Peaks Consolidation

Noticed after This Pre-release

Currently source code of Peaks Framework has been uploaded into this repository. Please see main.go for further information.

System Requirements

- ✓ 64-bit Ubuntu 20.04 or above /Windows or above for x86 CPU
- ✓ Minimum 6GB free memory space
- ✓ Minimum 10GB free hard disk space (Recommend SSD Hard Disk)
- ✓ Any CSV viewer app
- ✓ You can consider to use below old csv viewer
  https://github.com/do-account/PeaksDataFrameViewer

How to Run Script

In command line of current folder e.g. D:\Peaks230518, type

do ScriptFileName (please note that Linux file name is case sensitive)
if file name has white space, do “script file name”

It is no need to type “.txt”

If it is not function, please type ./do

If you run in Linux or Visual Studio Code, please type ./do

In Linux, you may need to execute authorization for do and RunAllScipt.sh.
chmod +x do
chmod +x RunAllScripts.sh

If you run script in other folder, please also copy the runtime in the target folder or set relevant environment path in your operating system.

Peaks Consolidation

License Agreement

The Trial Software is licensed to the End User free of charge during the Trial Period (from the first day you use to August 31, 2023). The following terms and conditions apply exclusively to the Trial Software:

1. **Proprietary Rights:** The Software is owned by Licensor and is protected by copyright laws and international treaty provisions. You will not remove any copyright notices from the Software.
2. **Termination:** This Agreement will terminate automatically if you fail to comply with any of the terms and conditions hereof.
3. **Disclaimer of Warranties:** The Software is provided “AS IS” without warranty of any kind.
4. **Limitation of Liability:** In no event shall Licensor be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this product.

Peaks Consolidation

Delivery

This delivery does not include the source code of the Peaks Framework and Peaks DataFrame. Source code of Peaks Framework will be uploaded to GitHub at a later stage. However, the implementation of ETL expression as described in the following scripts by other software developers is open source.

The Peaks DataFrame is a high-performance calculation engine that can be configured by the framework in either streaming mode or in-memory mode easily. It is not a good time to consider whether it will go open source. Currently, it is kept as proprietary software.

Below are files and folders included in this delivery:-

Peaks230518 (YYMMDD, it is the build date of the runtime)

The following logs indicate that the system passed a billion-row testing using 8-core 32GB memory.

The current setting of the billion-row test is using the system default value "CurrentSetting{StreamMB(500)Thread(100)}". You can increase or decrease it to fit for your machine

- ✓ BillionRowsTestingLog
  - ✓ DifferentCommandsStreaming.log
  - ✓ Distinct.log
  - ✓ FilterByDifferentCompareOperators.log
  - ✓ GroupBy.log
  - ✓ GroupByOneColumn.log
  - ✓ JoinTableFullStreaming.log
  - ✓ SelectColumnRow.log
  - ✓ SelectRow.log
- ✓ PolarsPeaksBenchmark (Page 11)
  - ✓ PeaksScrpt
  - ✓ PolarsScript
    - ✓ PeaksFilterByDifferentCompareOperators.log
    - ✓ PolarsFilterByDifferentCompareOperators.log
- ✓ Input
  - ✓ 0.01MillionRows.csv
  - ✓ 0.1MillionRows.csv
  - ✓ 1MillionRows.csv

- ✓ Master.csv
- ✓ Output

The following files are included in the root folder of “Peaks230518”

- ✓ do.exe for Windows 10/11 64-bit x86 CPU
- ✓ do for Ubuntu 64-bit x86 CPU
- ✓ Readme.doc (this document)
- ✓ ScriptFile.txt (see below)

Below is script files included in root folder of “Peaks230518”:-

Script file must have file extension “txt.

- ✓ DifferentCommands.txt (See page 7,8)
- ✓ DifferentCommandsStreaming.txt
- ✓ Distinct.txt
- ✓ FilterByDifferentCompareOperators.txt
- ✓ GroupBy.txt
- ✓ GroupByOneColumn.txt
- ✓ JoinTable.txt (See page 7)
- ✓ JoinTableFullStreaming.txt
- ✓ SelectColumn.txt
- ✓ SelectColumnRow.txt (See next page)
- ✓ SelectRow.txt
- ✓ SelectUnmatch.txt
- ✓ RunAllScripts.bat (For Windows only: to run all of above script)
- ✓ RunAllScripts.sh (For Linux only: to run all of above script)

The script file above explains how to use the following commands:-

- ✓ BuildKey2Value (not support streaming)
- ✓ Distinct
- ✓ GroupBy
- ✓ JoinKey2Balue
- ✓ ReadFile (not support streaming)
- ✓ Select
- ✓ SelectUnmatch
- ✓ WriteFile (not support streaming)

Github: https://github.com/hkpeaks/peaks-consolidation

Peaks Consolidation

SelectColumnRow.txt

Select Table from Column where Column(Condition) save as Table/File

Select{TableName/FileName | ColumnName1, ColumnName2, ColumnName3 =>
Column1(Condition1 Or Condition2)& Column2(Condition1 Or Condition2) ~
ReturnTable/FileName.csv}

If not declare column name, will output all columns from selected rows

e.g. Select{1MillionRows.csv |
Ledger(L10..L15,L50..L55,L82..L88)Account(12222..12888,15555..16888) ~ Table}

Streaming: If not declare Column(Condition), will output all rows from selected all rows

e.g. Select{1MillionRows.csv | ColumnName1, ColumnName2, ColumnName3 ~ Table}

Full Streaming: If output data greater than memory size

e.g. Select{1MillionRows.csv |
Ledger(L10..L15,L50..L55,L82..L88)Account(12222..12888,15555..16888) ~ BigTable.csv}

e.g. Select{D:\Peaks230518\Output\BigTable.csv | Ledger, Account, PartNo, Project,
Contact, Unit Code, D/C, Currency ~ DistinctBigTable.csv}

Select{1MillionRows.csv | Account, Quantity, Base Amount =>
Ledger(L10..L15,L50..L55,L82..L88)Account(12222..12888,15555..16888) ~ Table}

WriteFile{Table ~ SelectRow.csv}

Peaks Consolidation

The following command is not included in the script file above:-

- ✓ CurrentSetting

This command allows you to configure up to 3 parameters:

- ✓ StreamMB(): represents the size of each batch of stream in MegaByte. The default value is 500, and the maximum value you can figure is 10,000.
- ✓ Thread(): represents the number of partitions run in parallel for a batch of stream. The default value is 10 when file size >= 100,000,000 bytes; otherwise, it is 1. The maximum thread you can configure is 1000. If the file size is less than 100,000,000 bytes, it is not recommended to use more than 1 thread; otherwise, potential errors will occur.
- ✓ ForceInteger(): is designed for working with GroupBy. The system will get an integer value of the number.
- ✓ For example: CurrentSetting{StreamMB(1000)Thread(100)ForceInteger(True)}

The following script is not included in the folder “Peaks230518”:-You should ensure that you have enough hard disk space before expanding the file.

“%ExpandBy100Time.csv” is case sensitive. Maximum is 100, minimum is 1.

ReadFile{1MillionRows.csv ~ Table}

WriteFile{Table ~ %ExpandBy100Time.csv}

Peaks Consolidation

Example of Execution

```
D:\Peaks230518>do JoinTable
Development runtime for testing only
Build Date: 23-05-18 | Expiry Date: 23-08-31
Report Comment: github.com/hkpeaks/peaks-consolidation

ReadFile{Master.csv ~ Master}
   Master(5 x 99213)

BuildKeyValue{Master | Ledger,Account,Project ~ KeyValue}
   KeyValue(3 x 99213)

JoinKeyValue{1MillionRows.csv | Ledger,Account,Project => AllMatch(KeyValue)~ Table}
   Table(16 x 1000000)

WriteFile{Table ~ PeaksJoinTable.csv}
   PeaksJoinTable.csv(16 x 1000000)

Duration: 0.87 second
```

```
D:\Peaks230518>do DifferentCommands
Development runtime for testing only
Build Date: 23-05-18 | Expiry Date: 23-08-31
Report Comment: github.com/hkpeaks/peaks-consolidation

ReadFile{1MillionRows.csv ~ Table}
   Table(14 x 1000000)

Select{Project(>B28,<B22)Ledger(L10..L12,L53..L55,L82..L85)D/C(=D)}
   Table(14 x 20190)

Select{Contact(=C39,C32..C34)}
   Table(14 x 7250)

Select{Quantity(Float500..600)Base Amount(Float>30000)~ FilteredTable}
```

Peaks Consolidation

FilteredTable(14 x 98)

ReadFile{Master.csv ~ MasterTable}
MasterTable(5 x 99213)

BuildKeyValue{MasterTable | Ledger,Account,Project ~ KeyValue}
KeyValue(3 x 99213)

JoinKeyValue{FilteredTable | Ledger,Account,Project => AllMatch(KeyValue)~ JoinedTable}
JoinedTable(16 x 98)

GroupBy{JoinedTable | Cost Centre => Count()Max(Quantity)Min(Quantity)Min(Unit Price)Max(Unit Price)Sum(Base Amount)~ Cost Centre}
Cost Centre(7 x 66)

GroupBy{JoinedTable | Company => Count()Max(Exchange Rate)Min(Exchange Rate)Max(Unit Price)Min(Unit Price)Sum(Base Amount)~ Company}
Company(7 x 34)

WriteFile{FilteredTable ~ PeaksManyCommandsFilterData.csv}
PeaksManyCommandsFilterData.csv(7 x 98)

WriteFile{Cost Centre ~ PeaksManyCommandsGroupByCostCentre.csv}
PeaksManyCommandsGroupByCostCentre.csv(7 x 66)

WriteFile{Company ~ PeaksManyCommandsGroupByCompany.csv}
PeaksManyCommandsGroupByCompany.csv(7 x 34)

Duration: 0.613 second

Peaks Consolidation

Frequently Occur Error Message

- ✓ ** Command WriteFil not found **
- ✓ ** File 1MILLIONROW.CSV not found **
- ✓ ** Column Ledge not found **
- ✓ ** Table Tabl not found **
- ✓ Incorrect rule setting

```
Original Rule  :  Distinct{1MillionRows.csv|Ledger,Account,PartNo,Project
,Contact,UnitCode,D/C,Currency~TableDistinct{Table|Ledger~DistinctLedger}
Recognized Rule:  Distinct

** Fail to recognize this rule **
** Usually due to incomplete () or {} **
```

The software has been in development for three months and is still far from completion for production use. Since the exception handling system has not yet covered all possible errors due to improper setting of rules, you may encounter unreadable error messages. However, these error messages can be useful for developers for debugging purposes.

```
GroupBy{1MillionRows.csv | Ledger, Account, PartNo,Project,Contact,Unit C
ode, D/C,Currency > Sum(Quantity)Max(Quantity)Min(Quantity)Sum(Original A
mount)Sum(Base Amount)~ Table}
panic: runtime error: index out of range [-5]

goroutine 14 [running]:
do/peaks.CurrentDistinctGroupByFloat64(_, _, _, {{0xc00000e751, 0x5}, {0x
c000017380, 0x10}, {0xc00000e751, 0x5}, {0x0, ...}, ...}, ...)
        D:/Go/Peaks DataFrame/peaks/data_bending.go:909 +0x19c6
do/peaks.CurrentCommand(_, _, _, _, {{0x110425c, 0x4}, {0xc00000e640, 0x9
}, {0x0, 0x0}}, ...)
        D:/Go/Peaks DataFrame/peaks/controller.go:881 +0x6f0
do/peaks.ParallelDistinct.func1(0x0)
        D:/Go/Peaks DataFrame/peaks/controller.go:816 +0x15a
created by do/peaks.ParallelDistinct
        D:/Go/Peaks DataFrame/peaks/controller.go:814 +0x34b
```

Incorrect

GroupBy{1MillionRows.csv | Ledger, Account, PartNo,Project,Contact,Unit Code, D/C,Currency > Sum(Quantity) Max(Quantity) Min(Quantity) Sum(Original Amount) Sum(Base Amount) ~ Table}

It shall be =>

Github: https://github.com/hkpeaks/peaks-consolidation

Peaks Consolidation

Coming Development Versions

During the active development phase, new versions are typically published every month which include new functions and bug fixes. The following possible functions are considered to develop and include in future versions:-

- ✓ Read files from folders or based on a file list
- ✓ Parquet and JSON file format
- ✓ Append data by value and/or formula
- ✓ Split file by unique value of particular columns
- ✓ Sorting for billion rows by splitting file method
- ✓ Date format handling
- ✓ CrossTab
- ✓ Other Join table methods
- ✓ Conditional action
- ✓ Conditional mapping with effective date
- ✓ Accounting period
- ✓ Accounting entry automation
- ✓ Calculate multi-dimensional account balance (Periodical Balance covers daily, monthly and yearly)
- ✓ Amend table by Key
- ✓ Data reconciliation
- ✓ Regular expression
- ✓ In-memory DB by gRPC (requires two runtimes)
- ✓ Peaks Web GUI similar to https://youtu.be/6hwbQmTXzMc

You can see latest videos about the latest development of the software.
https://www.youtube.com/@accountingetl5158/videos

Peaks Consolidation

Benchmarking

The development of Peaks is motivated by this benchmarking https://h2oai.github.io/db-benchmark/. In Feb 2023 the author first time see this benchmark.

Revelent testing scripts and logs, see folder “Peaks230518\PolarsPeaksBenchmark”

Rows	Polars Ver 0.17.11	Peaks Ver 23-05-18
0.01 Million	0.078s	0.031s
0.1 Million	0.063s	0.071s
1 Million	0.375s	0.518s
10 Million	3.659s	1.544s
100 Million	41.472s	9.702s
1000 Million	717.149s	189.648s

The author noticed that Polars, R Data.Table, DuckDB and Apache Spark won for a particular benchmark test. Below are recommended settings you can use to benchmark these software.

Setting for Benchmark 1: JoinTable Focus

ReadFile{Master.csv ~ Master}
BuildKeyValue{Master | Ledger,Account,Project ~ KeyValue}

If the exported data is larger than the memory capacity, you can implement full streaming mode to write
the file by each stream.

JoinKeyValue{1000MillionRows.csv | Ledger,Account,Project => AllMatch(KeyValue) ~ PeaksJoinTableFullStreaming.csv}

Peaks Consolidation

Setting for Benchmark 2: Filter Focus

```
Select{1000MillionRows.csv | Ledger(L=99,<L20)Project(>B25,<B23) ~ Table}
Select{Currency(!=C06)}
Select{Account(<=11000, >=18000)}

## By default, all columns are deemed as text type. For filtering purposes, please specify float type for real numbers.
## This filter by float must be defined in a separate line.

Select{Quantity(Float100..300,Float600..900)}
Select{Contact(C32..C39)}
Select{Contact(!=C33) ~ Table2}

WriteFile{Table2 ~ PeaksFilterByDifferentCompareOperators.csv}

Select{Table2 | Ledger(=L10) ~ PeaksFilterByDifferentCompareOperatorsExistL10.csv}
Select{Table2 | Account(18000..19000) ~ PeaksFilterByDifferentCompareOperatorsExistAccountRange.csv}
Select{Table2 | Contact(=C33) ~ PeaksFilterByDifferentCompareOperatorsNothingC33.csv}
Select{Table2 | Account(12000..13000) ~ PeaksFilterByDifferentCompareOperatorsNothingAccountRange.csv}
```

Peaks Consolidation

Setting for Benchmark 3: Sundries

```
## You can configure path of input and output file
ReadFile{1000MillionRows.csv ~ Table}

## Filter Data From One Million Rows CSV File

Select{Project(>B28,<B22)Ledger(L10..L12,L53..L55,L82..L85)D/C(=D)}
Select{Contact(=C39,C32..C34)}

## By default, all columns are deemed as text type. For filtering purposes, please specify
float type for real numbers.
## This filter by float must be defined in a separate line.

Select{Quantity(Float500..600)Base Amount(Float>30000) ~ FilteredTable}

## Join "FilteredTable" from "MasterTable" to Append New Columns "Company" & "Cost
Centre"

ReadFile{Master.csv ~ MasterTable}
BuildKeyValue{MasterTable | Ledger,Account,Project ~ KeyValue}
JoinKeyValue{FilteredTable | Ledger,Account,Project => AllMatch(KeyValue) ~ JoinedTable}

## Summarized Data Based on New Columns "Company" & "Cost Centre"

GroupBy{JoinedTable | Cost Centre =>    Count() Max(Quantity) Min(Quantity) Min(Unit
Price) Max(Unit Price) Sum(Base Amount) ~ Cost Centre}
GroupBy{JoinedTable | Company =>    Count() Max(Exchange Rate) Min(Exchange Rate)
Max(Unit Price) Min(Unit Price) Sum(Base Amount) ~ Company}

## Output In-memory Table Data to CSV File

WriteFile{FilteredTable ~ PeaksManyCommandsFilterData.csv}
WriteFile{Cost Centre ~ PeaksManyCommandsGroupByCostCentre.csv}
WriteFile{Company ~ PeaksManyCommandsGroupByCompany.csv}
```