**Loading Data Frame to Relational Database with R**

Once you create a [data frame](http://www.r-tutor.com/r-introduction/data-frame) with [R](https://www.r-project.org/), you may need to load it to a relational database for data persistence. You might have a data transformation batch job written in R and want to load database in a certain frequency.

Here, I created a function to load data into a relational database. I opted to use [RODBC](https://www.rdocumentation.org/packages/RODBC/versions/1.3-15) because it is probably the easiest way to interact with databases with R. As long as you install and configure ODBC for whatever database you are using, this should work.

RODBC has many functions that does database operations for you. The insertion is taken care of by the [sqlSave](https://www.rdocumentation.org/packages/RODBC/versions/1.3-15/topics/sqlSave) method. [RODBC](https://www.rdocumentation.org/packages/RODBC/versions/1.3-15) has fantastic documentation so that you can check what options are available.

**Function Parameters**

The function below takes 6 parameters.

**dsn**: data source name that you configured for ODBC.

**user**: database user name.

**pw**: database password.

**tableName**: Name of the table including the schema

**df**: R data frame

**columnTypes**: List of SQL column types.

**Usage**

Call the function like this

[cc lang="rsplus" tab\_size="4" lines="-1"]

dsn = "my-database-dsn"

dbUser = "user"

dbPass = "password"

tableName = "datamart.customer\_dim"

df = dataFrameCreated

columnTypes <- list(account\_name="varchar(255)", customer\_id="int", last\_updated="date")

dbLoader(dsn, dbUser, dbPass, "usermanaged.session\_budget\_2018", dfTransformed, columnTypes)

[/cc]

**Function: dbLoader**

I could probably make drop table SQL statement as a parameter so that it can be used for any database. In this function, it is hard coded. The statement should work for most databases. Most of databases have the same drop statement syntax.

The db load pattern is the classic truncate and load. You can customise it to make it to upsert.

[cc lang="rsplus" tab\_size="4" lines="-1"]

dbLoader <- function(dsn, user, pw, tableName, df, columnTypes) {

channel <- odbcConnect(dsn, uid=user, pwd=pw)

print('Database connection initiated.')

tableName <- tableName

dropSQL <- sprintf('Drop Table If Exists %s;', tableName)

# Drop Table If exists

sqlQuery(channel, dropSQL)

print(sprintf('Executed %s', dropSQL))

# Insert Data

sqlSave(channel, df, tablename=tableName, fast=T, colnames=F, rownames = F, varTypes=columnTypes)

print(sprintf('DataFrame has been inserted into %s', tableName ))

# Close DB connection

close(channel)

print('DB connection closed.')

}

[/cc]

Easy!