Learning_Pandas_Part_1_BasicOperations

June 20, 2021

0.0.1 Prepared by Abhishek Kumar

0.0.2 https://www.linkedin.com/in/abhishekkumar-0311/

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# To get multiple outputs in the same cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

%matplotlib inline
```

1 A.) Data Input/Output

1.1 1. Dataframe creation

1.1.1 There are multiple ways to create a dataframe.

1. Through lists, which i have demonstrated here.
2. Through dictionaries
3. Using pd.DataFrame()
4. Using pd.from_records()

[3]:
```
salary = [['Google', 'Machine Learning Engineer', 121000],
['Google', 'Data Scientist', 109000],
['Google', 'Tech Lead', 129000],
['Facebook', 'Data Scientist', 103000]]

columns_name=['Company', 'Job','Salary']

emp_df = pd.DataFrame(salary,columns=columns_name)
emp_df
```

[3]:
```
    Company                        Job  Salary
0    Google  Machine Learning Engineer  121000
1    Google             Data Scientist  109000
```

```
2    Google                  Tech Lead  129000
3  Facebook             Data Scientist  103000
```

1.2 2. Import - Creating a dataframe from external file, here csv

```
[4]: # cov_df = pd.read_csv('E:\VCS\GitHub\Machine-Learning-with-Python\data\Data␣
      ↪USA Cart\covid19.csv')
     # cov_df.head()
```

1.3 3. Export - Writing a dataframe to external file, here csv

```
[5]: #pd.to_csv('path\df.csv')
```

2 B.) Data Operations

2.1 1. Copying - Creating a new dF from existing dF

2.1.1 SAS

1. In existing dataFrame - data xyz; set xyz; run;
2. In new dataFrame - data abc; set xyz; run;

2.1.2 Python

1. In existing dataFrame - df_have = df.have
2. In new dataFrame - df_want = df.have.copy()

Note : The manipulations in python can be chained on the right hand side. 1. Copying through variables - Changes made in new dataframe are also reflected in the old dataframe since the new variable 'emp_df_0' is just a pointer to old one 'emp_df'. 2. .copy() - df.copy() creates an independent copy of the dataset

```
[6]: # Copying through variables

     emp_df_0 = emp_df
     emp_df_0
```

```
[6]:    Company                        Job  Salary
     0    Google  Machine Learning Engineer  121000
     1    Google             Data Scientist  109000
     2    Google                  Tech Lead  129000
     3  Facebook             Data Scientist  103000
```

```
[7]: # Updating the new dF
     emp_df_0['Salary']=emp_df_0['Salary']+1000

     # Comparison of dataframes
     emp_df_0 == emp_df
```

```
# Both the dF have the same values
emp_df_0
emp_df
```

```
[7]:    Company   Job  Salary
     0     True  True    True
     1     True  True    True
     2     True  True    True
     3     True  True    True
```

```
[7]:     Company                        Job  Salary
     0    Google  Machine Learning Engineer  122000
     1    Google             Data Scientist  110000
     2    Google                  Tech Lead  130000
     3  Facebook             Data Scientist  104000
```

```
[7]:     Company                        Job  Salary
     0    Google  Machine Learning Engineer  122000
     1    Google             Data Scientist  110000
     2    Google                  Tech Lead  130000
     3  Facebook             Data Scientist  104000
```

```
[8]: # Using Copy() to create a new dF
     emp_df_1 = emp_df.copy() # df.copy(deep=false) - does not create an independent␣
      ↪copy

     # Updating the new dF
     emp_df_1['Salary']=emp_df_1['Salary']+1000

     # Comparison of dataframes
     emp_df_1 == emp_df

     # Both the dF have the same values
     emp_df_1
     emp_df
```

```
[8]:    Company   Job  Salary
     0     True  True   False
     1     True  True   False
     2     True  True   False
     3     True  True   False
```

```
[8]:     Company                        Job  Salary
     0    Google  Machine Learning Engineer  123000
     1    Google             Data Scientist  111000
     2    Google                  Tech Lead  131000
```

```
3  Facebook             Data Scientist  105000
```

```
[8]:     Company                        Job  Salary
     0     Google  Machine Learning Engineer  122000
     1     Google             Data Scientist  110000
     2     Google                  Tech Lead  130000
     3   Facebook             Data Scientist  104000
```

2.1.3 Analysis of dataFrame Comparison

```
[9]: # Comparison of dataframes  with == : It does not handle missing ( NaN ) values
     # Return False if there are NaN values.
     cmp_1 = emp_df_1 == emp_df
     cmp_1

     # Comparison Analysis
     cmp_1.all()
     cmp_1.all().sum()

     # Comparing 2 dataframes with .eq() : It does not handle missing ( NaN ) values
     # Return False if there are NaN values.
     emp_df.eq(emp_df_0).all()
     emp_df.eq(emp_df_1).all().sum()

     # Comparing 2 dataframes with .equals() : It handles missing ( NaN ) values
     emp_df.equals(emp_df_0)
     emp_df.equals(emp_df_1)
```

```
[9]:    Company   Job  Salary
     0     True  True   False
     1     True  True   False
     2     True  True   False
     3     True  True   False
```

```
[9]: Company     True
     Job         True
     Salary     False
     dtype: bool
```

```
[9]: 2
```

```
[9]: Company    True
     Job        True
     Salary     True
     dtype: bool
```

```
[9]: 2
```

```
[9]: True
```

```
[9]: False
```

2.2 2. Creation of new columns - Not based on condition

2.2.1 SAS

1. In existing dataFrame - data xyz; set xyz; new_col = 'i m new'; run;
2. In new dataFrame - data abc; set xyz; new_col = 'i m new'; run;

2.2.2 Python

1. In existing dataFrame - df_have['new_col'] = 'i m new'
2. In new dataFrame - df_want = df.have.copy() ; df_want['new_col'] = 'i m new'

Note : The manipulations in python can be chained on the right hand side.

```
[10]: # Column created in Existing dataFrame

emp_df_1['Hike_amt'] = emp_df_1['Salary']*0.1
emp_df_1
```

```
[10]:    Company                        Job  Salary  Hike_amt
      0   Google  Machine Learning Engineer  123000   12300.0
      1   Google             Data Scientist  111000   11100.0
      2   Google                  Tech Lead  131000   13100.0
      3 Facebook             Data Scientist  105000   10500.0
```

```
[11]: # Creating a new dataFrame, here using copy()
emp_df_2 = emp_df_1.copy()

# Column created in new dataFrame
# Hike_amt is same for all employess - 15%

emp_df_2['Hike_amt'] = emp_df_1['Salary']*0.15
emp_df_2
emp_df_1


emp_df_2['WFH_Status'] = 1
emp_df_2['Gender'] = 'M'
emp_df_2
```

```
[11]:    Company                        Job  Salary  Hike_amt
      0   Google  Machine Learning Engineer  123000   18450.0
      1   Google             Data Scientist  111000   16650.0
      2   Google                  Tech Lead  131000   19650.0
      3 Facebook             Data Scientist  105000   15750.0
```

```
[11]:     Company                       Job  Salary  Hike_amt
      0    Google  Machine Learning Engineer  123000   12300.0
      1    Google             Data Scientist  111000   11100.0
      2    Google                  Tech Lead  131000   13100.0
      3  Facebook             Data Scientist  105000   10500.0
```

```
[11]:     Company                       Job  Salary  Hike_amt  WFH_Status Gender
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M
```

2.3 3. Creation of new columns - Based on If-then/else condition

2.3.1 .assign() and .apply()

```
[12]: emp_df_3 = emp_df_2.copy()

      # Bonus_amt is decided based on Job Role
      # DS - 20% of Salary
      # MLE - 15% of Salary
      # TL - 10% of Salary

      def bonus(row):
          if row['Job'] == 'Data Scientist':
              return row['Salary']*0.2
          elif row['Job'] == 'Machine Learning Engineer':
              return row['Salary']*0.15
          elif row['Job'] == 'Tech Lead':
              return row['Salary']*0.1

      emp_df_3a =  emp_df_3.assign(bonus_amt=emp_df_3.apply(bonus,axis=1))
      emp_df_3a =  emp_df_3a.assign(Bonus_amt=emp_df_3a.apply(bonus,axis=1))

      emp_df_3a
```

```
[12]:     Company                       Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      1    22200.0    22200.0
      2    13100.0    13100.0
      3    21000.0    21000.0
```

2.3.2 .apply - to apply user-defined function

[13]:
```
# Bonus_amt is decided based on Job Role
# DS - 20% of Salary
# MLE - 15% of Salary
# TL - 10% of Salary

def bonus(row):
    if row['Job'] == 'Data Scientist':
        return row['Salary']*0.2
    elif row['Job'] == 'Machine Learning Engineer':
        return row['Salary']*0.15
    elif row['Job'] == 'Tech Lead':
        return row['Salary']*0.1

# Creation of dataframe
_emp_df_3a = emp_df_2.copy()

_emp_df_3a['Bonus_amt'] = _emp_df_3a.apply(bonus,axis=1)
_emp_df_3a
```

[13]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   Bonus_amt
0    18450.0
1    22200.0
2    13100.0
3    21000.0
```

2.3.3 .loc()

[14]:
```
emp_df_3b = emp_df_2.copy()
emp_df_3b
# Creating a new column Bonus_amt based on Job
# DS - 20% of Salary
# MLE - 15% of Salary
# TL - 10% of Salary

emp_df_3b.loc[emp_df_3b['Job'] == 'Data Scientist', 'Bonus_amt' ] =␣
 ↪emp_df_3b['Salary']*0.2
emp_df_3b
emp_df_3b.loc[emp_df_3b['Job'] == 'Machine Learning Engineer', 'Bonus_amt' ] =␣
 ↪emp_df_3b['Salary']*0.15
```

```
emp_df_3b
emp_df_3b.loc[emp_df_3b['Job'] == 'Tech Lead', 'Bonus_amt' ] =␣
↪emp_df_3b['Salary']*0.1
emp_df_3b
```

```
[14]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M
```

```
[14]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         Bonus_amt
      0        NaN
      1    22200.0
      2        NaN
      3    21000.0
```

```
[14]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         Bonus_amt
      0    18450.0
      1    22200.0
      2        NaN
      3    21000.0
```

```
[14]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         Bonus_amt
      0    18450.0
      1    22200.0
      2    13100.0
      3    21000.0
```

2.3.4 .loc()

[15]:
```
# Updating the column Bonus_amt on which the condition is based

emp_df_3b.loc[emp_df_3b.Bonus_amt >= 20000, 'Bonus_amt'] = emp_df_3b.
↪Bonus_amt-2000
emp_df_3b
```

[15]:
```
    Company                         Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google              Data Scientist  111000   16650.0           1      M
2    Google                   Tech Lead  131000   19650.0           1      M
3  Facebook              Data Scientist  105000   15750.0           1      M

   Bonus_amt
0    18450.0
1    20200.0
2    13100.0
3    19000.0
```

2.3.5 .apply() and lambda Fx

[16]:
```
# Using lambda function to

emp_df_3b2 = emp_df_3b.copy()
emp_df_3b2['Bonus_amt'] = emp_df_3b['Bonus_amt'].apply(lambda x: x+100 if x >=␣
↪20000 else x)
emp_df_3b2
```

[16]:
```
    Company                         Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google              Data Scientist  111000   16650.0           1      M
2    Google                   Tech Lead  131000   19650.0           1      M
3  Facebook              Data Scientist  105000   15750.0           1      M

   Bonus_amt
0    18450.0
1    20300.0
2    13100.0
3    19000.0
```

2.3.6 np.where()

[17]:
```
emp_df_3b['Bonus_amt'] = np.where(emp_df_3b['Bonus_amt'] >= 20000, emp_df_3b.
↪Bonus_amt+2000, emp_df_3b.Bonus_amt)
emp_df_3b
```

```
[17]:     Company                       Job  Salary  Hike_amt  WFH_Status Gender  \
     0    Google  Machine Learning Engineer  123000   18450.0           1      M
     1    Google             Data Scientist  111000   16650.0           1      M
     2    Google                  Tech Lead  131000   19650.0           1      M
     3  Facebook             Data Scientist  105000   15750.0           1      M

        Bonus_amt
     0    18450.0
     1    22200.0
     2    13100.0
     3    19000.0
```

2.3.7 List comprehension - another way to create another column conditionally.

- https://www.listendata.com/2019/07/python-list-comprehension-with-examples.html

1. When working with object dtypes in columns, list comprehensions typically outperform most other methods.

```
[70]: emp_df_3b['Bonus_amt'] = [ x+1000 if x <= 20000 else x for x in␣
       ↪emp_df_3b['Bonus_amt'] ]
      emp_df_3b
```

```
[70]:     Company                       Job  Salary  Hike_amt  WFH_Status Gender  \
     0    Google  Machine Learning Engineer  123000   18450.0           1      M
     1    Google             Data Scientist  111000   16650.0           1      M
     2    Google                  Tech Lead  131000   19650.0           1      M
     3  Facebook             Data Scientist  105000   15750.0           1      M

        Bonus_amt
     0    20450.0
     1    22200.0
     2    15100.0
     3    21000.0
```

```
[ ]:
```

2.3.8 np.select() - If there are more than two conditions then use np.select

```
[19]: emp_df_3c = emp_df_2.copy()
      emp_df_3c

      # Creating a new column Bonus_amt based on Job
      # DS - 20% of Salary
      # MLE - 15% of Salary
      # TL - 10% of Salary

      conditions = [
          (emp_df_3c['Job'] == 'Data Scientist'),
          (emp_df_3c['Job'] == 'Machine Learning Engineer'),
          (emp_df_3c['Job'] == 'Tech Lead')]

      choices = [emp_df_3c.Salary*.20, emp_df_3c.Salary*.15, emp_df_3c.Salary*.10]

      emp_df_3c['Bonus_amt'] = np.select(conditions, choices, default=10000)

      emp_df_3c
```

[19]:

```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M
```

[19]:

```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   Bonus_amt
0    18450.0
1    22200.0
2    13100.0
3    21000.0
```

2.3.9 Use If-else condition to REPLACE MISSING VALUES with CUSTOM VALUES

[74]:

```
# Setup : for dataframe creation
emp_df_3d = emp_df_3c.copy()

# Setup : Updating few values with NaN

emp_df_3d['Bonus_amt'] = np.where(emp_df_3d['Bonus_amt'] <= 21000, np.NaN,␣
 ↪emp_df_3d['Bonus_amt'] )
emp_df_3d

# Updating the missing values (NaN) with any custom value, here 10000

emp_df_3d.loc[emp_df_3d['Bonus_amt'].isna(), 'Bonus_amt' ] = 10000
emp_df_3d

# Updating the missing values (NaN) with any custom value, here 10000
# Using fillna()
```

```
emp_df_3d.Bonus_amt.fillna(10000, inplace=True)
emp_df_3d
```

```
[74]:    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         Bonus_amt
      0        NaN
      1    22200.0
      2        NaN
      3        NaN
```

```
[74]:    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         Bonus_amt
      0    10000.0
      1    22200.0
      2    10000.0
      3    10000.0
```

```
[74]:    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         Bonus_amt
      0    10000.0
      1    22200.0
      2    10000.0
      3    10000.0
```

```
[21]: #Mean Imputation
      #Description
      #Impute the mean value at all the missing values of the column␣
       ↪'Product_Base_Margin' and then print the percentage of missing values in␣
       ↪each column.

      import numpy as np
      import pandas as pd
      df = pd.read_csv('https://query.data.world/s/Hfu_PsEuD1Z_yJHmGaxWTxvkz7W_b0')
      df.loc[df['Product_Base_Margin'].isna(), ['Product_Base_Margin']] =␣
       ↪df['Product_Base_Margin'].mean()
      print(round((100*(df.isnull().sum()/len(df.index))),2))#Round off to 2 decimal␣
       ↪places.
```

```
Ord_id                 0.00
Prod_id                0.00
Ship_id                0.00
Cust_id                0.00
Sales                  0.24
Discount               0.65
Order_Quantity         0.65
Profit                 0.65
Shipping_Cost          0.65
Product_Base_Margin    0.00
dtype: float64
```

2.4 4. Filtering out Rows from dataframe

- https://www.listendata.com/2019/07/how-to-filter-pandas-dataframe.html

1. Equivalent of IF Statement and WHERE statement/option in SAS.
2. Here, Rows are filtered out to either update the existing dataframe or create a new dataframe

i. Positional indexing : df.iloc[np.where(filter_condition)] - fast

ii. Label indexing : df.loc[df['A'].isin(['foo'])] - fast

iii. Label indexing : df.loc[filter_condition] - slow

iv. Boolean indexing : df[filter_condition] - slower ()

v. df.query() API : df.query('(col_nm >operator< value)') - slowest (for large data, the query is very efficient. More so than the standard approach)

- filter_condition, something like df.A=='foo'
- newdf = df.query('origin == "JFK" & carrier == "B6" ')

vi. Pandas dataframe.filter() function is used to Subset rows or columns of dataframe according to labels in the specified index. Note that this routine does not filter a dataframe on its contents. The filter is applied to the labels of the index. DataFrame.filter(items=None, like=None, regex=None, axis=None)[source] - df.filter ( ) : df.filter(rows or columns)

2.4.1 Boolean Indexing

[22]:
```
# Setup for dataframe creation

emp_df_4a = emp_df_3c.copy()
emp_df_4a

emp_df_4b = emp_df_3c.copy()

# Setup : Updating few 'Bonus_amt' values with NaN

emp_df_4a['Bonus_amt'] = np.where(emp_df_4a['Bonus_amt'] <= 21000, np.NaN,␣
 ↪emp_df_4a['Bonus_amt'] )
emp_df_4a

# Filtering out rows in the existing dataframe
# Here, only the rows with the non-missing Bonus_amt are retained - .notna()
emp_df_4a = emp_df_4a[emp_df_4a.Bonus_amt.notna()]
emp_df_4a

# Filtering out rows to create a new dataframe
emp_df_4b_ = emp_df_4b[(emp_df_4b.Bonus_amt > 13100) & (emp_df_4b.Hike_amt >␣
 ↪16000) ]
emp_df_4b_
```

[22]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   Bonus_amt
0    18450.0
1    22200.0
2    13100.0
3    21000.0
```

[22]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   Bonus_amt
0        NaN
1    22200.0
2        NaN
```

```
3        NaN
```

[22]:
```
  Company             Job  Salary  Hike_amt  WFH_Status Gender  Bonus_amt
1  Google  Data Scientist  111000   16650.0           1      M    22200.0
```

[22]:
```
  Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0  Google  Machine Learning Engineer  123000   18450.0           1      M
1  Google             Data Scientist  111000   16650.0           1      M

   Bonus_amt
0    18450.0
1    22200.0
```

Note: 1. The mask (boolean indexes) can be created in different ways.

```
a. Creating a series of boolean indexes : mask = df['A'] == 'foo'

b. Using the underlying numpy array and forgo the overhead of creating another pd.Series : mas

c. Instead of df[mask] we will do this : pd.DataFrame(df.values[mask], df.index[mask], df.colu

d. Using pd.Series.isin to account for each element of df['A'] being in a set of values
```

2.4.2 .iloc[ ] and np.where() - Fast

[45]:
```
# Setup for dataframe creation

emp_df_4c = emp_df_3c.copy()
emp_df_4c

emp_df_4c_1 = emp_df_4c.iloc[np.where(emp_df_4c['Bonus_amt'] >= 20000)]
emp_df_4c_1

emp_df_4c['Bonus_amt'] >= 20000
```

[45]:
```
   Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   Bonus_amt
0    18450.0
1    22200.0
2    13100.0
3    21000.0
```

```
[45]:     Company             Job  Salary  Hike_amt  WFH_Status Gender  Bonus_amt
      1    Google  Data Scientist  111000   16650.0           1      M    22200.0
      3  Facebook  Data Scientist  105000   15750.0           1      M    21000.0
```

```
[45]: 0    False
      1     True
      2    False
      3     True
      Name: Bonus_amt, dtype: bool
```

```
[45]:     Company             Job  Salary  Hike_amt  WFH_Status Gender  Bonus_amt
      1    Google  Data Scientist  111000   16650.0           1      M    22200.0
      3  Facebook  Data Scientist  105000   15750.0           1      M    21000.0
```

```
[45]:     Company             Job  Salary  Hike_amt  WFH_Status Gender  Bonus_amt
      1    Google  Data Scientist  111000   16650.0           1      M    22200.0
      3  Facebook  Data Scientist  105000   15750.0           1      M    21000.0
```

2.4.3 .loc[ ] or without .loc[ ] with .isin() - fast

```
[24]: # Setup for dataframe creation

      emp_df_4d = emp_df_3c.copy()
      emp_df_4d


      # Comapring with a list of values , use .isin()
      emp_df_4d0 = emp_df_4d.loc[emp_df_4d.Job.isin(['Tech Lead', 'Data Scientist'])]
      emp_df_4d0


      # Showing 'not in' feature through ~ and .isin()
      emp_df_4d1 = emp_df_4d[~emp_df_4d.Job.isin(['Tech Lead', 'Data Scientist'])]
      emp_df_4d1
```

```
[24]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         Bonus_amt
      0    18450.0
      1    22200.0
      2    13100.0
      3    21000.0
```

```
[24]:     Company             Job  Salary  Hike_amt  WFH_Status Gender  Bonus_amt
      1    Google  Data Scientist  111000   16650.0           1      M    22200.0
      2    Google       Tech Lead  131000   19650.0           1      M    13100.0
      3  Facebook  Data Scientist  105000   15750.0           1      M    21000.0
```

```
[24]:   Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0  Google  Machine Learning Engineer  123000   18450.0           1      M

         Bonus_amt
      0    18450.0
```

2.4.4 df. QUERY( )

- https://www.listendata.com/2020/12/how-to-use-variable-in-query-in-pandas.html

```
[67]: # Setup for dataframe creation

emp_df_4e = emp_df_3c.copy()
emp_df_4e

# Mention Value Explicitly
emp_df_4e1 = emp_df_4e.query('Bonus_amt >= 21000')
emp_df_4e1

# Mention Value Explicitly
emp_df_4e2 = emp_df_4e.query('Job == "Data Scientist"')
emp_df_4e2

# Reference Method
Bmt = 21000
emp_df_4e3 = emp_df_4e.query("Bonus_amt == @Bmt")
emp_df_4e3

# Comparing Columns
emp_df_4e['Bmt'] = emp_df_4e['Bonus_amt']
emp_df_4e.loc[emp_df_4e.Company == 'Facebook', 'Bmt' ] = 50000
emp_df_4e
emp_df_4e4 = emp_df_4e.query("Bonus_amt == Bmt")
emp_df_4e4

Bmt = 20000
column1 = 'Bonus_amt'
# to pass column name as a variable in query
emp_df_4e5 = emp_df_4e.query("{0} >= @Bmt".format(column1))
emp_df_4e5
```

```
[67]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         Bonus_amt
      0    18450.0
      1    22200.0
      2    13100.0
      3    21000.0
```

```
[67]:     Company             Job  Salary  Hike_amt  WFH_Status Gender  Bonus_amt
      1    Google  Data Scientist  111000   16650.0           1      M    22200.0
      3  Facebook  Data Scientist  105000   15750.0           1      M    21000.0
```

```
[67]:     Company             Job  Salary  Hike_amt  WFH_Status Gender  Bonus_amt
      1    Google  Data Scientist  111000   16650.0           1      M    22200.0
      3  Facebook  Data Scientist  105000   15750.0           1      M    21000.0
```

```
[67]:     Company             Job  Salary  Hike_amt  WFH_Status Gender  Bonus_amt
      3  Facebook  Data Scientist  105000   15750.0           1      M    21000.0
```

```
[67]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         Bonus_amt      Bmt
      0    18450.0  18450.0
      1    22200.0  22200.0
      2    13100.0  13100.0
      3    21000.0  50000.0
```

```
[67]:   Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0  Google  Machine Learning Engineer  123000   18450.0           1      M
      1  Google             Data Scientist  111000   16650.0           1      M
      2  Google                  Tech Lead  131000   19650.0           1      M

         Bonus_amt      Bmt
      0    18450.0  18450.0
      1    22200.0  22200.0
      2    13100.0  13100.0
```

```
[67]:     Company             Job  Salary  Hike_amt  WFH_Status Gender  Bonus_amt  \
      1    Google  Data Scientist  111000   16650.0           1      M    22200.0
```

```
3  Facebook  Data Scientist  105000   15750.0           1       M    21000.0

       Bmt
1  22200.0
3  50000.0
```

2.4.5 np.WHERE ( )

[68]:
```
# Setup for dataframe creation

emp_df_4f = emp_df_3c.copy()
emp_df_4f

# Returns Index of the satisfied condition
emp_df_4f_1 = np.where(emp_df_4c['Bonus_amt'] >= 20000)
emp_df_4f_1
```

[68]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   Bonus_amt
0    18450.0
1    22200.0
2    13100.0
3    21000.0
```

[68]:
```
(array([1, 3], dtype=int64),)
```

[ ]:
```

```

[ ]:
```

```

2.5 5. Selecting and Dropping columns

1. Equivalent of KEEP and DROP statement/option in SAS

There are 3 popular techniques of COLUMNS SELECTION, either for displaying or else for creaing a new DataFrame

i. df_name[[‘col_a’,‘col_b’,‘col_c’]] - Variants can be : df_name[columns_list]

ii. Positional indexing - df_name.iloc[: , m:n]

iii. Label Indexing - df_name.loc[: , 'col_m':'col_n'] - Variants are : df_name.loc[: , ['col_a','col_b','col_c']] : df_name.loc[: , columns_list]

iv. pd.DataFrame(df , columns = columns_list)

There are 2 popular techniques of DROPPING COLUMNS

v. df.drop('col_a') or df.drop(['col_a','col_b','col_c']) or df.drop(columns_list)

vi. df.pop('col_a')

2.5.1 i. df_name[columns_list]

[25]:
```
# Setup : DataFrame creation
emp_df_5a = emp_df_3a.copy()

emp_df_5b = emp_df_3a.copy()



# Checking no. of columns in emp_df_5a
emp_df_5a.shape[1]

# Displaying the dataframe emp_df_5a with selected columns
emp_df_5a[['Company','Job','Salary']]

# Checking the no. of columns again - Now it is still the same as actual, since␣
 ↪the previous step was only for displaying df
emp_df_5a.shape[1]
emp_df_5a

# Selecting few columns and Re-assigning to the original dataframe emp_df_5a
emp_df_5a = emp_df_5a[['Company','Job','Salary']]

# Checking the no. of columns again - Now it is reduced to 3
emp_df_5a.shape[1]
emp_df_5a



# Creating a list of column names and then passing the list as parameter
colums_list = ['Company','Job','Salary','Gender']

# Selecting few columns and Re-assigning to the original dataframe emp_df_5a
emp_df_5b = emp_df_5b[colums_list]
emp_df_5b
```

[25]:
```
8
```

```
[25]:      Company                        Job  Salary
      0     Google  Machine Learning Engineer  123000
      1     Google             Data Scientist  111000
      2     Google                  Tech Lead  131000
      3   Facebook             Data Scientist  105000
```

```
[25]: 8
```

```
[25]:      Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0     Google  Machine Learning Engineer  123000   18450.0           1      M
      1     Google             Data Scientist  111000   16650.0           1      M
      2     Google                  Tech Lead  131000   19650.0           1      M
      3   Facebook             Data Scientist  105000   15750.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      1    22200.0    22200.0
      2    13100.0    13100.0
      3    21000.0    21000.0
```

```
[25]: 3
```

```
[25]:      Company                        Job  Salary
      0     Google  Machine Learning Engineer  123000
      1     Google             Data Scientist  111000
      2     Google                  Tech Lead  131000
      3   Facebook             Data Scientist  105000
```

```
[25]:      Company                        Job  Salary Gender
      0     Google  Machine Learning Engineer  123000      M
      1     Google             Data Scientist  111000      M
      2     Google                  Tech Lead  131000      M
      3   Facebook             Data Scientist  105000      M
```

2.5.2 ii. Positional indexing - df_name.iloc[: , m:n]

```
[26]: # Setup : DataFrame creation
      emp_df_5c = emp_df_3a.copy()
      emp_df_5c
      emp_df_5d = emp_df_3a.copy()

      # Here m and n are the columns indexes. while 'n' is exclusive.
      # So, it keeps column 1 and 2
      emp_df_5c = emp_df_5c.iloc[ : , 1:3]
      emp_df_5c
```

```
emp_df_5d
# Here m and n are the columns indexes. while 'n' is EXCLUSIVE.
# So, it keeps column 1 and 2
emp_df_5d_new = emp_df_5d.iloc[ : , 1:3]
emp_df_5d_new


# emp_df_5d_new['test_col'] = 'test_value'
# emp_df_5d_new
# emp_df_5d

# emp_df_5d_new['Salary'] = 1000
# emp_df_5d_new
# emp_df_5d
```

[26]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[26]:
```
                         Job  Salary
0  Machine Learning Engineer  123000
1             Data Scientist  111000
2                  Tech Lead  131000
3             Data Scientist  105000
```

[26]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[26]:
```
                         Job  Salary
0  Machine Learning Engineer  123000
```

```
1             Data Scientist  111000
2                  Tech Lead  131000
3             Data Scientist  105000
```

2.5.3 iii. Label Indexing - df_name.loc[: , 'col_m':'col_n'] - Variants are : df_name.loc[: , ['col_a','col_b','col_c']] : df_name.loc[: , columns_list]

[27]:
```
# Setup : DataFrame creation
emp_df_5e = emp_df_3a.copy()
emp_df_5e
emp_df_5f = emp_df_3a.copy()

# Here col_m and col_n are the column names/Labels. while 'col_n' is INCLUSIVE.
# So, it keeps column 1 and 2
emp_df_5e = emp_df_5e.loc[ : , 'Company':'Hike_amt']
emp_df_5e
```

[27]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[27]:
```
    Company                        Job  Salary  Hike_amt
0    Google  Machine Learning Engineer  123000   18450.0
1    Google             Data Scientist  111000   16650.0
2    Google                  Tech Lead  131000   19650.0
3  Facebook             Data Scientist  105000   15750.0
```

2.5.4 iv. pd.DataFrame(df , columns = columns_list)

[28]:
```
# Setup : DataFrame creation
emp_df_5f = emp_df_3a.copy()
emp_df_5f

columns_list = ['Company', 'Job', 'Salary', 'Hike_amt']

# Creating a new dF with the columns list
emp_df_5f_0 = pd.DataFrame(emp_df_5f, columns= columns_list)
emp_df_5f_0
```

```
# Updating the existing dF with the columns list
emp_df_5f = pd.DataFrame(emp_df_5f, columns= columns_list)
emp_df_5f
```

```
[28]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      1    22200.0    22200.0
      2    13100.0    13100.0
      3    21000.0    21000.0
```

```
[28]:     Company                        Job  Salary  Hike_amt
      0    Google  Machine Learning Engineer  123000   18450.0
      1    Google             Data Scientist  111000   16650.0
      2    Google                  Tech Lead  131000   19650.0
      3  Facebook             Data Scientist  105000   15750.0
```

```
[28]:     Company                        Job  Salary  Hike_amt
      0    Google  Machine Learning Engineer  123000   18450.0
      1    Google             Data Scientist  111000   16650.0
      2    Google                  Tech Lead  131000   19650.0
      3  Facebook             Data Scientist  105000   15750.0
```

2.5.5 v. df.drop('col_a') or df.drop(['col_a','col_b','col_c']) or df.drop(columns_list)

DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')[source]

0. Return new object with labels in requested axis removed.
1. labels : single label or list-like - Index or column labels to drop.
2. axis : int or axis name - Whether to drop labels from the index (0 / 'index') or columns (1 / 'columns').
3. index, columns : single label or list-like
4. Alternative to specifying axis (labels, axis=1 is equivalent to columns=labels).
5. level : int or level name, default None
6. inplace : bool, default False - If True, do operation inplace and return None.
7. errors : {'ignore', 'raise'}, default 'raise' - If 'ignore', suppress error and existing labels are dropped.

```
[29]: # Setup : DataFrame creation
      emp_df_5g = emp_df_3a.copy()
```

```
emp_df_5g
emp_df_5h = emp_df_3a.copy()

# Reflecting the column drop in the existing dataframe
emp_df_5g = emp_df_5g.drop('bonus_amt', axis=1)
emp_df_5g
# The above result could also be achieved using 'inplace=True' option
# Also, error = ignore, to ignore errors in case of missing columns
columns_list = ['Company', 'Job', 'Salary', 'Hike_amt']

emp_df_5h.drop(columns_list, axis=1, inplace = True, errors = 'ignore')
emp_df_5h
```

```
[29]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      1    22200.0    22200.0
      2    13100.0    13100.0
      3    21000.0    21000.0
```

```
[29]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         Bonus_amt
      0    18450.0
      1    22200.0
      2    13100.0
      3    21000.0
```

```
[29]:    WFH_Status Gender  bonus_amt  Bonus_amt
      0           1      M    18450.0    18450.0
      1           1      M    22200.0    22200.0
      2           1      M    13100.0    13100.0
      3           1      M    21000.0    21000.0
```

Note:

1. It is important to realize that there could be various ways to get the 'columns_list' that is passed in df.drop().
2. df.loc[ ]

3. df.iloc[ ]
4. df.columns[ ]

2.5.6 vi. df.pop('col_a')

DataFrame.pop(self: ~FrameOrSeries, item) → ~FrameOrSeries[source]

0. Return item and drop from frame. Raise KeyError if not found.
1. Parameters - item : str - Label of column to be popped.
2. Returns - Series

[30]:
```
# Setup : DataFrame creation
emp_df_5i = emp_df_3a.copy()
emp_df_5i
emp_df_5j = emp_df_3a.copy()


# Only a single column can be popped out.
emp_df_5i.pop('bonus_amt')
emp_df_5i
emp_df_5j
# the popped series can be stored in a new variable and can be re-used
pop_var = emp_df_5j.pop('bonus_amt')

# new_df['new_col'] = pop_var # this will throw error, as the dataframe does␣
↪not already exist
# new_df

# Instead use pd.dataFrame, to create a new dataframe
new_df = pd.DataFrame(pop_var)
new_df

new_df['col2'] = new_df
new_df
```

[30]:
```
    Company                      Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

```
[30]: 0    18450.0
      1    22200.0
      2    13100.0
      3    21000.0
      Name: bonus_amt, dtype: float64
```

```
[30]:     Company                       Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         Bonus_amt
      0    18450.0
      1    22200.0
      2    13100.0
      3    21000.0
```

```
[30]:     Company                       Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      1    22200.0    22200.0
      2    13100.0    13100.0
      3    21000.0    21000.0
```

```
[30]:    bonus_amt
      0    18450.0
      1    22200.0
      2    13100.0
      3    21000.0
```

```
[30]:    bonus_amt     col2
      0    18450.0  18450.0
      1    22200.0  22200.0
      2    13100.0  13100.0
      3    21000.0  21000.0
```

2.6 6. Selection of rows based on their serial number in dataFrame

2.6.1 SAS Counterpart

1. Sequential - FIRSTOBS= and OBS=
2. Random Access - POINT=

2.6.2 Pandas

1. Sequential - .head( ) and .tail( ) , also df.iloc[ ] and df.loc[ ]
2. Random Access - df.iloc[ ] and df.loc[ ]

```
[31]: # Setup : DataFrame creation
emp_df_6 = emp_df_3a.copy()
emp_df_6
```

```
[31]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

```
[32]: # SEQUENTIAL ACCESS

# By default, it displays 5 rows from top
emp_df_6.head()

# It displays 3 rows from top
emp_df_6.head(3)

# By default, it displays 5 rows from bottom
emp_df_6.tail()

# It displays 3 rows from top
emp_df_6.tail(3)

# Combination of head and tail results in slicing of dF
emp_df_6.head(3).tail(1)
```

```
[32]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
```

```
3    21000.0    21000.0
```

[32]:
```
  Company                       Job  Salary  Hike_amt  WFH_Status Gender  \
0  Google  Machine Learning Engineer  123000   18450.0           1      M
1  Google             Data Scientist  111000   16650.0           1      M
2  Google                  Tech Lead  131000   19650.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
```

[32]:
```
    Company                       Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[32]:
```
    Company             Job  Salary  Hike_amt  WFH_Status Gender  bonus_amt  \
1    Google  Data Scientist  111000   16650.0           1      M    22200.0
2    Google       Tech Lead  131000   19650.0           1      M    13100.0
3  Facebook  Data Scientist  105000   15750.0           1      M    21000.0

   Bonus_amt
1    22200.0
2    13100.0
3    21000.0
```

[32]:
```
  Company        Job  Salary  Hike_amt  WFH_Status Gender  bonus_amt  \
2  Google  Tech Lead  131000   19650.0           1      M    13100.0

   Bonus_amt
2    13100.0
```

[33]:
```
# SEQUENTIAL ACCESS
emp_df_6.iloc[0:2]           # end-index in EXCLUSIVE
emp_df_6.loc[0:2]            # end-index in INCLUSIVE

# RANDOM ACCESS
emp_df_6.iloc[[2,0,3]]       # INVALID indices/labels will throw error -␣
 ↪"indices are out-of-bounds"
```

```
# Any missing label in .loc[ ] will raise KeyError
emp_df_6.loc[[2,0,3]]
```

```
[33]:   Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0  Google  Machine Learning Engineer  123000   18450.0           1      M
      1  Google             Data Scientist  111000   16650.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      1    22200.0    22200.0
```

```
[33]:   Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0  Google  Machine Learning Engineer  123000   18450.0           1      M
      1  Google             Data Scientist  111000   16650.0           1      M
      2  Google                  Tech Lead  131000   19650.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      1    22200.0    22200.0
      2    13100.0    13100.0
```

```
[33]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      2    Google                  Tech Lead  131000   19650.0           1      M
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         bonus_amt  Bonus_amt
      2    13100.0    13100.0
      0    18450.0    18450.0
      3    21000.0    21000.0
```

```
[33]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      2    Google                  Tech Lead  131000   19650.0           1      M
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         bonus_amt  Bonus_amt
      2    13100.0    13100.0
      0    18450.0    18450.0
      3    21000.0    21000.0
```

Note : 1. Passing list-likes to .loc or [] with any missing label will raise KeyError in the future, you can use .reindex() as an alternative.

2.7 7. Renaming Column names

2.7.1 i. df.rename( )

2.7.2 ii. df.columns = col_list

2.7.3 iii. df.set_axis( )

i. DataFrame.rename(self, mapper=None, index=None, columns=None, axis=None, copy=True, inplace=False, level=None, errors='ignore')[source] df.rename() - Alter axes labels.

0. Function / dict values must be unique (1-to-1). Labels not contained in a dict / Series will be left as-is. Extra labels listed don't throw an error.
1. Parameters : mapperdict-like or function : Dict-like or functions transformations to apply to that axis' values. Use either mapper and axis to specify the axis to target with mapper, or index and columns.
2. indexdict-like or function : Alternative to specifying axis (mapper, axis=0 is equivalent to index=mapper).
3. columnsdict-like or function : Alternative to specifying axis (mapper, axis=1 is equivalent to columns=mapper).
4. axisint or str : Axis to target with mapper. Can be either the axis name ('index', 'columns') or number (0, 1). The default is 'index'.
5. copybool, default True : Also copy underlying data.
6. inplacebool, default False : Whether to return a new DataFrame. If True then value of copy is ignored.
7. levelint or level name, default None : In case of a MultiIndex, only rename labels in the specified level.
8. errors{'ignore', 'raise'}, default 'ignore' : If 'raise', raise a KeyError when a dict-like mapper, index, or columns contains labels that are not present in the Index being transformed. If 'ignore', existing keys will be renamed and extra keys will be ignored.
9. Returns DataFrame - DataFrame with the renamed axis labels.
10. Raises KeyError - If any of the labels is not found in the selected axis and "errors='raise'".

[34]:
```
# Setup : DataFrame creation
emp_df_7 = emp_df_3a.copy()
emp_df_7
```

[34]:
```
    Company                       Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
```

```
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

Using AXIS style parameters

[35]:
```
emp_df_7.rename(str.lower , axis='columns')                 # All columns to␣
 ↪LOWERCASE
emp_df_7.rename(str.upper , axis='columns')                 # All columns to␣
 ↪UPPERCASE
emp_df_7.rename(str.title , axis='columns')                 # All columns to␣
 ↪TITLECASE
emp_df_7

# Include inplace = True to modify the existing dataFrame
```

[35]:
```
    company                        job  salary  hike_amt  wfh_status gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[35]:
```
    COMPANY                        JOB  SALARY  HIKE_AMT  WFH_STATUS GENDER  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   BONUS_AMT  BONUS_AMT
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[35]:
```
    Company                        Job  Salary  Hike_Amt  Wfh_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M
```

```
   Bonus_Amt  Bonus_Amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[35]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[36]:
```python
# Setup : DataFrame creation
emp_df_7a = emp_df_3a.copy()
emp_df_7a

# Requires a dictionary of old names and new names to be passed to parameter␣
↪columns=
emp_df_7a.rename(columns={'bonus_amt':'Dup_Bonus_Amt'})
emp_df_7a

# Include inplace = True to modify the existing dataFrame
```

[36]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[36]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M
```

```
   Dup_Bonus_Amt  Bonus_amt
0        18450.0    18450.0
1        22200.0    22200.0
2        13100.0    13100.0
3        21000.0    21000.0
```

```
[36]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      1    22200.0    22200.0
      2    13100.0    13100.0
      3    21000.0    21000.0
```

ii. df.columns = col_list

```
[37]: # Setup : DataFrame creation
      emp_df_7c = emp_df_3a.copy()
      emp_df_7c

      # Assign columns list into df.columns

      emp_df_7c.columns = emp_df_7c.columns.str.replace('_',' ')
      emp_df_7c
```

```
[37]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      1    22200.0    22200.0
      2    13100.0    13100.0
      3    21000.0    21000.0
```

```
[37]:     Company                        Job  Salary  Hike amt  WFH Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M
```

```
   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

iii. df.set_axis( )

DataFrame.set_axis(self, labels, axis=0, inplace=False)[source] Assign desired index to given axis. Indexes for column or row labels can be changed by assigning a list-like or Index.

1. Parameters - labelslist-like, Index : The values for the new index.
2. axis{0 or 'index', 1 or 'columns'}, default 0 :The axis to update. The value 0 identifies the rows, and 1 identifies the columns.
3. inplacebool, default False : Whether to return a new %(klass)s instance.
4. Returns - renamed%(klass)s or None : An object of same type as caller if inplace=False, None otherwise.

[38]:
```
# Setup : DataFrame creation
emp_df_7d = emp_df_3a.copy()
emp_df_7d

# Full list of labels is passed as a list in set_axis() method
emp_df_7d.set_axis(['1','2','3','4','5','6','7','8'], axis='columns', inplace =␣
↪True)
emp_df_7d
```

[38]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[38]:
```
          1                          2       3        4  5  6        7  \
0    Google  Machine Learning Engineer  123000  18450.0  1  M  18450.0
1    Google             Data Scientist  111000  16650.0  1  M  22200.0
2    Google                  Tech Lead  131000  19650.0  1  M  13100.0
3  Facebook             Data Scientist  105000  15750.0  1  M  21000.0

         8
```

```
0   18450.0
1   22200.0
2   13100.0
3   21000.0
```

2.8 8. Sorting + Removing Duplicate rows + Keeping Duplicate rows

2.8.1 SAS

1. PROC SORT - To sort on columns and return the existing sorted dataset or new dataset
2. PROC SORT (nodupkey/nodup/noduprecs) - Sort and remove duplicate records on sort keys
3. PROC SORT (dupout)- Sort and remove and store duplicate records on sort keys in a new dataset

2.8.2 Python

1. df.sort_values() - sort rows based on key columns. Inplace parameter enables to update same dF or create new dF.
2. Sorting as well as removing duplicate rows is a 2-step and independent process unlike Proc sort in SAS. But fortunately, the chaining method makes it look like a one step. df.drop_duplicates() - remove the duplicate rows based on key columns.
3. df.duplicated() - return duplicated rows

i. DataFrame.sort_values(self, by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last', ignore_index=False)[source] Sort by the values along either axis.

1. Parameters - bystr or list of str : Name or list of names to sort by.

if axis is 0 or 'index' then by may contain index levels and/or column labels. if axis is 1 or 'columns' then by may contain column levels and/or index labels. Changed in version 0.23.0: Allow specifying index or column level names. - axis{0 or 'index', 1 or 'columns'}, default 0 - Axis to be sorted.

2. ascending - bool or list of bool, default True : Sort ascending vs. descending. Specify list for multiple sort orders. If this is a list of bools, must match the length of the by.

3. inplace - bool, default False -: If True, perform operation in-place.

4. kind - {'quicksort', 'mergesort', 'heapsort'}, default 'quicksort' - mergesort is the only stable algorithm. For DataFrames, this option is only applied when sorting on a single column or label.

5. na_position - {'first', 'last'}, default 'last' : Puts NaNs at the beginning if first; last puts NaNs at the end.

6. ignore_index - bool, default False : If True, the resulting axis will be labeled 0, 1, …, n - 1.

7. Returns - sorted_objDataFrame or None : DataFrame with sorted values if inplace=False, None otherwise.

[39]:
```python
# Setup : DataFrame creation
emp_df_8a = emp_df_3a.copy()
```

```
emp_df_8a

# Sort and create a new dF. Existing dF is not sorted.
emp_df_8a1 = emp_df_8a.sort_values(['Job','Salary'], ascending = True, inplace␣
↪= False )
emp_df_8a1

# Sort and save the existing dF, using INPLACE = TRUE
emp_df_8a.sort_values(by = ['Company','Job','Salary'], ascending = True,␣
↪inplace = True )
emp_df_8a
```

[39]:

```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[39]:

```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
3  Facebook             Data Scientist  105000   15750.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
0    Google  Machine Learning Engineer  123000   18450.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M

   bonus_amt  Bonus_amt
3    21000.0    21000.0
1    22200.0    22200.0
0    18450.0    18450.0
2    13100.0    13100.0
```

[39]:

```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
3  Facebook             Data Scientist  105000   15750.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
0    Google  Machine Learning Engineer  123000   18450.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M

   bonus_amt  Bonus_amt
3    21000.0    21000.0
1    22200.0    22200.0
0    18450.0    18450.0
2    13100.0    13100.0
```

ii. DataFrame.drop_duplicates(self, subset: Union[Hashable, Sequence[Hashable], NoneType] = None, keep: Union[str, bool] = 'first', inplace: bool = False, ignore_index: bool = False) → Union[ForwardRef('DataFrame'), NoneType] Return DataFrame with duplicate rows removed. Considering certain columns is optional. Indexes, including time indexes are ignored.

1. Parameters - subset column label or sequence of labels, optional : Only consider certain columns for identifying duplicates, by default use ALL of the COLUMNS.
2. keep - {'first', 'last', False}, default 'first' - Determines which duplicates (if any) to keep.

first : Drop duplicates except for the first occurrence. last : Drop duplicates except for the last occurrence. False : Drop all duplicates.

3. inplace - bool, default False : Whether to drop duplicates in place or to return a copy.
4. ignore_index - bool, default False : If True, the resulting axis will be labeled 0, 1, …, n - 1.
5. Returns - DataFrame : DataFrame with duplicates removed or None if inplace=True.

[40]:
```python
# Setup : DataFrame creation
emp_df_8b = emp_df_3a.copy()
emp_df_8b

emp_df_8b.drop_duplicates(['Job'], keep = 'first' , inplace = False)
emp_df_8b.drop_duplicates(['Job'], keep = 'last' , inplace = False)
emp_df_8b.drop_duplicates(['Job'], keep = False , inplace = False)
```

[40]:
```
    Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0    Google  Machine Learning Engineer  123000   18450.0           1      M
1    Google             Data Scientist  111000   16650.0           1      M
2    Google                  Tech Lead  131000   19650.0           1      M
3  Facebook             Data Scientist  105000   15750.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
3    21000.0    21000.0
```

[40]:
```
  Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
0  Google  Machine Learning Engineer  123000   18450.0           1      M
1  Google             Data Scientist  111000   16650.0           1      M
2  Google                  Tech Lead  131000   19650.0           1      M

   bonus_amt  Bonus_amt
0    18450.0    18450.0
1    22200.0    22200.0
2    13100.0    13100.0
```

```
[40]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      2    13100.0    13100.0
      3    21000.0    21000.0
```

```
[40]:   Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0  Google  Machine Learning Engineer  123000   18450.0           1      M
      2  Google                  Tech Lead  131000   19650.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      2    13100.0    13100.0
```

iiii.DataFrame.duplicated(self, subset: Union[Hashable, Sequence[Hashable], NoneType] = None, keep: Union[str, bool] = 'first') → 'Series'

0. Return boolean Series denoting duplicate rows.
1. Parameters - subset column label or sequence of labels, optional : Only consider certain columns for identifying duplicates, by default use all of the columns.
2. keep - {'first', 'last', False}, default 'first' : Determines which duplicates (if any) to mark.

first : Mark duplicates as True except for the first occurrence. last : Mark duplicates as True except for the last occurrence. False : Mark all duplicates as True.

3. Returns - Series

```
[41]: # Setup : DataFrame creation
emp_df_8c = emp_df_3a.copy()
emp_df_8c

# This return a boolean series indicating whether the row is a duplicate or not
emp_df_8c.duplicated(['Job'], keep = False)

# This returns the duplicated rows
emp_df_8c_dups = emp_df_8c[emp_df_8c.duplicated(['Job'], keep = False)]
emp_df_8c_dups

# Inclusion of TILDE (~), negate the boolean values and hence return the␣
 ↪NON-duplicated rows
emp_df_8c_nodups = emp_df_8c[~emp_df_8c.duplicated(['Job'], keep = False)]
emp_df_8c_nodups
```

```
[41]:     Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0    Google  Machine Learning Engineer  123000   18450.0           1      M
      1    Google             Data Scientist  111000   16650.0           1      M
      2    Google                  Tech Lead  131000   19650.0           1      M
      3  Facebook             Data Scientist  105000   15750.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      1    22200.0    22200.0
      2    13100.0    13100.0
      3    21000.0    21000.0
```

```
[41]: 0    False
      1     True
      2    False
      3     True
      dtype: bool
```

```
[41]:     Company             Job  Salary  Hike_amt  WFH_Status Gender  bonus_amt  \
      1    Google  Data Scientist  111000   16650.0           1      M    22200.0
      3  Facebook  Data Scientist  105000   15750.0           1      M    21000.0

         Bonus_amt
      1    22200.0
      3    21000.0
```

```
[41]:   Company                        Job  Salary  Hike_amt  WFH_Status Gender  \
      0  Google  Machine Learning Engineer  123000   18450.0           1      M
      2  Google                  Tech Lead  131000   19650.0           1      M

         bonus_amt  Bonus_amt
      0    18450.0    18450.0
      2    13100.0    13100.0
```

```
[ ]:
```