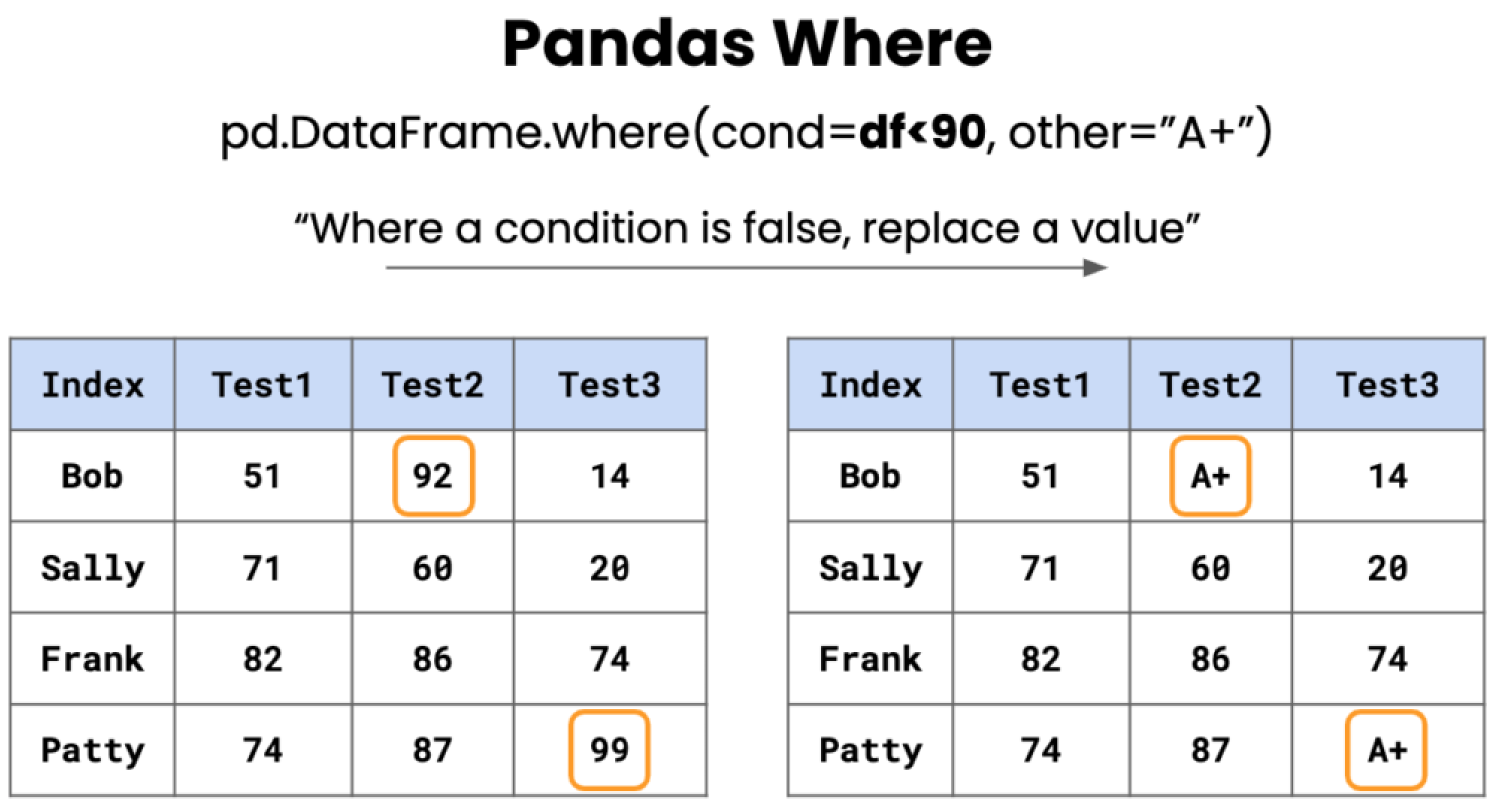
I] Filtering Data from the Dataframe

i) where()

- It is used to check one or multiple conditions of an expression in DataFrame.
- By default, it replaces with NaN value.
- It is similar to if-then/if else conditions. It provides a parameter to replace with any custom value to replace with another value when the condition becomes False.

Syntax:

> **DataFrame.where(cond = condition_to_check, other="Value To Fill")**



Pandas Where

pd.DataFrame.where(cond=**df<90**, other="A+")

"Where a condition is false, replace a value"

Index	Test1	Test2	Test3
Bob	51	92	14
Sally	71	60	20
Frank	82	86	74
Patty	74	87	99

Index	Test1	Test2	Test3
Bob	51	A+	14
Sally	71	60	20
Frank	82	86	74
Patty	74	87	A+

Let's create a DataFrame and explore the where() function usage with examples.

```
# Create DataFrame
import pandas as pd
import numpy as np
technologies= {
    'Courses':["Spark","PySpark","Spark","Python","PySpark"],
    'Fee' :[22000,25000,23000,24000,26000],
    'Discount':[1500,1000,1200,800,1300],
    'Duration':['30days','50days','30days','35days','40days']
          }
df = pd.DataFrame(technologies)
print(df)
```

```
        Courses    Fee  Discount Duration
```

```
0    Spark  22000      1500   30days
1  PySpark  25000      1000   50days
2    Spark  23000      1200   30days
3   Python  24000       800   35days
4  PySpark  26000      1300   40days
```

```
# Default example
df2=df.where(df.Fee > 23000)
print(df2)
```

```
   Courses      Fee  Discount Duration
0      NaN      NaN       NaN      NaN
1  PySpark  25000.0    1000.0   50days
2      NaN      NaN       NaN      NaN
3   Python  24000.0     800.0   35days
4  PySpark  26000.0    1300.0   40days
```

Update with Another Value

Now, let’s update with a custom value. The below example updates all rows of DataFrame with value ‘NA’ when condition Fee > 23000 becomes False.

```
# Use other param
df2=df.where(df.Fee > 23000,'NA')
print(df2)
```

```
   Courses    Fee Discount Duration
0       NA     NA       NA       NA
1  PySpark  25000     1000   50days
2       NA     NA       NA       NA
3   Python  24000      800   35days
4  PySpark  26000     1300   40days
```

where() with Multiple Columns & Conditions

```
# Where on multiple columns & conditions
cond1 = df.Fee > 23000
cond2 = df.Discount > 900

df2 = df.where(cond1 & cond2, other='NA')
print(df2)
```

```
   Courses    Fee Discount Duration
0       NA     NA       NA       NA
1  PySpark  25000     1000   50days
2       NA     NA       NA       NA
3       NA     NA       NA       NA
4  PySpark  26000     1300   40days
```

Update on Existing DataFrame

In order to update in place use inplace=True param. When used this param with the value true, where() function returns None.

```
# Updating on existing DataFrame
df.where(cond1 & cond2, 'NA')
print(df)
```

```
   Courses    Fee  Discount Duration
0    Spark  22000      1500   30days
1  PySpark  25000      1000   50days
2    Spark  23000      1200   30days
3   Python  24000       800   35days
4  PySpark  26000      1300   40days
```

ii) query()

- It is used to query the rows based on the expression (single or multiple column conditions).
- The query() method takes a query expression as a string parameter, which has to evaluate to either True of False.
- It returns the DataFrame where the result is True according to the query expression.

Syntax:

> **DataFrame.query(expr, inplace, *kwargs)**

```
import pandas as pd
import numpy as np
technologies= {
    'Courses':["Spark","PySpark","Hadoop","Python","Pandas"],
    'Fee' :[22000,25000,23000,24000,26000],
    'Duration':['30days','50days','30days', None,np.nan],
    'Discount':[1000,2300,1000,1200,2500]
          }
df = pd.DataFrame(technologies)
print(df)
```

```
   Courses    Fee Duration  Discount
0    Spark  22000   30days      1000
1  PySpark  25000   50days      2300
2   Hadoop  23000   30days      1000
3   Python  24000     None      1200
4   Pandas  26000      NaN      2500
```

Using DataFrame.query()

```
# Query all rows with Courses equals 'Spark'
a=df.query("Courses == 'Spark'")

print(a)
```

```
  Courses    Fee Duration  Discount
0   Spark  22000   30days      1000
```

..............................OR

In case you wanted to use a variable in the expression, use @ character.

```
# Query Rows by using Python variable

value='Spark'

a =df.query("Courses == @value")

print(a)
```

```
  Courses    Fee Duration  Discount
0   Spark  22000   30days      1000
```

If you notice the above examples return a new DataFrame after filtering the rows. if you wanted to update the existing DataFrame use inplace=True

```
# Replace current esisting DataFrame
a = df.query("Courses == 'Spark'",inplace=True)

print(a)
```

```
None
```

If you wanted to select based on column value not equals then use != operator.

```
# not equals condition

a = df.query("Courses != 'Spark'")

print(a)
```

```
Empty DataFrame
Columns: [Courses, Fee, Duration, Discount]
Index: []
```

Select Rows Based on List of Column Values

If you have values in a python list and wanted to select the rows based on the list of values, use in operator, it's like checking a value contains in a list of string values.

```
# Query Rows by list of values

a = df.query("Courses in ('Spark','PySpark')")

print(a)
```

```
  Courses    Fee Duration  Discount
0   Spark  22000   30days      1000
```

```
# Query Rows by list of values
values=['Spark','PySpark']

a = df.query("Courses in @values")

print(a)
```

```
  Courses    Fee Duration  Discount
0   Spark  22000   30days      1000
```

```
# Query Rows not in list of values

values=['Spark','PySpark']

a = df.query("Courses not in @values")

print(a)
```

```
Empty DataFrame
Columns: [Courses, Fee, Duration, Discount]
Index: []
```

Query with Multiple Conditions

```
a = df.query("Fee>= 200 and Fee <= 200")
print(a)
```

```
Empty DataFrame
Columns: [Courses, Fee, Duration, Discount]
Index: []
```

II] Unique Values, Value Count

How to count unique values of a column in pandas DataFrame?

Let’s create a DataFrame.

```
# Loading a Sample Pandas DataFrame
import pandas as pd
import numpy as np
df = pd.DataFrame.from_dict({
    'Name': ['Nik', 'Kate', 'Evan', 'Kyra', np.NaN],
    'Age': [33, 32, 40, 57, np.NaN],
    'Location': ['Toronto', 'London', 'New York', np.NaN, np.NaN]
})
print(df)
```

```
   Name   Age  Location
0   Nik  33.0   Toronto
1  Kate  32.0    London
2  Evan  40.0  New York
3  Kyra  57.0       NaN
4   NaN   NaN       NaN
```

unique()

- To get unique values in a column i.e. by removing duplicate values.
- **It includes NaN as unique value.**

```
# Get Unique Count using Series.unique()
a = df.Name.unique()

print(a)
```

```
['Nik' 'Kate' 'Evan' 'Kyra' nan]
```

```
a = df.Age.unique()

print(a)
```

```
[33. 32. 40. 57. nan]
```

```
a = df.Name.unique().size

print(a)
```

```
    5
```

............................OR

nunique()

nunique() **returns number of unique elements in the DataFrame excluding NaN values.** If you wanted to include NaN values use dropna parameter to False.

```
# Using Series.nunique()
count = df.Name.nunique()

print(count)
```

```
    4
```

```
# Using Series.nunique()
count = df.Age.nunique()

print(count)
```

```
    4
```

How to count Unique Values in Multiple Columns?

In order to get the count of unique values on multiple columns use pandas DataFrame.drop_duplicates() which drop duplicate rows from pandas DataFrame. This eliminates duplicates and return DataFrame with unique rows.

```
# Count unique on multiple columns
count = df[['Courses','Fee']].drop_duplicates()

print(count)
```

```
       Courses    Fee
    0    Spark  20000
    1  PySpark  25000
    2   Python  22000
    3   Pandas  30000
    4   Python  25000
```

How to Count the number of occurrence of any value?

In case if you want to get the frequency of a column use Series.value_counts(), This returns the Count of Frequency of a Value in Column

```
print(df.Courses.value_counts())
```

```
    Spark      2
    Python     2
    Pandas     2
```

```
PySpark    1
Name: Courses, dtype: int64
```

III] Pandas Membership

isin() function

It exists in both pandas DataFrame & Series which is used to check if the object (Series or DataFrame) contains the elements from list, Series, Dict.

It returns same as caller object of booleans indicating if each row cell/element is in values.

To Check if a value exists in a DataFrame using in & not in operator.

isin() is used

```
import pandas as pd
df = pd.DataFrame({
    'Courses' :['Spark','Python','Java'],
    'Fee' :[22000,25000,23000,],
    'Duration':['30days','50days','30days']
          })
print(df)
```

```
  Courses    Fee Duration
0   Spark  22000   30days
1  Python  25000   50days
2    Java  23000   30days
```

DataFrame.isin() Example

> When a python list is passed as a parameter value to the DataFrame.isin() function, it checks whether each cell value from DataFrame is present in the list, if found, shows True otherwise False (When a value is not present). The resultant DataFrame just contains boolean values.

```
# isin() with list of values
print(df.isin(['Spark','Python',23000,'50days']))
```

```
   Courses    Fee  Duration
0     True  False     False
1     True  False      True
2    False   True     False
```

Using Dictionary

The above example doesn't check values in a specific DataFrame column, In order to check the values in a specific column use Dictionary object as param. When a python Dict is passed as a param to the isin(), you should have a column name as the key and elements you wanted to check as Dict value. With this, you can check values in multiple columns.

```
# check by column name
print(df.isin({'Courses': ['Spark', 'Python']}))
```

```
   Courses    Fee  Duration
0     True  False     False
1     True  False     False
2    False  False     False
```

Complete Example of DataFrame & Series isin()

```
# Create a pandas DataFrame.
import pandas as pd
import numpy as np

df = pd.DataFrame({
    'Courses' :['Spark','Python','Java'],
    'Fee' :[22000,25000,23000,],
    'Duration':['30days','50days','30days']
          })
print(df)

# List of values
print(df.isin(['Spark','Python',23000,'50days']))

# check by column
print(df.isin({'Courses': ['Spark', 'Python',23000]}))
```

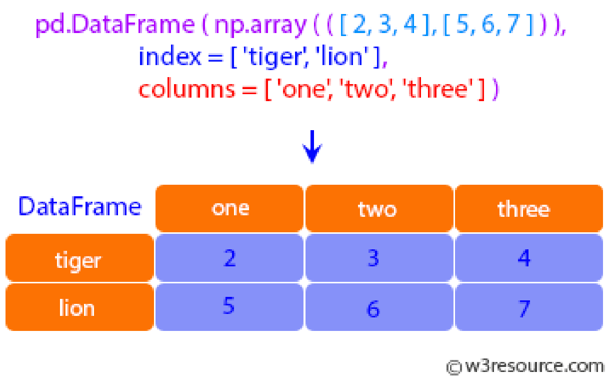
```
  Courses    Fee Duration
0   Spark  22000   30days
1  Python  25000   50days
2    Java  23000   30days
   Courses    Fee  Duration
0     True  False     False
1     True  False      True
2    False   True     False
   Courses    Fee  Duration
0     True  False     False
1     True  False     False
2    False  False     False
```

Additional Points:

i) Filtering in DataFrame

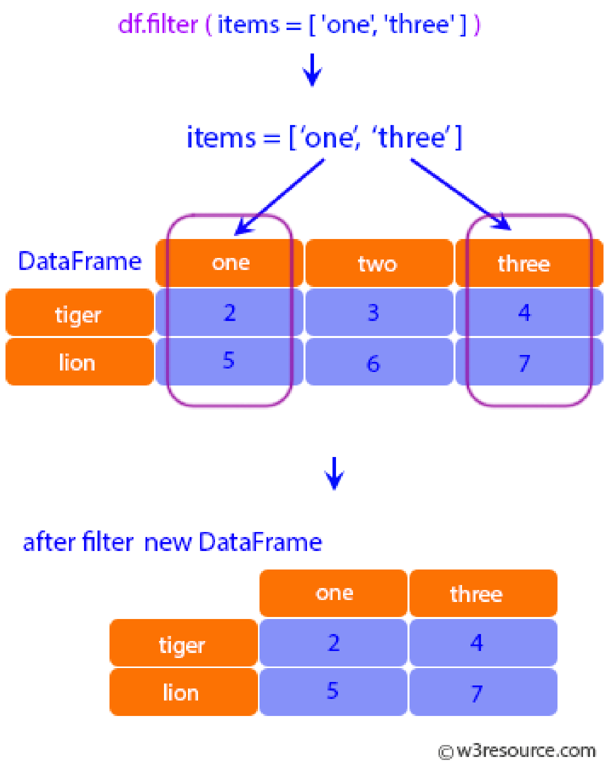
```
import numpy as np
import pandas as pd

df = pd.DataFrame(np.array(([2, 3, 4], [5, 6, 7])),
                  index=['tiger', 'lion'],
                  columns=['one', 'two', 'three'])
```


pd.DataFrame (np.array (([2, 3, 4], [5, 6, 7])),
index = ['tiger', 'lion'],
columns = ['one', 'two', 'three'])
DataFrame
one
two
three
tiger
2
3
4
lion
5
6
7
© w3resource.com

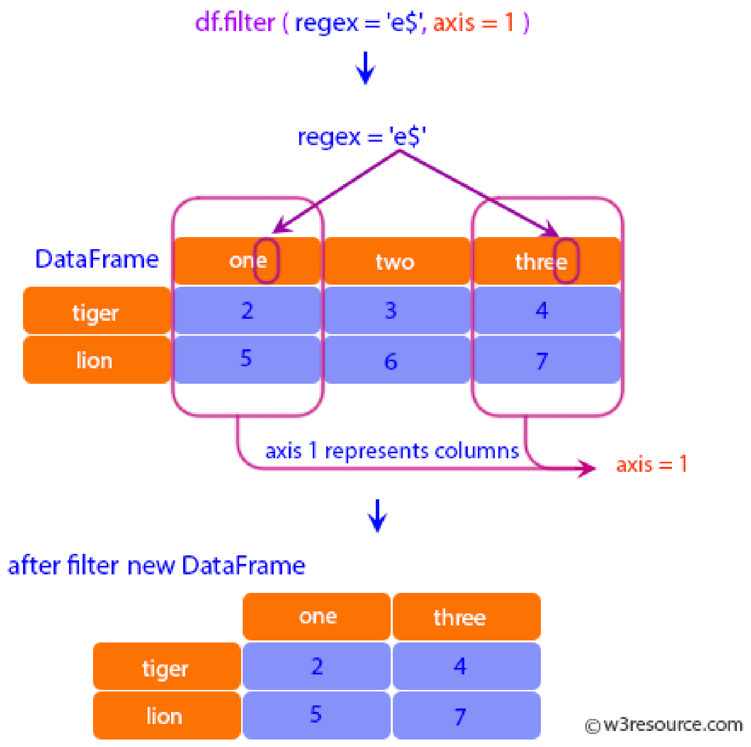
```
# select columns by name
df.filter(items=['one', 'three'])
```

	one	three
tiger	2	4
lion	5	7



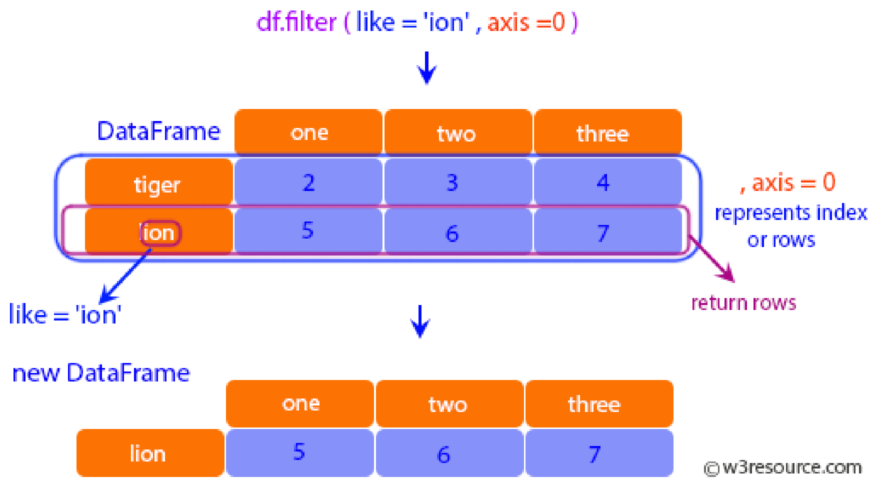
```
# select columns by regular expression
df.filter(regex='e$', axis=1)
```

	one	three
tiger	2	4
lion	5	7



```
# select rows containing 'ion'
df.filter(like='ion', axis=0)
```

	one	two	three
lion	5	6	7



Additional Points:

i) filter()

- Pandas **filter() function** filters the the data in the DataFame based on a condition.
- The returned DataFrame contains only rows and columns that are specified with the function.
- It doesn’t update the existing DataFrame instead it always returns a new one.

Syntax:

DataFrame.filter(items, like, regex, axis)

Parameters:

1. item – Takes list of axis labels that you wanted to filter.
2. like – Takes axis string label that you wanted to filter
3. regex – regular expression
4. axis – {0 or 'index', 1 or 'columns', None}, default None. When not specified it used columns.

Let's create a pandas DataFrame from Dict and understand usage with examples.

```
import pandas as pd
technologies= {
    'Courses':["Spark","PySpark","Spark","Java","PySpark","PHP"],
    'Fee' :[22000,25000,23000,24000,26000,27000],
    'Duration':['30days','50days','30days','60days','35days','30days']
          }
df = pd.DataFrame(technologies)
print(df)
```

```
        Courses    Fee Duration
     0    Spark  22000   30days
     1  PySpark  25000   50days
     2    Spark  23000   30days
     3     Java  24000   60days
     4  PySpark  26000   35days
     5      PHP  27000   30days
```

1) Filter Columns by Labels

By default pandas.DataFrame.filter() select the columns by labels you specified using item, like, and regex parameters.

You can also explicitly specify axis=1 to select columns.

```
# Filter columns

df2=df.filter(items=['Courses','Fee'])

print(df2)
```

```
        Courses    Fee
     0    Spark  22000
     1  PySpark  25000
     2    Spark  23000
     3     Java  24000
     4  PySpark  26000
     5      PHP  27000
```

Note that items parameter is used to match on exact values

2) Filter Rows by Index

Use axis=0 on filter() function to filter rows by index (indices).

The below example filters rows by index 3 and 5.

```
# Filter rows
```

```
df2=df.filter(items=[3,5], axis=0)
print(df2)
```

```
  Courses    Fee Duration
3    Java  24000   60days
5     PHP  27000   30days
```

To filter columns with regular expressions, use regex parameter.

The below example filters column that ends with the character 'e'.

```
# Filter column names by regex
df2 = df.filter(regex='e$', axis=1)
print(df2)
```

```
     Fee
0  22000
1  25000
2  23000
3  24000
4  26000
5  27000
```

The below example filters column that ends with the character 'n'.

```
# Filter column names by regex
df2 = df.filter(regex='n$', axis=1)
print(df2)
```

```
  Duration
0   30days
1   50days
2   30days
3   60days
4   35days
5   30days
```

```
# Filter Columns using like
df2 = df.filter(like='ration', axis=1)
print(df2)
```

```
  Duration
0   30days
1   50days
2   30days
3   60days
4   35days
5   30days
```

```
# Filter row using like
df2 = df.filter(like='4', axis=0)
print(df2)
```

```
   Courses    Fee Duration
4  PySpark  26000   35days
```

ii) Filtering in CSV Files

```python
# importing pandas as pd
import pandas as pd

# Creating the dataframe
df = pd.read_csv("/content/drive/MyDrive/nba.csv")

# Print the dataframe
df
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
453	Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
454	Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24.0	C	26.0	7-0	231.0	Kansas	947276.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

```python
# applying filter function
df.filter(["Name", "College", "Salary"])
```

	Name	College	Salary
0	Avery Bradley	Texas	7730337.0
1	Jae Crowder	Marquette	6796117.0
2	John Holland	Boston University	NaN
3	R.J. Hunter	Georgia State	1148640.0
4	Jonas Jerebko	NaN	5000000.0
...	...	...	...
453	Shelvin Mack	Butler	2433333.0
454	Raul Neto	NaN	900000.0
455	Tibor Pleiss	NaN	2900000.0
456	Jeff Withey	Kansas	947276.0
457	NaN	NaN	NaN

458 rows × 3 columns

+ Code + Text