Import

In [ ]:

```
# Import a Pandas Library

import pandas as p
```

Load Data

In [ ]:

```
# Import a CSV File

Bike = p.read_csv('/content/drive/MyDrive/Data Set/Bike Train.csv')
```

Basic Information

In [ ]:

```
# Display Basic Information like Columns, column names and data type of the columns

Bike.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   datetime    10886 non-null  object 
 1   season      10886 non-null  int64  
 2   holiday     10886 non-null  int64  
 3   workingday  10886 non-null  int64  
 4   weather     10886 non-null  int64  
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64  
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64  
 10  registered  10886 non-null  int64  
 11  count       10886 non-null  int64  
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

In [ ]:

```
# Display No of Rows and Columns

Bike.shape
```

Out[ ]:

```
(10886, 12)
```

Row Operations

In [ ]:

```
# Fetch First 5 Rows

Bike.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

In [ ]:

```
# Fetch First 10 Rows ( Specify n in Brackets )

Bike.head(10)
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
5	2011-01-01 05:00:00	1	0	0	2	9.84	12.880	75	6.0032	0	1	1
6	2011-01-01 06:00:00	1	0	0	1	9.02	13.635	80	0.0000	2	0	2
7	2011-01-01 07:00:00	1	0	0	1	8.20	12.880	86	0.0000	1	2	3
8	2011-01-01 08:00:00	1	0	0	1	9.84	14.395	75	0.0000	1	7	8
9	2011-01-01 09:00:00	1	0	0	1	13.12	17.425	76	0.0000	8	6	14

In [ ]:

```
# Fetch Last 5 Rows

Bike.tail()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

In [ ]:

```
# Fetch Last 10 Rows ( Specify n in Brackets )

Bike.tail(10)
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
10876	2012-12-19 14:00:00	4	0	1	1	17.22	21.210	50	12.9980	33	185	218
10877	2012-12-19 15:00:00	4	0	1	1	17.22	21.210	50	19.0012	28	209	237
10878	2012-12-19 16:00:00	4	0	1	1	17.22	21.210	50	23.9994	37	297	334
10879	2012-12-19 17:00:00	4	0	1	1	16.40	20.455	50	26.0027	26	536	562
10880	2012-12-19 18:00:00	4	0	1	1	15.58	19.695	50	23.9994	23	546	569
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

In [ ]:

```
# Fetch Row Based on Index

Bike.iloc[:3]
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32

In [ ]:

```
# Fetch Row Based on Index

Bike.iloc[0:8]
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
5	2011-01-01 05:00:00	1	0	0	2	9.84	12.880	75	6.0032	0	1	1
6	2011-01-01 06:00:00	1	0	0	1	9.02	13.635	80	0.0000	2	0	2
7	2011-01-01 07:00:00	1	0	0	1	8.20	12.880	86	0.0000	1	2	3

In [ ]:

```
# Fetch Row Based on Condition

Bike[ Bike['temp'] <= 9 ].head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
7	2011-01-01 07:00:00	1	0	0	1	8.20	12.880	86	0.0000	1	2	3
48	2011-01-03 01:00:00	1	0	1	1	8.20	8.335	44	27.9993	0	2	2
49	2011-01-03 04:00:00	1	0	1	1	6.56	6.820	47	26.0027	0	1	1
50	2011-01-03 05:00:00	1	0	1	1	6.56	6.820	47	19.0012	0	3	3
51	2011-01-03 06:00:00	1	0	1	1	5.74	5.305	50	26.0027	0	30	30

In [ ]:

```
# Fetch Row Based on Double Condition

Bike[ ( Bike['temp'] <= 9 ) & ( Bike['humidity'] <= 45 ) ].head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
48	2011-01-03 01:00:00	1	0	1	1	8.20	8.335	44	27.9993	0	2	2
54	2011-01-03 09:00:00	1	0	1	1	6.56	6.820	43	26.0027	7	81	88
55	2011-01-03 10:00:00	1	0	1	1	7.38	8.335	43	16.9979	11	33	44
56	2011-01-03 11:00:00	1	0	1	1	8.20	9.090	40	22.0028	10	41	51
98	2011-01-05 07:00:00	1	0	1	1	7.38	9.090	43	12.9980	1	87	88

Column Operations

In [ ]:

```
# Display column names

Bike.columns
```

Out[ ]:

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

In [ ]:

```
# Display Single Column

Bike['datetime'].head()
```

Out[ ]:

```
0    2011-01-01 00:00:00
1    2011-01-01 01:00:00
2    2011-01-01 02:00:00
3    2011-01-01 03:00:00
4    2011-01-01 04:00:00
Name: datetime, dtype: object
```

In [ ]:

```
# Display Multiple Columns

Bike[['datetime', 'temp', 'humidity']].head()
```

Out[ ]:

	datetime	temp	humidity
0	2011-01-01 00:00:00	9.84	81
1	2011-01-01 01:00:00	9.02	80
2	2011-01-01 02:00:00	9.02	80
3	2011-01-01 03:00:00	9.84	75
4	2011-01-01 04:00:00	9.84	75

In [ ]:

```
# Create a New Column

Bike['Am I a Joke to You?'] = 0
Bike.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	Am I a Joke to You?
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	0
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	0
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	0
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	0

In [ ]:

```python
# Changing the values of the new column using the pre existing columns

Bike['Am I a Joke to You?'] = Bike['count'] - Bike['registered']
Bike.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	Am I a Joke to You?
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	3
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	8
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	5
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	3
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	0

In [ ]:

```python
# Rename Column

Bike.rename(columns = {"Am I a Joke to You?":"Laugh"}, inplace = True)
Bike.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	Laugh
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	3
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	8
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	5
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	3
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	0

In [ ]:

```python
# Drop a Column

Bike.drop(columns = {"Laugh"}, inplace = True)
Bike.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

In [ ]:

```python
# Finding if one column is equal to another

Bike['holiday'].equals(Bike['workingday'])
```

Out[ ]:

```
False
```

In [ ]:

```python
# Changing the value of the column based on some condition

Bike['holiday'] = Bike['holiday'].apply(lambda x : "Yup" if x == 1 else "Nope" )
Bike.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	Nope	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	Nope	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	Nope	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	Nope	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	Nope	0	1	9.84	14.395	75	0.0	0	1	1

In [ ]:

```python
# Changing the value of the column based on some condition defined in seperate function

def Compare(x):
  if x == 'Yup':
    return 1
  else:
    return 0
```

```python
Bike['holiday'] = Bike['holiday'].apply(Compare)
Bike.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

Statistics

In [ ]:

```python
# Descriptive Statistics for the entire Dataset

Bike.describe()
```

Out[ ]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

In [ ]:

```python
# Mean of one Column

Bike['season'].mean()
```

Out[ ]:

```
2.5066139996325556
```

In [ ]:

```python
# Median of one Column

Bike['season'].median()
```

Out[ ]:

```
3.0
```

In [ ]:

```python
# Mode of one Column

Bike['season'].mode()
```

Out[ ]:

```
0    4
dtype: int64
```

Unique() Function

In [ ]:

```python
# Find Unique values in the Column

Bike['season'].unique()
```

Out[ ]:

```
array([1, 2, 3, 4])
```

In [ ]:

```python
# Find the count of Unique values in the Column

Bike['season'].nunique()
```

Out[ ]:

```
4
```

Date & Time

In [ ]:

```
# Format the Datetime column to the right format

Bike['datetime'] = p.to_datetime(Bike['datetime'])
Bike.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  int64
 2   holiday     10886 non-null  int64
 3   workingday  10886 non-null  int64
 4   weather     10886 non-null  int64
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
dtypes: datetime64[ns](1), float64(3), int64(8)
memory usage: 1020.7 KB
```

In [ ]:

```
Bike.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

In [ ]:

```
# Create seperate Columns for Year, Month, Day, etc

Bike['Year'] = Bike['datetime'].dt.year
Bike['Month'] = Bike['datetime'].dt.month
Bike['Day'] = Bike['datetime'].dt.month
Bike['Day_Of_Week'] = Bike['datetime'].dt.dayofweek
Bike['Hour'] = Bike['datetime'].dt.hour
Bike['Minute'] = Bike['datetime'].dt.minute
Bike['Seconds'] = Bike['datetime'].dt.second

Bike.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	Year	Month	Day	Day_Of_Week	Hour	Minute	Seconds
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	2011	1	1	5	0	0	0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	2011	1	1	5	1	0	0
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2011	1	1	5	2	0	0
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	2011	1	1	5	3	0	0
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	2011	1	1	5	4	0	0

Copy

In [ ]:

```
# Copy Entire Data Set to an Object

copy = Bike.copy()
copy.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	Year	Month	Day	Day_Of_Week	Hour	Minute	Seconds
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	2011	1	1	5	0	0	0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	2011	1	1	5	1	0	0
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2011	1	1	5	2	0	0
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	2011	1	1	5	3	0	0
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	2011	1	1	5	4	0	0

In [ ]:

```
# Copy specific Columns to an Object

copy = Bike[['Year','Month','Day']].copy()
copy.head()
```

Out[ ]:

	Year	Month	Day
0	2011	1	1
1	2011	1	1
2	2011	1	1
3	2011	1	1
4	2011	1	1

Index Add/Remove

In [ ]:

```
# Set Column as an Index

Bike.set_index('datetime', inplace=True)
Bike.head()
```

Out[ ]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	Year	Month	Day	Day_Of_Week	Hour	Minute	Seconds
datetime																		
2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	2011	1	1	5	0	0	0
2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	2011	1	1	5	1	0	0
2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2011	1	1	5	2	0	0
2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	2011	1	1	5	3	0	0
2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	2011	1	1	5	4	0	0

In [ ]:

```
# Remove the Column which was set as an Index

Bike.reset_index('datetime', inplace=True)
Bike.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	Year	Month	Day	Day_Of_Week	Hour	Minute	Seconds
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	2011	1	1	5	0	0	0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	2011	1	1	5	1	0	0
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2011	1	1	5	2	0	0
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	2011	1	1	5	3	0	0
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	2011	1	1	5	4	0	0

Group By

In [ ]:

```
# Groupby a Column

Bike.groupby(['season'])['count'].sum()  # Get the total number of bicycles in each season
```

Out[ ]:

```
season
1    312498
2    588282
3    640662
4    544034
Name: count, dtype: int64
```

In [ ]:

```
# The Columns used in groupby will not become an index in the new dataframe

Bike.groupby(['season'], as_index=False)['count'].sum()
```

Out[ ]:

	season	count
0	1	312498
1	2	588282

	season	count
1	2	588282
2	3	640662
3	4	544034

In [ ]:

```
# Group by more than One Column

Bike.groupby(['season', 'weather'])['temp'].agg(['min', 'max', 'mean']) # Get Min and Max Temparature in each Season and Month
```

Out[ ]:

		min	max	mean
season	**weather**			
1	**1**	0.82	29.52	12.539147
	2	3.28	29.52	12.626853
	3	3.28	22.96	12.152322
	4	8.20	8.20	8.200000
2	**1**	9.84	38.54	23.180822
	2	9.84	34.44	22.490932
	3	9.84	33.62	21.001518
3	**1**	15.58	41.00	29.227264
	2	18.86	39.36	28.048344
	3	19.68	37.72	26.788040
4	**1**	5.74	30.34	16.235711
	2	6.56	29.52	16.970037
	3	9.84	26.24	18.626756

Sort

In [ ]:

```
# Single Sort

Bike.sort_values('count', ascending=False)[['datetime', 'count']].head()
```

Out[ ]:

	datetime	count
9345	2012-09-12 18:00:00	977
9320	2012-09-11 17:00:00	970
9297	2012-09-10 18:00:00	968
9752	2012-10-10 17:00:00	948
9896	2012-10-16 17:00:00	943

In [ ]:

```
# Multiple Sort

Bike.sort_values(['windspeed', 'temp'], ascending=False)[['datetime', 'windspeed', 'temp']].head()
```

Out[ ]:

	datetime	windspeed	temp
2755	2011-07-03 17:00:00	56.9969	32.80
2756	2011-07-03 18:00:00	56.9969	32.80
760	2011-02-15 01:00:00	51.9987	12.30
868	2011-02-19 15:00:00	50.0021	18.04
6988	2012-04-09 12:00:00	47.9988	22.14

In [ ]:

```
# Sort without making the sorting element as Index

Bike['count'].sort_values(ascending = False).reset_index().head()
```

Out[ ]:

	index	count
0	9345	977
1	9320	970
2	9297	968
3	9752	948
4	9896	943

In [ ]:

```
# Sort by making the sorting element as Index
# Sort by making the sorting element as index

Bike['count'].sort_values(ascending = False).head()
```

Out[ ]:

```
9345    977
9320    970
9297    968
9752    948
9896    943
Name: count, dtype: int64
```

Largest & Smallest

In [ ]:

```
# Find the N Largest of the specified column

Bike.nlargest(5, ['count'])[['datetime', 'count']]
```

Out[ ]:

	datetime	count
9345	2012-09-12 18:00:00	977
9320	2012-09-11 17:00:00	970
9297	2012-09-10 18:00:00	968
9752	2012-10-10 17:00:00	948
9896	2012-10-16 17:00:00	943

In [ ]:

```
# Find the N Smallest of the specified column

Bike.nsmallest(5, ['count'])[['datetime', 'count']]
```

Out[ ]:

	datetime	count
4	2011-01-01 04:00:00	1
5	2011-01-01 05:00:00	1
30	2011-01-02 07:00:00	1
49	2011-01-03 04:00:00	1
71	2011-01-04 02:00:00	1

Join & Merge

In [ ]:

```
# Create 2 new Dataframes with Sample Data

d1 = p.DataFrame({'ID': ['c1', 'c2', 'c3', 'c4', 'c5', 'c6'],
                    'Name': ['A', 'B', 'C', 'D', 'E', 'F']})
d2 = p.DataFrame({'ID': ['c2', 'c1', 'c3','c1'],
                  'Order ID': ['O2', 'O1', 'O3', 'O4']})
```

In [ ]:

```
d1.head()
```

Out[ ]:

	ID	Name
0	c1	A
1	c2	B
2	c3	C
3	c4	D
4	c5	E

In [ ]:

```
d2.head()
```

Out[ ]:

	ID	Order ID
0	c2	O2
1	c1	O1
2	c3	O3
3	c1	O4

In [ ]:

```
# Join dataframes
# Join dataframes

d1.join(d2.set_index('ID'), on='ID')
```

Out[ ]:

	ID	Name	Order ID
0	c1	A	O1
0	c1	A	O4
1	c2	B	O2
2	c3	C	O3
3	c4	D	NaN
4	c5	E	NaN
5	c6	F	NaN

In [ ]:

```
# Merge Inner

p.merge(d1, d2, how='inner', on='ID')
```

Out[ ]:

	ID	Name	Order ID
0	c1	A	O1
1	c1	A	O4
2	c2	B	O2
3	c3	C	O3

In [ ]:

```
# Merge Left

p.merge(d1, d2, how='left', on='ID')
```

Out[ ]:

	ID	Name	Order ID
0	c1	A	O1
1	c1	A	O4
2	c2	B	O2
3	c3	C	O3
4	c4	D	NaN
5	c5	E	NaN
6	c6	F	NaN

Null Values, Fillna & Dropna

In [ ]:

```
# Import a CSV File

Titanic = p.read_csv('/content/drive/MyDrive/Data Set/Titanic Train.csv')
```

In [ ]:

```
# Find number of null values in each column

Titanic.isnull().sum()
```

Out[ ]:

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

In [ ]:

```
# Find all columns with null values

Titanic.columns[Titanic.isnull().any()]
```

Out[ ]:

```
Index(['Age', 'Cabin', 'Embarked'], dtype='object')
```

In [ ]:

```
Titanic['Embarked'].mode()
```

Out[ ]:

```
0    S
dtype: object
```

In [ ]:

```
# Display the values and its no of occurance in the specified column

Titanic['Embarked'].value_counts()
```

Out[ ]:

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [ ]:

```
# Replaces nan value with another specified

Titanic['Embarked'].fillna(value='S', inplace=True) # Fillna replaces nan value with S
```

In [ ]:

```
Titanic.columns[Titanic.isnull().any()]
```

Out[ ]:

```
Index(['Age', 'Cabin'], dtype='object')
```

In [ ]:

```
# Drop rows if all columns have null values

print('Shape before drop:', Titanic.shape)
Titanic.dropna(how='all', inplace=True)
print('Shape after drop:', Titanic.shape)
```

```
Shape before drop: (891, 12)
Shape after drop: (891, 12)
```

In [ ]:

```
# Drop rows if any columns have null values

print('Shape before drop:', Titanic.shape)
Titanic.dropna(how='any', inplace=True)
print('Shape after drop:', Titanic.shape)
```

```
Shape before drop: (891, 12)
Shape after drop: (185, 12)
```

Duplicates

In [ ]:

```
d1 = p.DataFrame({
    'Brand': ['Yum Yum', 'Yum Yum', 'Indomie', 'Indomie', 'Indomie', 'Indomie'],
    'Style': ['cup', 'cup', 'cup', 'pack', 'pack', 'pack'],
    'Rating': [4, 4, 4.5, 5, 5, 4.7]
})

d1
```

Out[ ]:

	Brand	Style	Rating
0	Yum Yum	cup	4.0
1	Yum Yum	cup	4.0
2	Indomie	cup	4.5
3	Indomie	pack	5.0
4	Indomie	pack	5.0
5	Indomie	pack	4.7

In [ ]:

```
d1.drop_duplicates(keep='first')
```

Out[ ]:

	Brand	Style	Rating
0	Yum Yum	cup	4.0
2	Indomie	cup	4.5
3	Indomie	pack	5.0
5	Indomie	pack	4.7