In [1]:

```
import pandas as pd

df=pd.read_csv('titanic.csv')
df
```

Out[1]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Gender       891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [7]:

```
df['SibSp']=df['SibSp'].astype('float') #convert data type to float and assign to same column
```

In [9]:

```
df['changed']=df['SibSp'].astype('float')#convert data type to float and assign to a new column
```

In [15]:

```
df['Name'].astype('int') #can't enforce conversion always. Error is due to invalid conversion
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-15-88fe770625cc> in <module>
----> 1 df['Name'].astype('int')

~/.local/lib/python3.8/site-packages/pandas/core/generic.py in astype(self, dtype, copy, errors)
   5696         else:
   5697             # else, only a single dtype is given
-> 5698             new_data = self._data.astype(dtype=dtype, copy=copy, errors=errors)
   5699             return self._constructor(new_data).__finalize__(self)
   5700

~/.local/lib/python3.8/site-packages/pandas/core/internals/managers.py in astype(self, dtype, copy,
errors)
    580
    581     def astype(self, dtype, copy: bool = False, errors: str = "raise"):
--> 582         return self.apply("astype", dtype=dtype, copy=copy, errors=errors)
    583
    584     def convert(self, **kwargs):

~/.local/lib/python3.8/site-packages/pandas/core/internals/managers.py in apply(self, f, filter, **k
wargs)
    440                 applied = b.apply(f, **kwargs)
    441             else:
--> 442                 applied = getattr(b, f)(**kwargs)
    443             result_blocks = _extend_blocks(applied, result_blocks)
    444

~/.local/lib/python3.8/site-packages/pandas/core/internals/blocks.py in astype(self, dtype, copy, er
rors)
    623             vals1d = values.ravel()
    624             try:
--> 625                 values = astype_nansafe(vals1d, dtype, copy=True)
    626             except (ValueError, TypeError):
    627                 # e.g. astype_nansafe can fail on object-dtype of strings

~/.local/lib/python3.8/site-packages/pandas/core/dtypes/cast.py in astype_nansafe(arr, dtype, copy,
skipna)
    872         # work around NumPy brokenness, #1987
    873         if np.issubdtype(dtype.type, np.integer):
--> 874             return lib.astype_intsafe(arr.ravel(), dtype).reshape(arr.shape)
    875
    876         # if we have a datetime/timedelta array of objects

pandas/_libs/lib.pyx in pandas._libs.lib.astype_intsafe()

ValueError: invalid literal for int() with base 10: 'Braund, Mr. Owen Harris'
```

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Gender       891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    float64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
 12  changed      891 non-null    float64
dtypes: float64(4), int64(4), object(5)
memory usage: 90.6+ KB
```

In [18]:

```
df.isna().sum() #shows a sum of missing values in every column
```

Out[18]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Gender           0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
changed          0
dtype: int64
```

In [19]:

```
#drop missing data row-wise if any column in that row has a missing data field
df.dropna(axis=0,how='any')
```

Out[19]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	changed
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1.0	0	PC 17599	71.2833	C85	C	1.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1.0	0	113803	53.1000	C123	S	1.0
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0.0	0	17463	51.8625	E46	S	0.0
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1.0	1	PP 9549	16.7000	G6	S	1.0
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0.0	0	113783	26.5500	C103	S	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1.0	1	11751	52.5542	D35	S	1.0
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0.0	0	695	5.0000	B51 B53 B55	S	0.0
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0.0	1	11767	83.1583	C50	C	0.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0.0	0	112053	30.0000	B42	S	0.0
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0.0	0	111369	30.0000	C148	C	0.0

183 rows × 13 columns

In [20]:

```
#drop missing data column-wise if any row in that column has a missing data field
df.dropna('Cabin',how='any',axis=1,inplace=True)
df
```

Out[20]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Embarked	changed
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1.0	0	A/5 21171	7.2500	S	1.0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1.0	0	PC 17599	71.2833	C	1.0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0.0	0	STON/O2. 3101282	7.9250	S	0.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1.0	0	113803	53.1000	S	1.0
4	5	0	3	Allen, Mr. William Henry	male	35.0	0.0	0	373450	8.0500	S	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0.0	0	211536	13.0000	S	0.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0.0	0	112053	30.0000	S	0.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1.0	2	W./C. 6607	23.4500	S	1.0
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0.0	0	111369	30.0000	C	0.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0.0	0	370376	7.7500	Q	0.0

891 rows × 12 columns

In [22]:

```
df.isna().sum()
```

Out[22]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Gender           0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Embarked         2
changed          0
dtype: int64
```

In [23]:

```
df.dropna(axis=0,how='any')
```

Out[23]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Embarked	changed
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1.0	0	A/5 21171	7.2500	S	1.0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1.0	0	PC 17599	71.2833	C	1.0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0.0	0	STON/O2. 3101282	7.9250	S	0.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1.0	0	113803	53.1000	S	1.0
4	5	0	3	Allen, Mr. William Henry	male	35.0	0.0	0	373450	8.0500	S	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0.0	5	382652	29.1250	Q	0.0
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0.0	0	211536	13.0000	S	0.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0.0	0	112053	30.0000	S	0.0
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0.0	0	111369	30.0000	C	0.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0.0	0	370376	7.7500	Q	0.0

712 rows × 12 columns

In [26]:

```
#fill missing values with 0


df['Age'].fillna(value=0)
```

Out[26]:

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888     0.0
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

In [27]:

```
#use previous row's value to fill the missing value
df['Age'].fillna(method='ffill')
```

Out[27]:

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    19.0
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

In [28]:

```
#use next row's value to fill the missing value

df['Age'].fillna(method='bfill')
```

Out[28]:

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    26.0
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

In [32]:

```
#map a different value to be used for filling for each column
df.fillna( {'Age':0,'Embarked':'-'} )
```

Out[32]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Embarked	changed
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1.0	0	A/5 21171	7.2500	S	1.0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1.0	0	PC 17599	71.2833	C	1.0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0.0	0	STON/O2. 3101282	7.9250	S	0.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1.0	0	113803	53.1000	S	1.0
4	5	0	3	Allen, Mr. William Henry	male	35.0	0.0	0	373450	8.0500	S	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0.0	0	211536	13.0000	S	0.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0.0	0	112053	30.0000	S	0.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	0.0	1.0	2	W./C. 6607	23.4500	S	1.0
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0.0	0	111369	30.0000	C	0.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0.0	0	370376	7.7500	Q	0.0

891 rows × 12 columns

In [42]:

```
#convert a nested dictionary to data frame
pd.DataFrame(
    {
    0:{'name':"Harshit" , "age":24 , "gender":"male"},
    1:{'name':"Akshay" , "age":30 , "gender":"male"}
    })
```

Out[42]:

	0	1
name	Harshit	Akshay
age	24	30
gender	male	male

In [47]:

```
#transpose to interchange rows and columns if required
df1=pd.DataFrame(
    {
    "e1":{'name':"Harshit" , "age":24 , "gender":"male"},
    "e2":{'name':"Akshay" , "age":30 , "gender":"male"}
    }).T

df1.index
```

Out[47]:

```
Index(['e1', 'e2'], dtype='object')
```

In [36]:

```
#a single row requires a manual index to be provided
pd.DataFrame({'name':"Harshit" , "age":24 , "gender":"male"},index=[0])
```

Out[36]:

	name	age	gender
0	Harshit	24	male

Merge and Concat Demonstration

In [53]:

```
#creating a data frame

df1=pd.DataFrame({

    "city":['Mumbai','Dubai','LA','Chicago'],
    "temperature":[28,45,18,15],
    "humidity":[56,78,57,90]
})

df1
```

Out[53]:

	city	temperature	humidity
0	Mumbai	28	56
1	Dubai	45	78
2	LA	18	57
3	Chicago	15	90

In [54]:

```
#another one with a different value in certain positions
df2=pd.DataFrame({
    "city":['Mumbai','Dubai','LA','London'],
    "temperature":[28,45,18,10],
    "humidity":[56,78,57,67]
})

df2
```

Out[54]:

	city	temperature	humidity
0	Mumbai	28	56
1	Dubai	45	78
2	LA	18	57
3	London	10	67

In [56]:

```
#attach & append two data frames together row wise
pd.concat([df1,df2],ignore_index=True)
```

Out[56]:

	city	temperature	humidity
0	Mumbai	28	56
1	Dubai	45	78
2	LA	18	57
3	Chicago	15	90
4	Mumbai	28	56
5	Dubai	45	78
6	LA	18	57
7	London	10	67

In [60]:

```
#concat in columnar fashion if required.
#Most useful if columns from one df need to be placed next to columns of other
pd.concat([df1,df2],axis=1)
```

Out[60]:

	city	temperature	humidity	city	temperature	humidity
0	Mumbai	28	56	Mumbai	28	56
1	Dubai	45	78	Dubai	45	78
2	LA	18	57	LA	18	57
3	Chicago	15	90	London	10	67

In [61]:

```
df1
```

Out[61]:

	city	temperature	humidity
0	Mumbai	28	56
1	Dubai	45	78
2	LA	18	57
3	Chicago	15	90

In [62]:

```
df2
```

Out[62]:

	city	temperature	humidity
0	Mumbai	28	56
1	Dubai	45	78
2	LA	18	57
3	London	10	67

In [63]:

```
df1.merge(df2,on='city',how='inner') #inner join for rows associated with cities common in both df
```

Out[63]:

	city	temperature_x	humidity_x	temperature_y	humidity_y
0	Mumbai	28	56	28	56
1	Dubai	45	78	45	78
2	LA	18	57	18	57

In [65]:

```
df1.drop('humidity',axis=1,inplace=True)
df2.drop('temperature',axis=1,inplace=True)

df1
```

Out[65]:

	city	temperature
0	Mumbai	28
1	Dubai	45
2	LA	18
3	Chicago	15

In [66]:

```
df2
```

Out[66]:

	city	humidity
0	Mumbai	56
1	Dubai	78
2	LA	57
3	London	67

In [69]:

```
df1.merge(df2,on='city',how='inner') #common rows
```

Out[69]:

	city	temperature	humidity
0	Mumbai	28	56
1	Dubai	45	78
2	LA	18	57

In [70]:

```
df1.merge(df2,on='city',how='outer') #common rows
```

Out[70]:

	city	temperature	humidity
0	Mumbai	28.0	56.0
1	Dubai	45.0	78.0
2	LA	18.0	57.0
3	Chicago	15.0	NaN
4	London	NaN	67.0

In [71]:

```
df1.merge(df2,on='city',how='left') #common rows
```

Out[71]:

	city	temperature	humidity
0	Mumbai	28	56.0
1	Dubai	45	78.0
2	LA	18	57.0
3	Chicago	15	NaN

In [72]:

```
df1.merge(df2,on='city',how='right') #common rows
```

Out[72]:

	city	temperature	humidity
0	Mumbai	28.0	56
1	Dubai	45.0	78
2	LA	18.0	57
3	London	NaN	67

In [76]:

```
df1.merge(df1,on='city') #common rows
```

Out[76]:

	city	temperature_x	temperature_y
0	Mumbai	28	28
1	Dubai	45	45
2	LA	18	18
3	Chicago	15	15

In [91]:

```
#joining on indices
df1.merge(df2,left_index=True,right_index=True)
```

Out[91]:

	city_x	temperature	city_y	humidity
0	Mumbai	28	Mumbai	56
1	Dubai	45	Dubai	78
2	LA	18	LA	57
3	Chicago	15	London	67

In [92]:

```
#selecting columns while joining

# select df1.temperature,df2.humidity from df1 inner join df2 where

df1[['city','temperature']].merge(df2[['city','humidity']],on='city')
```

Out[92]:

	city	temperature	humidity
0	Mumbai	28	56
1	Dubai	45	78
2	LA	18	57

In [93]:

```
ydf=pd.read_csv('/home/harshit/Downloads/YESBANK.csv')
ydf
```

Out[93]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-01-16	334.000000	338.500000	328.000000	333.899994	319.873657	470267.0
1	2018-01-17	336.000000	343.750000	331.250000	342.500000	328.112457	653618.0
2	2018-01-18	350.000000	356.500000	333.100006	340.250000	325.956970	2419109.0
3	2018-01-19	348.000000	352.000000	339.250000	348.299988	333.668793	1659646.0
4	2018-01-22	349.000000	358.000000	349.000000	355.250000	340.326874	663569.0
...	...	...	...	...	...	...	...
484	2020-01-09	47.150002	48.450001	46.299999	47.299999	47.299999	6835915.0
485	2020-01-10	47.599998	48.349998	43.900002	44.799999	44.799999	15918973.0
486	2020-01-13	43.400002	44.000000	41.200001	42.099998	42.099998	10763969.0
487	2020-01-14	41.750000	41.750000	36.549999	38.549999	38.549999	18250917.0
488	2020-01-15	38.549999	41.099998	36.650002	39.799999	39.799999	19876620.0

489 rows × 7 columns

In [95]:

```
ydf['previous']=ydf['Volume'].shift(1) #shift al values in column downwards
```

In [97]:

```
ydf['Next']=ydf['Volume'].shift(-1) #shift values upward by 1 position
```

In [98]:

```
ydf
```

Out[98]:

	Date	Open	High	Low	Close	Adj Close	Volume	previous	Next
0	2018-01-16	334.000000	338.500000	328.000000	333.899994	319.873657	470267.0	NaN	653618.0
1	2018-01-17	336.000000	343.750000	331.250000	342.500000	328.112457	653618.0	470267.0	2419109.0
2	2018-01-18	350.000000	356.500000	333.100006	340.250000	325.956970	2419109.0	653618.0	1659646.0
3	2018-01-19	348.000000	352.000000	339.250000	348.299988	333.668793	1659646.0	2419109.0	663569.0
4	2018-01-22	349.000000	358.000000	349.000000	355.250000	340.326874	663569.0	1659646.0	659804.0
...	...	...	...	...	...	...	...	...	...
484	2020-01-09	47.150002	48.450001	46.299999	47.299999	47.299999	6835915.0	6522690.0	15918973.0
485	2020-01-10	47.599998	48.349998	43.900002	44.799999	44.799999	15918973.0	6835915.0	10763969.0
486	2020-01-13	43.400002	44.000000	41.200001	42.099998	42.099998	10763969.0	15918973.0	18250917.0
487	2020-01-14	41.750000	41.750000	36.549999	38.549999	38.549999	18250917.0	10763969.0	19876620.0
488	2020-01-15	38.549999	41.099998	36.650002	39.799999	39.799999	19876620.0	18250917.0	NaN

489 rows × 9 columns

In [104]:

```
ydf['difference']=(ydf['Volume']-ydf['previous'])/ydf['previous'] #obtain the discount
```

In [105]:

```
ydf
```

Out[105]:

	Date	Open	High	Low	Close	Adj Close	Volume	previous	Next	difference
0	2018-01-16	334.000000	338.500000	328.000000	333.899994	319.873657	470267.0	NaN	653618.0	NaN
1	2018-01-17	336.000000	343.750000	331.250000	342.500000	328.112457	653618.0	470267.0	2419109.0	0.389887
2	2018-01-18	350.000000	356.500000	333.100006	340.250000	325.956970	2419109.0	653618.0	1659646.0	2.701105
3	2018-01-19	348.000000	352.000000	339.250000	348.299988	333.668793	1659646.0	2419109.0	663569.0	-0.313943
4	2018-01-22	349.000000	358.000000	349.000000	355.250000	340.326874	663569.0	1659646.0	659804.0	-0.600174
...	...	...	...	...	...	...	...	...	...	...
484	2020-01-09	47.150002	48.450001	46.299999	47.299999	47.299999	6835915.0	6522690.0	15918973.0	0.048021
485	2020-01-10	47.599998	48.349998	43.900002	44.799999	44.799999	15918973.0	6835915.0	10763969.0	1.328726
486	2020-01-13	43.400002	44.000000	41.200001	42.099998	42.099998	10763969.0	15918973.0	18250917.0	-0.323828
487	2020-01-14	41.750000	41.750000	36.549999	38.549999	38.549999	18250917.0	10763969.0	19876620.0	0.695556
488	2020-01-15	38.549999	41.099998	36.650002	39.799999	39.799999	19876620.0	18250917.0	NaN	0.089075

489 rows × 10 columns