Работа с датафреймами на Julia

Создание датафрейма

Для начала скачаем датасеты из R и пакет DataFrames для работы с ними:

```
begin
    using Pkg
    Pkg.add(["DataFrames", "RDatasets"])

    using DataFrames
    using RDatasets
end
```

Сохраняем датасет diamonds как DataFrame:

```
diamonds =
```

	Carat	Cut	Color
1	0.23	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
2	0.21	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "
3	0.23	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
4	0.29	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "
5	0.31	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
6	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
7	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
8	0.26	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
9	0.22	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
10	0.23	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "

more

```
diamonds = dataset("ggplot2", "diamonds")
```

Функции first и last позволяют вывести на экран первые и последние строки датафрейма соответственно:

	Carat	Cut	Color

	Carat	Cut	Color
1	0.23	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "E" (
2	0.21	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "E" (
3	0.23	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "E" (
4	0.29	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "I" (
5	0.31	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "J" (

```
first(diamonds, 5)
```

	Carat	Cut	Color
1	0.72	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "D" (
2	0.72	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "D" (
3	0.7	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "D" (
4	0.86	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "H" (
5	0.75	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "D" (

```
last(diamonds, 5)
```

Выбор отдельных столбцов

По названию стобцы могут быть выбраны одним из следующих способов:

```
Float64[0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.3, 0.23,
```

```
diamonds.Carat
```

```
Float64[0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.3, 0.23,
```

```
diamonds."Carat"
```

```
Float64[0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.3, 0.23,
```

```
diamonds[!, :Carat]
```

```
Float64[0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.3, 0.23,
```

```
diamonds[!, "Carat"]
```

Поскольку df[!, :col] не создает копию, изменение элементов получаемого таким способом вектора приводит к изменению значений в исходном df.

```
Float64[111.0, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.3, 0.23,
```

```
begin
    vec = diamonds[!, "Carat"]
    vec[1] = 111 # изменяем первое значение вектора на 111
    vec
end
```

	Carat	Cut	Color
1	111.0	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "E" (
2	0.21	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "E" (
3	0.23	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "E" (
4	0.29	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "I" (
5	0.31	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "J" (

```
first(diamonds, 5) # значение в df не изменилось на 111
```

Для создания копии столбца надо использовать df[:, :col] , изменение полученного вектора не будет влиять на исходный df:

```
Float64[222.0, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.3, 0.23,
```

```
begin
    vec2 = diamonds[:, "Carat"]
    vec2[1] = 222 # изменяем первое значение вектора на 222
    vec2
end
```

	Carat	Cut	Color
1	111.0	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "E" (
2	0.21	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "E" (
3	0.23	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "E" (
4	0.29	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "I" (
5	0.31	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "J" (

```
first(diamonds, 5) # значение в df не изменилось на 222
```

```
0.23
```

```
diamonds[1, 1] = 0.23 # вернем исходное значение
```

Выбор подмножества датафрейма

Можно выбрать подмножество датафрейма с помощью индексов. Двоеточие : означает, что нужно выбрать все элементы (строки или столбца, в зависимости от позиции).

	Carat	Cut	Color
1	0.21	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "E" (
2	0.23	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "E" (
3	0.29	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "I" (
4	0.31	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "J" (
5	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "J" (
6	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "I" (
7	0.26	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "H" (
8	0.22	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "E" (
9	0.23	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "H" (

```
diamonds[2:10, :]
```

	Carat	Cut	Color
1	0.23	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "E" (
2	0.31	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "J" (
3	0.23	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "H" (

```
diamonds[[1, 5, 10], :]
```

	Carat	Color	Cut
1	0.23	CategoricalValue{String,UInt8} "E" (2/	CategoricalValue{String,UInt8} "Good
2	0.29	CategoricalValue{String,UInt8} "I" (6/	CategoricalValue{String,UInt8} "Prem
3	0.31	CategoricalValue{String,UInt8} "J" (7/	CategoricalValue{String,UInt8} "Good
4	0.24	CategoricalValue{String,UInt8} "J" (7/	CategoricalValue{String,UInt8} "Very

```
diamonds[3:6, ["Carat", "Color", "Cut"]]
```

	Carat	Color	Cut

	Carat	Color	Cut
1	0.23	CategoricalValue{String,UInt8} "H" (5/	CategoricalValue{String,UInt8} "Very
2	0.3	CategoricalValue{String,UInt8} "J" (7/	CategoricalValue{String,UInt8} "Good
3	0.23	CategoricalValue{String,UInt8} "J" (7/	CategoricalValue{String,UInt8} "Idea
4	0.22	CategoricalValue{String,UInt8} "F" (3/	CategoricalValue{String,UInt8} "Prem
5	0.31	CategoricalValue{String,UInt8} "J" (7/	CategoricalValue{String,UInt8} "Idea
6	0.2	CategoricalValue{String,UInt8} "E" (2/	CategoricalValue{String,UInt8} "Prem

```
diamonds[10:15, [:Carat, :Color, :Cut]]
```

Стоит отметить, что df[:, [:col]] и df[!, [:col]] возвращают объект DataFrame, а df[:, :col] и df[!, :col] - вектор:

	Carat
1	0.23
2	0.21
3	0.23
4	0.29
5	0.31
6	0.24
7	0.24
8	0.26
9	0.22
10	0.23
	more
53940	0.75

```
diamonds[:, [:Carat]]
```

```
Float64[0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.3, 0.23,
```

```
diamonds[:, :Carat]
```

Селектор Not позволяет выбрать все столбцы, кроме обозначенного подмножества:

	Cut	Color
1	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "E" (2/ Catego

	Cut	Color
2	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "E" (2/ Catego
3	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "E" (2/ Catego
4	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "I" (6/ Catego
5	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "J" (7/ Catego

```
diamonds[1:5, Not("Carat")]
```

	Color	Clarity	Depth
1	CategoricalValue{String,UInt8} "E" (2/	CategoricalValue{String,UInt8} "SI2" (	61.5
2	CategoricalValue{String,UInt8} "E" (2/	CategoricalValue{String,UInt8} "SI1" (	59.8
3	CategoricalValue{String,UInt8} "E" (2/	CategoricalValue{String,UInt8} "VS1" (	56.9
4	CategoricalValue{String,UInt8} "I" (6/	CategoricalValue{String,UInt8} "VS2" (	62.4
5	CategoricalValue{String,UInt8} "J" (7/	CategoricalValue{String,UInt8} "SI2" (	63.3

```
diamonds[1:5, Not(["Carat", "Cut"])]
```

Селектор Between позволяет выбрать все столбцы между указанными:

	X	Y	Z
1	3.95	3.98	2.43
2	3.89	3.84	2.31
3	4.05	4.07	2.31
4	4.2	4.23	2.63
5	4.34	4.35	2.75

```
diamonds[1:5, Between("X", "Z")]
```

С помощью индексов можно осуществлять выбор наблюдений (строк), удовлетворяющих условиям на значения переменных (столбцов):

	Carat	Cut	Color
1	0.7	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "

	Carat	Cut	Color
2	0.86	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
3	0.7	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
4	0.71	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
5	0.78	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
6	0.7	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
7	0.7	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
8	0.96	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
9	0.73	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "

```
diamonds[diamonds.Carat .> 0.5, :] # все наблюдения со значением "Carat" строго больше 0.5
```

	Carat	Cut	Color
1	0.6	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "H
2	0.51	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "E
3	0.6	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "H
4	0.53	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "I
5	0.53	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "J
6	0.52	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "F
7	0.51	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "J
8	0.51	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "J
9	0.53	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "F
10	0.58	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "F

more

8633	0.82	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "H

```
# все наблюдения, где "Carat" строго больше 0.5 и "Price" строго меньше 2500
diamonds[(diamonds.Carat.> 0.5) .& (diamonds.Price .<2500) , :]
```

	Carat	Cut	Color
1	0.7	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
2	0.7	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
3	0.74	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
4	0.8	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
5	0.75	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
6	0.74	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "

	Carat	Cut	Color
7	0.81	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
8	0.59	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "

```
# все наблюдения, где "Carat" строго больше 0.5 и значение "Cut" равно "Ideal"
diamonds[(diamonds.Carat.> 0.5) .& (diamonds.Cut .== "Ideal") , :]
```

	Carat	Cut	Color
1	0.7	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
2	0.86	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
3	0.7	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
4	0.71	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
5	0.78	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
6	0.7	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
7	0.7	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
8	0.96	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
9	0.73	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
10	0.8	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "

more

25913	0.75	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "

```
# все наблюдения, где "Carat" строго больше 0.5 или "Price" строго больше 2750
diamonds[(diamonds.Carat .> 0.5) .| (diamonds.Price .> 2750), :]
```

Работа со столбцами

select, select!, transform и transform!

С помощью функций select и select! можно осуществлять выбор, переименование и транформацию столбцов датафрейма.

Функция select создает новый объект DataFrame. Она всегда возвращает объект DataFrame, даже если выбран только один столбец:

	Carat
1	0.23

	Carat
2	0.21
3	0.23
4	0.29
5	0.31
6	0.24
7	0.24
8	0.26
9	0.22

```
select(diamonds, :Carat) # датафрейм с одним столбцом
```

	Carat	Cut	Price
1	0.23	CategoricalValue{String,UInt8} "Ideal"	326
2	0.21	CategoricalValue{String,UInt8} "Premiu	326
3	0.23	CategoricalValue{String,UInt8} "Good"	327
4	0.29	CategoricalValue{String,UInt8} "Premiu	334
5	0.31	CategoricalValue{String,UInt8} "Good"	335
6	0.24	CategoricalValue{String,UInt8} "Very G	336
7	0.24	CategoricalValue{String,UInt8} "Very G	336
8	0.26	CategoricalValue{String,UInt8} "Very G	337
9	0.22	CategoricalValue{String,UInt8} "Fair"	337
10	0.23	CategoricalValue{String,UInt8} "Very G	338

more

```
select(diamonds, ["Carat", "Cut", "Price"]) # выбор нескольких указанных столбцов
```

	Carat	Cut	Depth	Table	Price	X
1	0.23	CategoricalValue{String,UInt8} "Ideal"	61.5	55.0	326	3.95
2	0.21	CategoricalValue{String,UInt8} "Premiu	59.8	61.0	326	3.89
3	0.23	CategoricalValue{String,UInt8} "Good"	56.9	65.0	327	4.05
4	0.29	CategoricalValue{String,UInt8} "Premiu	62.4	58.0	334	4.2
5	0.31	CategoricalValue{String,UInt8} "Good"	63.3	58.0	335	4.34
6	0.24	CategoricalValue{String,UInt8} "Very G	62.8	57.0	336	3.94
7	0.24	CategoricalValue{String,UInt8} "Very G	62.3	57.0	336	3.95

	Carat	Cut	Depth	Table	Price	X
8	0.26	CategoricalValue{String,UInt8} "Very G	61.9	55.0	337	4.07

```
select(diamonds, Not(["Clarity", "Color"])) # выбор всех столбцов, кроме указанных
```

	Carat	Cut	Color
1	0.23	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
2	0.21	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "
3	0.23	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
4	0.29	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "
5	0.31	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
6	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
7	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
8	0.26	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
9	0.22	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
10	0.23	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "

more

53940	0.75	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "

```
select(diamonds, Between("Carat", "Price")) # выбор диапазона столбцов
```

	Карат	Цена	Огранка
1	0.23	326	CategoricalValue{String,UInt8} "Ideal"
2	0.21	326	CategoricalValue{String,UInt8} "Premiu
3	0.23	327	CategoricalValue{String,UInt8} "Good"
4	0.29	334	CategoricalValue{String,UInt8} "Premiu
5	0.31	335	CategoricalValue{String,UInt8} "Good"
6	0.24	336	CategoricalValue{String,UInt8} "Very G
7	0.24	336	CategoricalValue{String,UInt8} "Very G
8	0.26	337	CategoricalValue{String,UInt8} "Very G
9	0.22	337	CategoricalValue{String,UInt8} "Fair"
10	0.23	338	CategoricalValue{String,UInt8} "Very G

more

53940	0.75	2757	CategoricalValue{String,UInt8} "Ideal"

```
# переименование столбцов
select(diamonds, "Carat" => "Карат", "Price" => "Цена", "Cut" => "Огранка")
```

	Price	LogPrice
1	326	5.7869
2	326	5.7869
3	327	5.78996
4	334	5.81114
5	335	5.81413
6	336	5.81711
7	336	5.81711
8	337	5.82008
9	337	5.82008
10	338	5.82305

more

53940 2757 7.9219

```
# трансформация столбца
select(diamonds, "Price", "Price" => (x -> log.(x)) => "LogPrice")
```

	Price	Price_log
1	326	5.7869
2	326	5.7869
3	327	5.78996
4	334	5.81114
5	335	5.81413
6	336	5.81711
7	336	5.81711
8	337	5.82008
9	337	5.82008
10	338	5.82305

more

53940 2757 7.9219

```
# альтернативный способ сделать то же самое
select(diamonds, "Price", "Price" => ByRow(log))
```

Чтобы не создавать копию датафрейма, можно использовать функцию select!

(для изменений inplace)

Функции transform и transform! работают аналогично select и select!. Единственное отличие - сохраняются все исходные столбцы:

	Carat	Cut	Color
1	0.23	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
2	0.21	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "
3	0.23	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
4	0.29	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "
5	0.31	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
6	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
7	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
8	0.26	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
9	0.22	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
10	0.23	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "

more

```
# добавляем новый столбец (логарифм цены) в исходный датафрейм
transform!(diamonds, "Price" => ByRow(log))
```

	Carat	Cut	Color
1	0.23	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
2	0.21	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "
3	0.23	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
4	0.29	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "
5	0.31	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
6	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
7	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
8	0.26	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
9	0.22	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
10	0.23	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "

more

```
diamonds # в датафрейм добавился столбец
```

Фреймворки для работы с данными и запросами

DataFramesMeta

Догрузим и запустим необходимые пакеты:

```
begin
    Pkg.add(["DataFramesMeta", "Query"])

    using DataFramesMeta
    using Query
end
```

Рассматриваемый пакет предоставляет короткий способ вызова столбцов DataFrame.

Вызов столбца тут осуществляется только по его названию. Помимо этого можно создавать цепи вызовом макроса @linq: оператор |> подает результат вычисления предыдущего шага на вход в следующее выражение. Например:

	Mass	Price
1	0.35	552
2	0.3	552
3	0.3	552
4	0.3	552
5	0.42	552
6	0.28	553
7	0.32	553
8	0.31	553
9	0.31	553
10	0.24	553
	more	
53101	0.75	2757

```
@linq diamonds |>
    where(:Price .> 500) |>
    select(Mass=:Carat, :Price)
```

Данный фреймворк также позволяет делить данные из исходного датасета на группы вычислять функции от групп, комбинировать полученные результаты.

Например, для группировки качественных данных (типа CategoricalValue) можно использовать функцию sort(), а для создания нового столбца на основе имеющихся – функцию transform(). Чтобы применить функцию ко всему столбцу (поточечно) непосредственно после ее названия нужно поставить знак точки:

	Carat	Cut	Color
1	0.22	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
2	0.86	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
3	0.96	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
4	0.7	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
5	0.7	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
6	0.91	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
7	0.91	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
8	0.98	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
9	0.84	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "
10	1.01	CategoricalValue{String,UInt8} "Fair"	CategoricalValue{String,UInt8} "

more

```
@linq diamonds |>
    sort(:Cut) |>
    transform(LogRatio = log.(:Price ./ :Carat))
```

Query

Другой способ извлекать необходимые паттерны из датасета – пакет для обработки запросов Query.

Запрос начинается с макроса @from. Первый аргумент – это переменная, пробегающая строки таблицы, второй – непосредственно запрашиваемая таблица.

Команда @where отфильтровывает строки, для которых значение в указанном столбце не удовлетворяет заданному условию.

Команда @select сохраняет требуемые данные в новый столбец. Чтобы выход имел вид таблицы следует использовать {}.

При помощи команды @collect можно задать тип объекта, желаемый получить в качестве ответа на запрос (по умолчанию возвращается iterator).

Тело запроса (часть кода после @from) начинается с команды begin и заканчивается командой end.

Выполним те же операции, что и в предыдущем разделе, но с ипользованием фреймворка Query:

```
q1 =
```

	Mass	Price

	Mass	Price
1	0.35	552
2	0.3	552
3	0.3	552
4	0.3	552
5	0.42	552
6	0.28	553
7	0.32	553
8	0.31	553
9	0.31	553
10	0.24	553

```
q1 = @from i in diamonds begin
    @where i.Price > 500
    @select {Mass=i.Carat, i.Price}
    @collect DataFrame
end
```

Вызов команды @collect без указания типа данных вернет ответ на запрос в формате array:

```
q2 =
  Int32[2757, 2757, 2757, 2759, 2759, 2759, 2759, 2759, 2760, 2760, 2760, 2760, 2
```

```
q2 = @from i in diamonds begin
    @where i.Price > 500 && i.Carat > 0.5
    @select i.Price
    @collect
end
```

Summarizing

Команда describe показывает статистическую информацию о датафрейме, среди которой такие значения: среднее, минимальное, максимальное значения, медиана, количество пропусков, тип переменной:

	variable	mean	min	median	
1	:Carat	0.79794	0.2	0.7	5.01
2	:Cut	nothing	CategoricalValue{String,UInt8} "Fair"	nothing	CategoricalVal
3	:Color	nothing	CategoricalValue{String,UInt8} "D" (1/	nothing	CategoricalVal
4	:Clarity	nothing	CategoricalValue{String,UInt8} "I1" (1	nothing	CategoricalVal
5	:Depth	61.7494	43.0	61.8	79.0

	variable	mean	min	median	
6	:Table	57.4572	43.0	57.0	95.0
7	:Price	3932.8	326	2401.0	18823
8	:X	5.73116	0.0	5.7	10.74
9	:Y	5.73453	0.0	5.71	58.9

```
describe(diamonds)
```

Данную команду можно применить также не только к целому датафрейму, но и к его отдельной части, например:

	variable	mean	min	median	r
1	:Cut	nothing	CategoricalValue{String,UInt8} "Fair"	nothing	CategoricalValue{

```
describe(diamonds[!, [:Cut]])
```

Также с помощью пакета Statistics можно найти статистические значения для отдельного столбца. Например, среднее:

```
using Statistics
```

```
0.7979397478680014
```

```
mean(diamonds[!, "Carat"]) #среднее
```

Аналогично можно найти другие значения:

```
2401.0
```

```
median(diamonds[!, "Price"]) #медиана
```

```
18823
```

```
maximum(diamonds[!, "Price"]) #максимум
```

```
326
```

```
minimum(diamonds[!, "Price"]) #минимум
```

Функция combine позволяет применять функцию к отдельному столбцу. Например, следующим образом:

	Price_sum
1	212135217

```
combine(diamonds, :Price .=> sum) #сумма по столбцу
```

	Price_maximum
1	18823

```
combine(diamonds, :Price => maximum) #максимальное значение по столбцу
```

Select позволяет получить тот же результат, только на выходе дает то же число строк, которое изначально в датафрейме:

	Price_maximum
1	18823
2	18823
3	18823
4	18823
5	18823
6	18823
7	18823
8	18823
9	18823
10	18823
	more
53940	18823

```
select(diamonds, :Price => maximum) #максимальное значение по столбцу
```

Работа со столбцами

Все функции, которые меняют датафрейм по умолчанию копируют его столбцы. Так, например:

```
diamonds2 =
```

	Carat	Cut	Color
1	0.23	CategoricalValue{String,UInt8} "Ideal"	CategoricalValue{String,UInt8} "
2	0.21	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "

	Carat	Cut	Color
3	0.23	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
4	0.29	CategoricalValue{String,UInt8} "Premiu	CategoricalValue{String,UInt8} "
5	0.31	CategoricalValue{String,UInt8} "Good"	CategoricalValue{String,UInt8} "
6	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
7	0.24	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "
8	0.26	CategoricalValue{String,UInt8} "Very G	CategoricalValue{String,UInt8} "

```
false
```

```
diamonds2.Price === diamonds.Price
```

Тем не менее, будьте осторожны! Функции, которые заканчиваются символом ! могут изменить столбец, который ей скормили в качестве аргумента. Обратите внимание, к примеру, на такую ситуацию:

```
price =
  Int32[326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 342, 344, 34
```

```
price = diamonds.Price
```

```
df =
```

	price
1	326
2	326
3	327
4	334
5	335
6	336
7	336
8	337
9	337
10	338
	more
53940	2757

```
df = DataFrame(price=price)
```

	price

	price
1	326
2	326
3	327
4	334
5	335
6	336
7	336
8	337
9	337

```
sort!(df)
```

```
Int32[326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 342, 344, 34
```

```
price
```

```
100
```

```
df.price[1] = 100
```

	price
1	100
2	326
3	327
4	334
5	335
6	336
7	336
8	337
9	337
10	338

more

53940	18823

```
df
```

```
Int32[326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 342, 344, 34
```

```
price
```

Как Вы могли заметить, в указанном выше примере вектор цен price, взятый из таблицы diamonds не был изменён, так как конструктор DataFrame(price=price) по умолчанию создаёт

копию.

Тем не менее, если Вам вдруг понадобится обратиться напрямую к некоторому столбцу датафрейма, это возможно сделать при помощи функции eachcol.

`df2 =`

	price
1	326
2	326
3	327
4	334
5	335
6	336
7	336
8	337
9	337
10	338
more	
53949	2757

```
df2 = DataFrame(price=price)
```

```
true
```

```
df2.price == price
```

```
true
```

```
df2[!, 1] !== price
```

```
true
```

```
eachcol(df2)[1] === df2.price
```

Будьте осторожны, когда меняете столбцы из датафрейма таким способом!