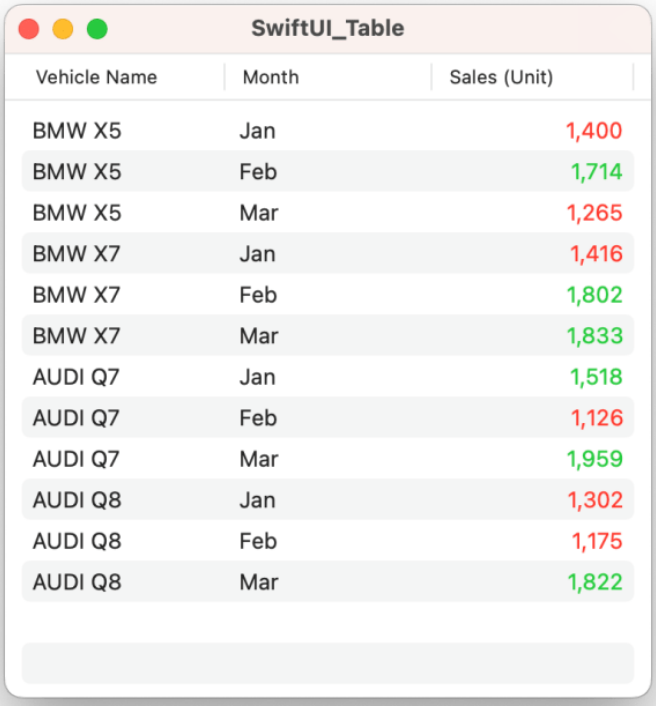
SwiftUI Table and SwiftUI TableColumn

ddas.tech/swiftui-table-and-swiftui-tablecolumn/

August 4, 2023

In this post we will create a couple of sample applications to see usage of **SwiftUI Table and SwiftUI TableColumn**



Lets start with defining the model

```
struct ProductSalesRecord: Identifiable, Codable{
    var prodName: String
    var month: String
    var unitSales: Int
    var id =  UUID()
    init(prodName: String, month: String, unitSales: Int) {
        self.prodName = prodName
        self.month = month
        self.unitSales = unitSales
    }
}
```

In the below sample we have created salesData which is an array of ProductSalesRecord model

```
//  ContentView.swift
//  SwiftUI_Table
//  Created by Debasis Das on 8/2/23.

import SwiftUI

struct ContentView: View {
    @State var salesData: [ProductSalesRecord] = {
            var data:[ProductSalesRecord] = []
            let prodNames = ["BMW X5","BMW X7","AUDI Q7","AUDI Q8"]
            let upperBound = 2000
            let lowerBound = 1000
            for prodName in prodNames {
                for month in ["Jan","Feb","Mar"]{
                    let rec = ProductSalesRecord(prodName: prodName, month: month,
unitSales: Int(arc4random_uniform(UInt32(upperBound - lowerBound))) + lowerBound)
                    data.append(rec)
                }
            }
            return data
        }()

    var body: some View {
        Table(salesData) {
            TableColumn("Vehicle Name", value: \.prodName)
                .width(min: 100,max: 400)
            TableColumn("Month", value: \.month)
            TableColumn("Sales (Unit)"){ sd in
                Text("\(sd.unitSales)")
                    .frame(minWidth: 0, maxWidth: .infinity,alignment:.trailing)
                    .foregroundColor(sd.unitSales > 1500 ? .green : . red)
            }
        }
    }
}
```

Lets try another example of SwiftUI Table

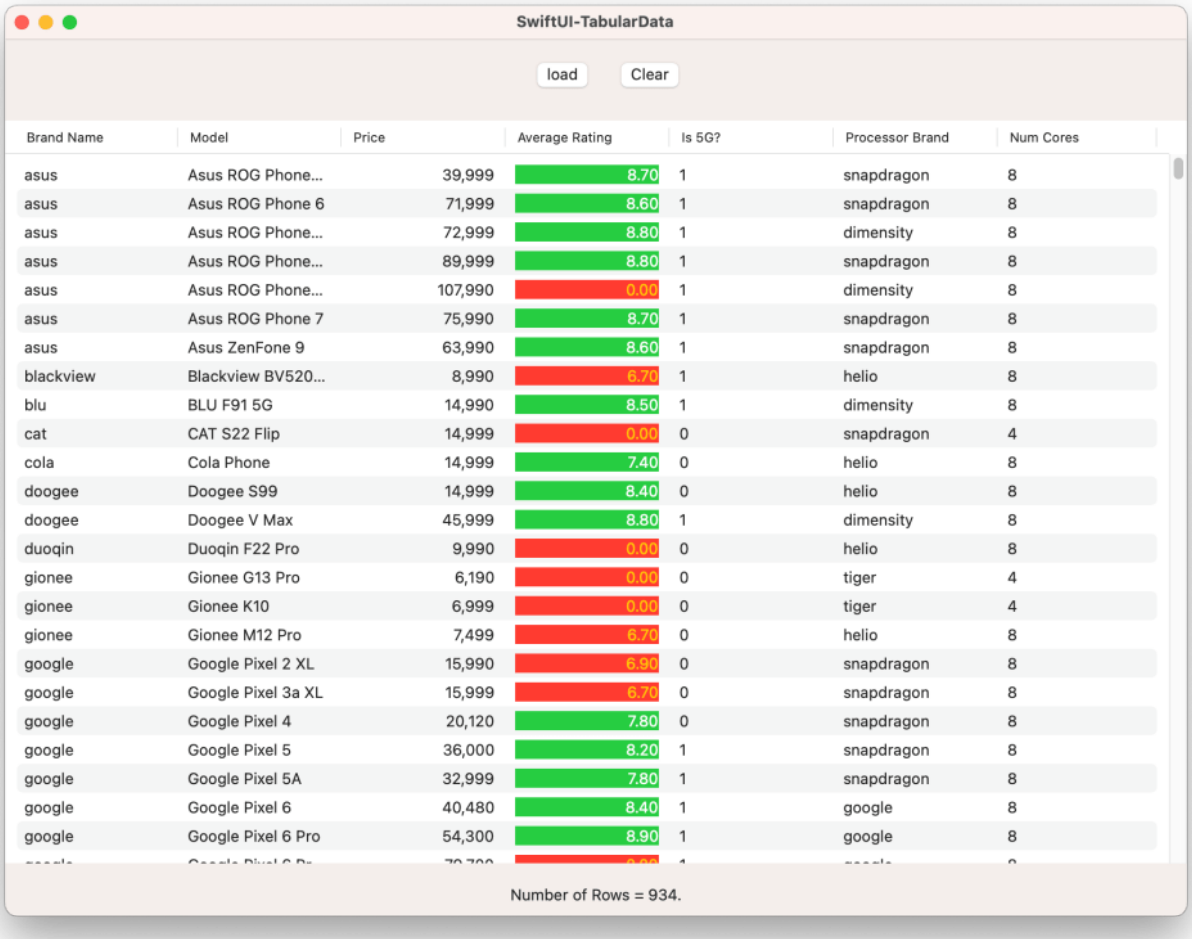
Now we will try to load data from a csv using Swift TabularData and then map the same to a swift data structure and create a SwiftUI table

https://www.kaggle.com/datasets/abhijitdahatonde/real-world-smartphones-dataset

```
┌──────┬──────────┬──────────────────────────┬──────────┬──────────┬──────────┬──────────────┬──────────┬
│      │ brand_name │ model                    │ price    │ avg_rating │ 5G_or_not │ processor_brand │ num_cores │
│      processor_speed │ battery_capacity │ fast_charging_available │ 12  ⋮
│      │ <String>   │ <String>                 │ <Int>    │ <Double>   │ <Int>     │ <String>        │ <Int>     │
│      <Double>      │ <Int>            │ <Int>                │          │ more ⋮    │           │                 │           │
├──────┼──────────┼──────────────────────────┼──────────┼──────────┼──────────┼──────────────┼──────────┼
│      │          │                          │          │          │           │                 │           │
│ 0    │ asus     │ Asus ROG Phone 5s 5G     │ 39,999   │      8.7 │         1 │ snapdragon      │         8 │
│      2.9 │ 6,000 │                          │ 1 │ ⋮
│ 1    │ asus     │ Asus ROG Phone 6         │ 71,999   │      8.6 │         1 │ snapdragon      │         8 │
│      3.2 │ 6,000 │                          │ 1 │ ⋮
│ 2    │ asus     │ Asus ROG Phone 6 Batman Edition │ 72,999 │      8.8 │         1 │ dimensity       │         8 │
│      3.2 │ 6,000 │                          │ 1 │ ⋮
│ 3    │ asus     │ Asus ROG Phone 6 Pro 5G  │ 89,999   │      8.8 │         1 │ snapdragon      │         8 │
│      3.2 │ 6,000 │                          │ 1 │ ⋮
│ 4    │ asus     │ Asus ROG Phone 6D Ultimate │ 107,990 │      nil │         1 │ dimensity       │         8 │
│      3.2 │ 6,000 │                          │ 1 │ ⋮
```

The above dataset has the following columns

```
["brand_name", "model", "price", "avg_rating", "5G_or_not", "processor_brand",
"num_cores", "processor_speed", "battery_capacity", "fast_charging_available",
"fast_charging", "ram_capacity", "internal_memory", "screen_size", "refresh_rate",
"num_rear_cameras", "os", "primary_camera_rear", "primary_camera_front",
"extended_memory_available", "resolution_height", "resolution_width"]
```



Brand Name	Model	Price	Average Rating	Is 5G?	Processor Brand	Num Cores
asus	Asus ROG Phone...	39,999	8.70	1	snapdragon	8
asus	Asus ROG Phone 6	71,999	8.60	1	snapdragon	8
asus	Asus ROG Phone...	72,999	8.80	1	dimensity	8
asus	Asus ROG Phone...	89,999	8.80	1	snapdragon	8
asus	Asus ROG Phone...	107,990	0.00	1	dimensity	8
asus	Asus ROG Phone 7	75,990	8.70	1	snapdragon	8
asus	Asus ZenFone 9	63,990	8.60	1	snapdragon	8
blackview	Blackview BV520...	8,990	6.70	1	helio	8
blu	BLU F91 5G	14,990	8.50	1	dimensity	8
cat	CAT S22 Flip	14,999	0.00	0	snapdragon	4
cola	Cola Phone	14,999	7.40	0	helio	8
doogee	Doogee S99	14,999	8.40	0	helio	8
doogee	Doogee V Max	45,999	8.80	1	dimensity	8
duoqin	Duoqin F22 Pro	9,990	0.00	0	helio	8
gionee	Gionee G13 Pro	6,190	0.00	0	tiger	4
gionee	Gionee K10	6,999	0.00	0	tiger	4
gionee	Gionee M12 Pro	7,499	6.70	0	helio	8
google	Google Pixel 2 XL	15,990	6.90	0	snapdragon	8
google	Google Pixel 3a XL	15,999	6.70	0	snapdragon	8
google	Google Pixel 4	20,120	7.80	0	snapdragon	8
google	Google Pixel 5	36,000	8.20	1	snapdragon	8
google	Google Pixel 5A	32,999	7.80	1	snapdragon	8
google	Google Pixel 6	40,480	8.40	1	google	8
google	Google Pixel 6 Pro	54,300	8.90	1	google	8

Let's start with defining the model. Although the data set has 22 columns we will only map the first few columns for this demo app

```
struct SmartPhoneModel: Identifiable{
    var brandName:String
    var model: String
    var price: Int
    var averageRating: Double
    var is5G:Int
    var processorBrand: String
    var numCores: Int
    var id = UUID()

    init(brandName:String, model:String , price:Int, averageRating: Double, is5G :
Int, processorBrand: String, numCores: Int) {
        self.brandName = brandName
        self.model = model
        self.price = price
        self.averageRating = averageRating
        self.is5G = is5G
        self.processorBrand = processorBrand
        self.numCores = numCores
    }
}
```

In the below code we will use TabularData framework to load the data from csv into a DataFrame and then iterate through each row of the DataFrame to map to native swift data structure/model.

Once the data is loaded we will create a SwiftUI Table and render the same

```
//  ContentView.swift
//  SwiftUI-TabularData
//  Created by Debasis Das on 8/2/23.
//

import SwiftUI
import TabularData

struct ContentView: View {
    @State private var isDataLoaded = false
    @State private var tableData: [SmartPhoneModel] = []
    var body: some View {
        HStack{
            Button("load",action:loadDataFromCsv).padding(20)
            Button("Clear"){
                self.tableData = []
            }
        }
        if isDataLoaded {
                    Table(tableData){
                        TableColumn("Brand Name", value: \.brandName)
                        TableColumn("Model", value: \.model)
                        TableColumn("Price"){ sd in
                            Text("\(sd.price)")
                                .frame(minWidth: 0, maxWidth:
.infinity,alignment:.trailing)
                        }
                        TableColumn("Average Rating"){ sd in
                            Text(String(format: "%.2f", sd.averageRating))
                                .frame(minWidth: 0, maxWidth:
.infinity,alignment:.trailing)
                                .background(sd.averageRating > 7.0 ? .green : . red)
                                .foregroundColor(sd.averageRating > 7.0 ? .white : .
yellow)
                        }
                        TableColumn("Is 5G?"){ sd in
                            Text("\(sd.is5G)")
                        }
                        TableColumn("Processor Brand", value: \.processorBrand)
                        TableColumn("Num Cores"){ sd in
                            Text("\(sd.numCores)")
                        }

                    }
                        Text("Number of Rows = \(tableData.count).").padding([.top,
.bottom], 10)
            } else {
                        Text("Click the button above to load the table.")
            }

    }
```

```
    func loadDataFromCsv(){
        DispatchQueue.main.asyncAfter(deadline: .now() + 0.25) {
            var data:[SmartPhoneModel] = []
            if let filePath = Bundle.main.url(forResource: "smartphones",
withExtension: "csv"){
                let options = CSVReadingOptions(hasHeaderRow: true, delimiter: ",")
                guard let fileUrl = URL(string: filePath.absoluteString) else {
                    fatalError("Error creating Url")
                }
                var df = try! DataFrame(
                    contentsOfCSVFile: fileUrl,
                    options: options)
                let rows = df.rows
                for row in rows{
                    let brand_name = row["brand_name"] as? String ?? ""
                    let model = row["model"] as? String ?? ""
                    let price = row["price"] as? Int ?? 0
                    let avg_rating = row["avg_rating"] as? Double ?? 0.0
                    let is5G = row["5G_or_not"] as? Int ?? 0
                    let processor_brand = row["processor_brand"] as? String ?? ""
                    let num_cores = row["num_cores"] as? Int ?? 0
                    let record = SmartPhoneModel(brandName: brand_name, model: model,
price: price, averageRating: avg_rating, is5G: is5G, processorBrand: processor_brand,
numCores: num_cores)
                    data.append(record)
                }
            }
            self.tableData = data
            self.isDataLoaded = true

        }
    }
}
```

You can check other approaches of creating a NSTableView below.

1. Loading a NSTableView in SwiftUI using NSViewRepresentable https://ddas.tech/nstableview-in-swiftui-sample-code/
2. Loading a NSViewController with NSTableView in SwiftUI using NSViewControllerRepresentable https://ddas.tech/nsviewcontroller-in-swiftui/