures due to misconfiguration like mismatch between verifier module name and verifier path.

## 1.2 Suboptimal Verification Query

Verifier queries are often not optimized. Many optional filters exist may not be used due to developers’ unfamiliarity with their existence. For example:

* not using `LIMIT` and `min\_key`, causing the query to go through the entire table,
* using `{current\_date}` instead of `{computed\_date}`, causing `days\_start` to not be factored into the filtering parameter.

# 2. Solution

## 2.1 Enforce Required Parameters

Developer will be used to provide all required parameters to ensure completeness.

## 2.2 Scaffold Optimized Verifier Configurations

For every type of verification test, a scaffolded verifier configuration will be provided, where optional filtering arguments will be scaffolded as suggestions for the developer to use, but of course, free to be deleted if unneeded.

## 2.3 Validate Inputs

Validation will be done on every provided input. For example, `module` can only contain alphanumeric characters in snake case, `run\_on` will be validated against the Calendar Service to determine if the MIC or Squarepoint ID is valid.

## 2.4 Optional Help on Parameters

At every input step along the CLI flow, user can type “?” for additional information on the current parameter. An explanation and examples will be given.

## 2.5 Display Upcoming Verifier Schedules

At the end of the CLI flow, a summary will be generated and displayed to the developer based on the inputs. This summary will contain examples of the next few datetimes when the verifier will be run on, for manual validation.

The developer can also run a command `make verifier-summary <verifier\_yaml\_path>` to generate the above summary report on the verifier at the given path.

# 3. `make scraper-new` CLI Flow

# 4. `make scraper-summary <verifier\_yaml\_path>`

At the end of the `make scraper-new`, right after the YAML file is generated, `make scraper-summary <verifier\_yaml\_path>` will be ran on the verifier just created to generate a summary report on the verifier schedule, as well as optimization suggestions. For instance, for a general\_tss test, if `min=10`, `max=10` has been set, `limit=11` will be suggested. For a general\_sql test, if “%(now)s” is in `verification\_command` string but `expand\_psycopg\_params` is not set, then a suggestion will be made to set `expand\_psycopg\_params=true`.