Use Case:

There is a database that requires upgrades (migrations) via the execution of numbered SQL scripts stored in a specified folder, named such as '045.createtable.sql'

- The scripts may contain any simple SQL statement(s) to any table of your choice, e.g. 'INSERT INTO testTable VALUES("045.createtable.sql");'
(you may create any number of files to test your migration script (or script the creation of said files))
They need only include 'INSERT INTO testTable VALUES("045.createtable.sql");' (replace 045 with actual file name)

- There may be gaps in the SQL file name numbering and there isn't always a . (dot) after the beginning number
(ie: you could have 01.initialDeployment.sql, 02.createTableX.sql, 03 createTableY.sql, 4.createX.sql, 5 createX.sql etc.)
- The file name ALWAYS begins with a number, may or may not have a leading 0

- The database upgrade is based on looking up the current version in the database and comparing this number to the numbers in the script names
- The table where the current db version is stored is called 'versionTable', with a single row for the version, called 'version'
- If the version number from the db matches the highest number from the scripts then nothing is executed
- All scripts that contain a number higher than the current db version will be executed against the database in numerical order
- In addition, the database version table is updated after the script execution with the executed script's number
- Your script will be executed automatically via a program, and must satisfy these command line input parameters exactly in order to run:

 - './your-script.your-lang directory-with-sql-scripts username-for-the-db db-host db-name db-password'

- Script should exit with non 0 on failure, should also output INFO and ERROR blocks (such as : “can't read your migrations directory, can't connect to db etc.)

Requirements:
- Supported Languages: Python3, (Golang, Java (Groovy) if Python is not known)
- You can use a MySQL or PostgreSQL database
- Script will be accompanied by instructions on how to test this (ie: a readme file)
- Test will be sent back as a zip folder which will include the .git folder, so we can review the stage of development. One single commit will not suffice.

NOTE: you may use docker-compose, dockerfile or any way of setting up your local db test should come with a readme, if possible explaining how one can set this up locally to test