Topics

- **Pandas**
- **Data Pre-Processing**
- pandas means panel data which represents the data in the form of rows and columns
- columns/default(axis=0) and rows/user(axis=1)
- pandas mainly used for 3 purposes mainly

    1. Data Analysis
        - analysing/explaining the data set in a clear manner this includes
            - basic information such as no.of columns,no.of rows,textual information,statistics of the data etc.
    2. Data Manipulation and
        - Applying some modifications /changes in data
        - merging,concatenation and join
    3. Data Cleaning
        - Removal of null data
        - dropping the unnecessary data from the dataset
- **Data Pre-Processing**
    - 1,2&3 procedures altogether called as data preprocessing.

In [1]:
```
import pandas as pd
```

In [2]:
```
pd.__version__
```

Out[2]:
```
'1.4.4'
```

In [4]:
```
data=open("salary.csv")
file=data.read() # can be read in text format( str in python)
print(file,type(file))
```

...

In [6]:
```
sal=pd.read_csv("salary.csv")
df=pd.DataFrame(sal)
df
```

...

In [7]:
```
print(df)
```

...

In [8]:
```
# basic analysis includes sample,tail,head,columns,rows
```

In [9]:
```
#while reading the file we can ignore some columns by using
#usecols=['Emp Id']
#index means ?
```

In [12]:
```
df.index
```

Out[12]:
```
RangeIndex(start=0, stop=15, step=1)
```

In [14]:
```
df.index=[v for v in range(1,16)] # you can add user index manually
df
```

...

In [15]:
```
# if you want to retrieve only specific columns ?
df['Emp Id'] # single[] it will be 1 dimensional(series)
```

...

In [17]:
```
df[['Emp Sal','Experience']]
```

...

In [18]:
```
df[['Emp Sal','Experience']].count()
# under each col we have 15 samples/records
```

Out[18]:
```
Emp Sal       15
Experience    15
dtype: int64
```

In [19]:
```
df['Emp Id'].sum() # vertical sum:sums the entire column values
```

Out[19]:
```
88700
```

In [20]:
```
# now I want the horizontal sum
# you are a st,if you want to add Total (col) that contains the total

```

In [21]:
```
#loc & iloc
# manual operations
```

In [22]:
```
df['Total']=df[['Emp Id','Emp Sal']].sum(axis=1)
df
```

...

In [29]:
```
# using loc and iloc
df.loc[3:,['Experience','Name']] # specific slicing
```
...

In [30]:
```
df.iloc[2,5:] # under sal col,from 5th record to final

```
...

In [31]:
```
df.iloc[5:,2] # what is accepting as the default
# row5 to RowFinal(3rd col) as the second index
# axis=0:col & axis=1(row)
```
...

In [32]:
```
df.info() # textual information
```
...

In [34]:
```
df.describe() # gives the statistics
```
...

In [36]:
```
df.Name.count()
```
Out[36]:
```
15
```

In [37]:
```
df['Emp Id'].sum()
```
Out[37]:
```
88700
```

In [38]:
```
df.isna() # boolean df
```
...

In [39]:
```
df.isnull()
```
...

In [41]:
```
df.isna().sum() # frequency of null data
```
...

In [42]:
```
df.isna().count()# 15 records under each columns:false outcome
```
...

```
In [44]: df.Experience.count()

Out[44]: 15
```

```
In [45]: df.Experience.value_counts()
         # counts the frequency of data items/values
```

...

```
In [46]: df.Name.value_counts()
```

...

```
In [47]: # count,value_counts,sum
```

```
In [48]: # manipulation
             - merge the similar columns
             - join the different dfs
             - concatenation of differents dfs
                 - arranging those dfs side by side
```

...

```
In [49]: "vanaja "+"    keerthana"
```

...

```
In [50]: df
```

...

```
In [58]: dic={'f':[1,2,3,4],'s':(4,5,6,7),'th':[1,2,3,8]}
         dic2={'f':[1,2,5,4],'s':(4,5,9,7),'th':[1,2,7,8]}
         df1=pd.DataFrame(dic,index=[2001,2002,2003,2004])
         df2=pd.DataFrame(dic2,index=[2001,2002,2004,2004])
         df1
```

...

```
In [59]: df2
```

...

```
In [62]: df2.merge(df1)
```

...

```
In [61]: df1.merge(df2) # merging the df2 cols into df1
```

...

In [75]:
```
df1.join(df2)
```
...

In [71]:
```
pd.concat((df1,df2)) # columns of df2 added under the columns of df1
```
...

In [76]:
```
# adding new columns to the existed file
```

In [81]:
```
#data cleaning means removal of NaN data
df.isna().value_counts()# boolean data frame
# actual df after removal of null data
```
...

In [82]:
```
df.isna()
```
...

In [87]:
```
df
```
...

In [85]:
```
df.dropna() # results in empty but why?
# by default columns
```
...

In [89]:
```
new_df=df.dropna(axis=1) # data cleaning
new_df
```
...

In [90]:
```
df
```
...

In [98]:
```
#new_df['% of increment']=
new_df['Experience'==2]=0.1*new_df['Emp Sal'] included in %increment
new_df['Experience'==3]=0.25*new_df['Emp Sal']
new_df['Experience'==4]=0.4*new_df['Emp Sal']
new_df['Experience'==5]=0.6*new_df['Emp Sal']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\Anaconda\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3628             try:
-> 3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:

~\Anaconda\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

~\Anaconda\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: False

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_26524\1399801261.py in <module>
      1 #new_df['% of increment']=
----> 2 if new_df['Experience'==2]:
      3     new_df['Emp Sal']+=0.2*new_df['Emp Sal']

~\Anaconda\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   3503             if self.columns.nlevels > 1:
   3504                 return self._getitem_multilevel(key)
-> 3505             indexer = self.columns.get_loc(key)
   3506             if is_integer(indexer):
   3507                 indexer = [indexer]

~\Anaconda\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:
-> 3631                 raise KeyError(key) from err
   3632             except TypeError:
   3633                 # If we have a listlike key, _check_indexing_error will raise

KeyError: False
```

In [ ]:
```
0.
```