Dealing with Columns

- In order to deal with columns, we perform basic operations on columns like selecting, deleting, adding and renaming.

A) Column Selection

- In Order to select a column in Pandas DataFrame, we can either access the columns by calling them by their columns name.

Example 1: Basic Method

```
# Import pandas package
import pandas as pd

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# select two columns
print(df[['Name', 'Qualification']])
```

```
      Name Qualification
0     Jai           Msc
1  Princi            MA
2  Gaurav           MCA
3    Anuj           Phd
```

```
# Importing pandas as pd
from pandas import DataFrame

# Creating a data frame
Data = {'Name': ['Mohe', 'Shyni', 'Parul', 'Sam'],
        'ID': [12, 43, 54, 32],
        'Place': ['Delhi', 'Kochi', 'Pune', 'Patna']
       }

df = DataFrame(Data, columns = ['Name', 'ID', 'Place'])

# Print original data frame
print("Original data frame:")
display(df)
```

```
print("Selected column: ")
display(df[['Name', 'ID']] )
```

```
Original data frame:
```

	Name	ID	Place
0	Mohe	12	Delhi
1	Shyni	43	Kochi
2	Parul	54	Pune
3	Sam	32	Patna

```
Selected column:
```

	Name	ID
0	Mohe	12
1	Shyni	43
2	Parul	54
3	Sam	32

Example 2: Select second to fourth column.

```
# Import pandas package
import pandas as pd

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# select all rows
# and second to fourth column
df[df.columns[1:4]]
```

	Age	Address	Qualification
0	27	Delhi	Msc
1	24	Kanpur	MA
2	22	Allahabad	MCA
3	32	Kannauj	Phd

Example 3: Using loc[ ]

Select two columns

```
# Import pandas package
import pandas as pd

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# select three rows and two columns
df.loc[1:3, ['Name', 'Qualification']]
```

	Name	Qualification
1	Princi	MA
2	Gaurav	MCA
3	Anuj	Phd

Select one to another columns. In our case we select column name “Name” to “Address”.

```
# Import pandas package
import pandas as pd

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# select two rows and
# column "name" to "Address"
# Means total three columns
df.loc[0:1, 'Name':'Address']
```

	Name	Age	Address
0	Jai	27	Delhi
1	Princi	24	Kanpur

First filtering rows and selecting columns by label format and then Select all columns.

```
# Import pandas package
import pandas as pd

# Define a dictionary containing employee data
```

```
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']
       }
# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# .loc DataFrame method
# filtering rows and selecting columns by label
# format
# df.loc[rows, columns]
# row 1, all columns
df.loc[0, :]
```

```
Name               Jai
Age                 27
Address          Delhi
Qualification      Msc
Name: 0, dtype: object
```

Example 4: Using iloc[ ]

Select first two column.

```
# Import pandas package
import pandas as pd

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Remember that Python does not
# slice inclusive of the ending index.
# select all rows
# select first two column
df.iloc[:, 0:2]
```

	Name	Age
0	Jai	27
1	Princi	24
2	Gaurav	22
3	Anuj	32

Select all or some columns, one to another using .iloc.

```
# Import pandas package
import pandas as pd

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# iloc[row slicing, column slicing]
df.iloc [0:2, 1:3]
```

	Age	Address
0	27	Delhi
1	24	Kanpur

B) Column Addition:

Method #1: By declaring a new list as a column.

In Order to add a column in Pandas DataFrame, we can declare a new list as a column and add to a existing Dataframe.

```
# Import pandas package
import pandas as pd

# Define a dictionary containing Students data
data = {'Name': ['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Height': [5.1, 6.2, 5.1, 5.2],
        'Qualification': ['Msc', 'MA', 'Msc', 'Msc']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Declare a list that is to be converted into a column
address = ['Delhi', 'Bangalore', 'Chennai', 'Patna']

# Using 'Address' as the column name
# and equating it to the list
df['Address'] = address

# Observe the result
print(df)
```

```
     Name  Height Qualification    Address
0     Jai     5.1           Msc      Delhi
1  Princi     6.2            MA  Bangalore
```

```
     2  Gaurav     5.1           Msc    Chennai
     3    Anuj     5.2           Msc      Patna
```

Method #2: By using a dictionary

We can use a Python dictionary to add a new column in pandas DataFrame. Use an existing column as the key values and their respective values will be the values for a new column.

```
# Import pandas package
import pandas as pd

# Define a dictionary containing Students data
data = {'Name': ['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Height': [5.1, 6.2, 5.1, 5.2],
        'Qualification': ['Msc', 'MA', 'Msc', 'Msc']}

# Define a dictionary with key values of
# an existing column and their respective
# value pairs as the # values for our new column.
address = {'Delhi': 'Jai', 'Bangalore': 'Princi',
           'Patna': 'Gaurav', 'Chennai': 'Anuj'}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Provide 'Address' as the column name
df['Address'] = address

# Observe the output
print(df)
```

```
          Name  Height Qualification    Address
     0     Jai     5.1           Msc      Delhi
     1  Princi     6.2            MA  Bangalore
     2  Gaurav     5.1           Msc      Patna
     3    Anuj     5.2           Msc    Chennai
```

Method #3: By using DataFrame.insert()

It gives the freedom to add a column at any position we like and not just at the end. It also provides different options for inserting the column values.

```
# Import pandas package
import pandas as pd

# Define a dictionary containing Students data
data = {'Name': ['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Height': [5.1, 6.2, 5.1, 5.2],
        'Qualification': ['Msc', 'MA', 'Msc', 'Msc']}

# Convert the dictionary into DataFrame
```

```
df = pd.DataFrame(data)

# Using DataFrame.insert() to add a column
df.insert(2, "Age", [21, 23, 24, 21], True)

# Observe the result
print(df)
```

```
      Name  Height  Age Qualification
0      Jai     5.1   21           Msc
1   Princi     6.2   23            MA
2   Gaurav     5.1   24           Msc
3     Anuj     5.2   21           Msc
```

This method will create a new dataframe with a new column added to the old dataframe.

C) Column Deletion

Pandas provide data analysts a way to delete and filter data frame using .drop() method. Rows or columns can be removed using index label or column name using this method.

Syntax:

> **DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')**

Parameters:

labels: String or list of strings referring row or column name.

axis: int or string value, 0 'index' for Rows and 1 'columns' for Columns.

index or columns: Single label or list. index or columns are an alternative to axis and cannot be used together.

level: Used to specify level in case data frame is having multiple level index. inplace: Makes changes in original Data Frame if True.

errors: Ignores error if any value from the list doesn't exists and drops rest of the values when errors = 'ignore'

Return type: Dataframe with dropped values

Example #1 : Dropping columns with column name

```
# importing pandas module
import pandas as pd

# making data frame from csv file
data = pd.read_csv("/content/drive/MyDrive/nba.csv", index_col ="Name" )

# dropping passed columns
```

```
data.drop(["Team", "Weight"], axis = 1, inplace = True)

# display
print(data)
```

```
                Number Position   Age Height            College     Salary
Name
Avery Bradley      0.0       PG  25.0    6-2              Texas  7730337.0
Jae Crowder       99.0       SF  25.0    6-6          Marquette  6796117.0
John Holland      30.0       SG  27.0    6-5  Boston University        NaN
R.J. Hunter       28.0       SG  22.0    6-5      Georgia State  1148640.0
Jonas Jerebko      8.0       PF  29.0   6-10                NaN  5000000.0
...                ...      ...   ...    ...                ...        ...
Shelvin Mack       8.0       PG  26.0    6-3             Butler  2433333.0
Raul Neto         25.0       PG  24.0    6-1                NaN   900000.0
Tibor Pleiss      21.0        C  26.0    7-3                NaN  2900000.0
Jeff Withey       24.0        C  26.0    7-0             Kansas   947276.0
NaN                NaN      NaN   NaN    NaN                NaN        NaN

[458 rows x 6 columns]
```

Example #2: Dropping Rows by index label

In his code, A list of index labels is passed and the rows corresponding to those labels are dropped using .drop() method.

```
# importing pandas module
import pandas as pd

# making data frame from csv file
data = pd.read_csv("/content/drive/MyDrive/nba.csv", index_col ="Name" )

# dropping passed values
data.drop(["Avery Bradley", "John Holland", "R.J. Hunter",
                           "R.J. Hunter"], inplace = True)

# display
data
```

Name	Team	Number	Position	Age	Height	Weight	College	Salary
Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0

Double-click (or enter) to edit

Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
...	...	...	...	...	...	...	...	...
Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	900000.0
Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0	NaN	2900000.0
Jeff Withey	Utah Jazz	24.0	C	26.0	7-0	231.0	Kansas	947276.0
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

455 rows × 8 columns