README

- **Author:** David R. Darling
- **Date:** 25 Nov 2014
- **Email:** david.darling@khepry.com

Overview

The `PyDqsStatsGen.py` Python 3 script will read a standard CSV file and calculate the following statistics for each desired column:

- minimum width
- maximum width
- average width
- non-blanks count
- coverage percent (non-blanks * 100.0 / dataRows)
- frequency of values (by value and occurrence ascending and descending)
    - with percent of total
- frequency of widths
    - with percent of total

The program will produce two file outputs:

- a log file
- an HTML file of statistics

In addition, the program will produce two database outputs:

- five (5) tables in an SQLite database file
- five (5) tables in the specified RDBMS database (e.g. PostgreSQL, MySQL, SQL Server)

The five (5) database tables produced are as follows:

- `ColCountMisMatches`: a table of those rows having a column count that does NOT match the header column count.
- `DqsFileStats`: a table of file-level statistics.
- `DqsMinMaxAvgCvgStats`: a table of `minWidth`, `maxWidth`, `avgWidth`, `nonBlanks`, and `cvgPct` values for each column.
- `DqsValueFreqs`: a table of value frequencies and percents of total for each desired column.
- `DqsWidthFreqs`: a table of width frequencies and percents of total for each desired column.

Running the Program

Navigate to the folder containing the `PyDqsStatsGen.py` script and follow the appropriate instructions below which are dependent upon the operating system under which the program is being executed.

- Consult the `PyDqsStatsGen.ini` section of the documentation below and edit the `PyDqsStatsGen.ini` file as desired, saving it when modifications to it are complete.

- **Linux**
    - Right-click in the PyDqsStatsGen "distributable" folder
    - Within the resulting context menu, Click on "Open in Terminal"

    - At the command prompt within the resulting terminal window, execute the following command:
        - `python3 PyDqsStatsGen.py --iniFilePath PyDqsStatsGen.ini`
- **Windows**
    - Hold down the `Shift` key and right-click in the PyDqsStatsGen "distributable" folder
    - Within the resulting context menu, Click on "Open Command Window here"
    - At the command prompt within the resulting terminal window, execute the following command:
        - `python PyDqsStatsGen.py --iniFilePath PyDqsStatsGen.ini`

Navigate to the `tgtSubFolder` as specified within the `PyDqsStatsGen.ini` file. If the default output and sub-folder structure was utilized, then two output files should appear within the sub-folder specified by the `tgtSubFolder` setting:

- *`srcFileBaseName`*`.html`
- *`srcFileBaseName`*`.sqlite`

PyDqsStatsGen.ini

[DEFAULT]

- `maxRows = 1000`
    - This setting controls how many data rows of the source file will be processed. A setting of '0' (zero) signals the program that the number of data rows to be processed will be unlimited.
- `flushCount = 10000`
    - In the context of the PyDqsStatsGen program, this setting controls how many data rows are to be processed between progress notifications made to the system console and the program's log.
- `maxColCountMisMatches = 0`
    - This setting controls how many data rows can have column count mismatches before the program's terminates further processing. A setting of '0' (zero) signals the program that the number of column count mismatches will be unlimited.
- `maxHtmlCount = 5`
    - This setting controls how many value frequencies will be display for each column on the DQS HTML statistics report. A setting of '0' (zero) signals the program that the number of values frequencies for each column on the DQS HTML statistics report will be unlimited.
- `maxJdbcCount = 5`
    - This setting controls how many value frequencies will be output for each column in the DQS JDBC statistics table. A setting of '0' (zero) signals the program that the number of values frequencies for each column in the DQS JDBC statistics table will be unlimited.
- `runDateFormatString = %A %d %b %Y %I:%M %p %Z`
    - This setting controls the output format of the execution date in both the DQS HTML report and DQS JDBC tables. The formatting codes are consistent with the Python `strftime` function.
- `makoHtmlTemplateName = DqsStatsHtml.mako`
    - This setting specifies the name of the Python MAKO template used to control the generation of the DQS HTML statistics report. The template name should be in form of a valid file path.
- `makoJdbcTemplateName = DqsStatsJdbc.mako`

    - This setting specifies the name of the Python MAKO template used to control the generation of the DQS JDBC statistics tables and the records that populate them. The template name should be in form of a valid file path.

[folders]

- `outFolder = ~/temp`
    - This setting specifies the parent folder of the sub-folders defined below. It should be in the form of a valid folder path. A tilde (~) character is used as a shortcut for the current user's home folder.
- `logSubFolder = logFiles`
    - This setting specifies the sub-folder within which the program's log file(s) will be written.
- `tgtSubFolder = tgtFiles`
    - This setting specifies the sub-folder within which the program's target file(s) will be written. At this point in time, the target files are as follows:
        - The DQS HTML statistics file which, by default, is named: *srcFileBaseName.html*.
        - The DQS SQLite statistics file which, by default, is named: *srcFileBaseName.sqlite*.

[srcSpecs]

- `srcFullPath = '~/path/to/the/source/file.txt'`
    - This setting is used to specify the name of the source file for which statistics are to be calculated.
- `srcDelim = ,`
    - This setting is used to specify the character used as a column delimiter.
- `srcHeaderRows = 1`
    - This setting is used to specify the number of header rows in the source file. A source file must contain at least one header row.
- `dataProvider = NPPES`
    - This is an acronym for the provider of the source data file. This acronym will appear in each row of each table within the resulting statistical databases (e.g. SQLite, MySQL, PostgreSQL, SQL Server).
- `acceptColNames = NPI,Entity Type Code,Replacement NPI,Employer Identification Number (EIN)`
    - This is a comma-delimited list of columns upon which statistics are to be calculated. An empty ACCEPT list will signal the program to calculate statistics on ALL of each row's columns.
- `ignoreColNames = Entity Type Code`
    - This is a comma-delimited list of the columns within the `acceptColNames` setting **for which statistics are *NOT* to be calculated.** An empty IGNORE list will signal the program to calculate statistics on ALL of each the columns within the `acceptColNames` setting.
- `uniqueColNames = NPI`
    - This is a comma-delimited list of the columns within the `acceptColNames` setting **for which value frequency statistics are *NOT* to be calculated.** An empty UNIQUE list will signal the program to calculate statistics on ALL of each the columns within the `acceptColNames` setting (other than those omitted via the `ignoreColNames` setting).

[logging]

- `logFileName = PyDqsStatsGen.log`

    - This setting is used to specify the name of the log file for the PyDqsStatsGen program. If it is just a file name, then the log will be found under the sub-folder named in the `logSubFolder` setting. If it is a full path, then the log file will be written to that full path.

- `maxStdOutLvl = INFO`

    - This setting is used to control the level of messages that appear in the program's STDOUT console. Valid values for this setting are below, but for the most part it should be left as `INFO`.
        - `INFO`
        - `DEBUG`
        - `WARNING`
        - `ERROR`
        - `CRITICAL`

[jdbcSpecs]

- These specifications are used to provide values needed by the program to output DQS statistics to database tables. The `jdbcDropTableIfExistsCompliant` setting controls whether a `DROP TABLE IF EXISTS ...` syntax is used to drop a database table. This syntax works for SQLite, MySQL, and PostgreSQL, but does NOT work for SQL Server, as another SQL command is used to drop tables.
- Valid values for the `jdbcType` setting are as follows:

    - `pgsql` (used for connecting to a PostgreSQL database)
    - `mysql` (used for connecting to a MySQL database)
    - `mssql` (used for connecting to a SQL Server database)

```
# PostgreSQL parameters
jdbcType = pgsql
jdbcHost = localhost
jdbcPort = 5432
jdbcDatabase = dqsvalidator
jdbcUID = dqsvalidator
jdbcPWD = [redacted]
jdbcDropTableIfExistsCompliant = True

# # MySQL parameters
# jdbcType = mysql
# jdbcHost = localhost
# jdbcPort = 3306
# jdbcDatabase = dqsvalidator
# jdbcUID = dqsvalidator
# jdbcPWD = [redacted]
# jdbcDropTableIfExistsCompliant = True

# # SQL Server parameters
# jdbcType = mssql
# jdbcHost = Khepry-ASUS-LT1
# jdbcPort = 1433
# jdbcDatabase = dqsvalidator
# jdbcUID = dqsvalidator
# jdbcPWD = [redacted]
# jdbcDropTableIfExistsCompliant = False
```

Program Prerequisites and Dependencies

- **Linux**
  - python 3.4
    - libraries
      - python3
      - python3-all
      - python3-dev
      - python3-mako
      - python3-mysql.connector
      - python3-pip
      - python3-psycopg2
      - python-pysqlite2
      - sqlite3
      - `sudo pip3 install pymssql`
    - connectors
      - ***NOTE: add freetds.conf file to the {home} folder of the user***
        - [global]
          - tds version = auto
      - freetds-bin
      - freetds-common
      - freetds-dev
      - libdbd-freetds
      - tdsodbc
      - unixodbc
      - unixodbc-dev
- **Windows**
  - Python 3.4
    - Python 3.4 Windows 64-bit Graphical Installer
    - Once the graphical installer is finished, open a command window with "Run as Administrator", and then execute the following commands:
      - `c:\Anaconda3\Scripts\conda.exe update conda`
      - `c:\Anaconda3\Scripts\conda.exe update pip`
      - `pip install mako`
      - `pip install python-psycopg`
    - MySQL Connector/Python 3.4 Installer
    - MySQL Connector/Python (Platform Independent)
      - Download the ZIP archive from dev.mysql.com and install from the command line.
      - (Choose 'Platform Independent' from the drop-down box. That will show you the ZIP and TAR archive).
      - Place unzipped folder within the Anaconda Python directory.
      - Using command prompt, go into the directory of the mysql connector:
        - Shell path> `python setup.py install`
    - SQL Server (pymssql 3.4)
      - pick the 64-bit version of pymssql