In examples below, drafted a couple R utility functions with the MonetDBLite back-end by mimicking the sqldf function. There are several interesting observations shown in the benchmark comparison.  
– The data import for csv data files is more efficient with MonetDBLite than with the generic read.csv function or read.csv.sql function in the sqldf package.  
– The data manipulation for a single data frame, such as selection, aggregation, and subquery, is also significantly faster with MonetDBLite than with the sqldf function.  
– However, the sqldf function is extremely efficient in joining 2 data frames, e.g. inner join in the example.

# IMPORT

monet.read.csv <- function(file) {

monet.con <- DBI::dbConnect(MonetDBLite::MonetDBLite(), ":memory:")

suppressMessages(MonetDBLite::monetdb.read.csv(monet.con, file, "file", sep = ","))

result <- DBI::dbReadTable(monet.con, "file")

DBI::dbDisconnect(monet.con, shutdown = T)

return(result)

}

microbenchmark::microbenchmark(monet = {df <- monet.read.csv("Downloads/nycflights.csv")}, times = 10)

#Unit: milliseconds

# expr min lq mean median uq max neval

# monet 528.5378 532.5463 539.2877 539.0902 542.4301 559.1191 10

microbenchmark::microbenchmark(read.csv = {df <- read.csv("Downloads/nycflights.csv")}, times = 10)

#Unit: seconds

# expr min lq mean median uq max neval

# read.csv 2.310238 2.338134 2.360688 2.343313 2.373913 2.444814 10

# SELECTION AND AGGREGATION

monet.sql <- function(df, sql) {

df\_str <- deparse(substitute(df))

monet.con <- DBI::dbConnect(MonetDBLite::MonetDBLite(), ":memory:")

suppressMessages(DBI::dbWriteTable(monet.con, df\_str, df, overwrite = T))

result <- DBI::dbGetQuery(monet.con, sql)

DBI::dbDisconnect(monet.con, shutdown = T)

return(result)

}

microbenchmark::microbenchmark(monet = {monet.sql(df, "select \* from df sample 3")}, times = 10)

#Unit: milliseconds

# expr min lq mean median uq max neval

# monet 422.761 429.428 439.0438 438.3503 447.3286 453.104 10

microbenchmark::microbenchmark(sqldf = {sqldf::sqldf("select \* from df order by RANDOM() limit 3")}, times = 10)

#Unit: milliseconds

# expr min lq mean median uq max neval

# sqldf 903.9982 908.256 925.4255 920.2692 930.0934 963.6983 10

microbenchmark::microbenchmark(monet = {monet.sql(df, "select origin, median(distance) as med\_dist from df group by origin")}, times = 10)

#Unit: milliseconds

# expr min lq mean median uq max neval

# monet 450.7862 456.9589 458.6389 458.9634 460.4402 465.2253 10

microbenchmark::microbenchmark(sqldf = {sqldf::sqldf("select origin, median(distance) as med\_dist from df group by origin")}, times = 10)

#Unit: milliseconds

# expr min lq mean median uq max neval

# sqldf 833.1494 836.6816 841.952 843.5569 846.8117 851.0771 10

microbenchmark::microbenchmark(monet = {monet.sql(df, "with df1 as (select dest, avg(distance) as dist from df group by dest), df2 as (select dest, count(\*) as cnts from df group by dest) select \* from df1 inner join df2 on (df1.dest = df2.dest)")}, times = 10)

#Unit: milliseconds

# expr min lq mean median uq max neval

# monet 426.0248 431.2086 437.634 438.4718 442.8799 451.275 10

microbenchmark::microbenchmark(sqldf = {sqldf::sqldf("select \* from (select dest, avg(distance) as dist from df group by dest) df1 inner join (select dest, count(\*) as cnts from df group by dest) df2 on (df1.dest = df2.dest)")}, times = 10)

#Unit: seconds

# expr min lq mean median uq max neval

# sqldf 1.013116 1.017248 1.024117 1.021555 1.025668 1.048133 10

# MERGE

monet.sql2 <- function(df1, df2, sql) {

df1\_str <- deparse(substitute(df1))

df2\_str <- deparse(substitute(df2))

monet.con <- DBI::dbConnect(MonetDBLite::MonetDBLite(), ":memory:")

suppressMessages(DBI::dbWriteTable(monet.con, df1\_str, df1, overwrite = T))

suppressMessages(DBI::dbWriteTable(monet.con, df2\_str, df2, overwrite = T))

result <- DBI::dbGetQuery(monet.con, sql)

DBI::dbDisconnect(monet.con, shutdown = T)

return(result)

}

tbl1 <- monet.sql(df, "select dest, avg(distance) as dist from df group by dest")

tbl2 <- monet.sql(df, "select dest, count(\*) as cnts from df group by dest")

microbenchmark::microbenchmark(monet = {monet.sql2(tbl1, tbl2, "select \* from tbl1 inner join tbl2 on (tbl1.dest = tbl2.dest)")}, times = 10)

#Unit: milliseconds

# expr min lq mean median uq max neval

# monet 93.94973 174.2211 170.7771 178.487 182.4724 187.3155 10

microbenchmark::microbenchmark(sqldf = {sqldf::sqldf("select \* from tbl1 inner join tbl2 on (tbl1.dest = tbl2.dest)")}, times = 10)

#Unit: milliseconds

# expr min lq mean median uq max neval

# sqldf 19.49334 19.60981 20.29535 20.001 20.93383 21.51837 10