Contracts

Contracts tell us how to use a function. For example: `num-min :: (a :: Number, b :: Number) -> Number` tells us that the name of the function is `num-min`, it takes two inputs (both Numbers), and it evaluates to a `Number`. From the contract, we know `num-min(4, 6)` will evaluate to a `Number`. **Use the blank line under each contract for notes or sample code for that function!**

Name		Domain		Range
`box-plot`	`::`	`(t :: Table, col :: String)`	`->`	`Image`
`modified-box-plot`	`::`	`(t :: Table, col :: String)`	`->`	`Image`
`scatter-plot`	`::`	`(t :: Table, labels :: String, xs :: String, ys :: String)`	`->`	`Image`
`image-scatter-plot`	`::`	`(t :: Table, xs :: String, ys :: String, f :: (Row -> Image))`	`->`	`Image`
`r-value`	`::`	`(t :: Table, xs :: String, ys :: String)`	`->`	`Number`
`lr-plot`	`::`	`(t :: Table, labels :: String, xs :: String, ys :: String)`	`->`	`Image`
`random-rows`	`::`	`(t :: Table, num-rows :: Number)`	`->`	`Table`
`<Table>.row-n`	`::`	`(n :: Number)`	`->`	`Row`
`<Table>.order-by`	`::`	`(col :: String, increasing :: Boolean)`	`->`	`Table`
`<Table>.filter`	`::`	`(test :: (Row -> Boolean))`	`->`	`Table`
`<Table>.build-column`	`::`	`(col :: String, builder :: (Row -> Any))`	`->`	`Table`
`bar-chart-summarized`	`::`	`(t :: Table, labels :: String, values :: String)`	`->`	`Image`
`pie-chart-summarized`	`::`	`(t :: Table, labels :: String, values :: String)`	`->`	`Image`