In [1]:

```
import pandas as pd
import statsmodels.api as sm
import numpy as np
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale,StandardScaler
```

!pip install researchpy : it combines pandas,scipy.stats and statsmodels to get more complete information in single API call

In [2]:

```
data = pd.read_csv("bike_sharing.csv")
```

In [3]:

```
data.head()
```

Out[3]:

	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

Shape

In [4]:

```
data.shape
```

Out[4]:

```
(10886, 12)
```

Check the head

In [5]:

```
data.head()
```

Out[5]:

	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

Check the information

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
datetime      10886 non-null object
season        10886 non-null int64
holiday       10886 non-null int64
workingday    10886 non-null int64
weather       10886 non-null int64
temp          10886 non-null float64
atemp         10886 non-null float64
humidity      10886 non-null int64
windspeed     10886 non-null float64
casual        10886 non-null int64
registered    10886 non-null int64
count         10886 non-null int64
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.6+ KB
```

Most of values are int64 and float64 types

Only datetime is having object data type

Check the null values for each columns

In [7]:

```
data.isnull().sum()
```

Out[7]:

```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

We found that cloumns are not having null values

Correlation between temp and atemp

In [8]:

```
data['temp'].corr(data['atemp'])
```

Out[8]:

```
0.9849481104817077
```

We found that temp and atemp is highly correlated to each other that means both are proving the same information. We are dropping one of column. Dropping the DateTime for simplicity

Drop DateTime

In [9]:

```
data.drop(['datetime','atemp'],axis=1,inplace=True)
```

In [10]:

```
data.head()
```

Out[10]:

	season	holiday	workingday	weather	temp	humidity	windspeed	casual	registered	count
0	1	0	0	1	9.84	81	0.0	3	13	16
1	1	0	0	1	9.02	80	0.0	8	32	40
2	1	0	0	1	9.02	80	0.0	5	27	32
3	1	0	0	1	9.84	75	0.0	3	10	13
4	1	0	0	1	9.84	75	0.0	0	1	1

Check the Unique Values in each column

In [11]:

```
data.apply(lambda x : x.nunique())
```

Out[11]:

```
season          4
holiday         2
workingday      2
weather         4
temp           49
humidity       89
windspeed      28
casual        309
registered    731
count         822
dtype: int64
```

Check is there any duplicate

In [12]:

```
data.duplicated().sum()
```

Out[12]:

```
21
```

Remove the duplication

In [13]:

```
data.drop_duplicates(inplace=True)
```

Preparing Data for t-test

Now we will perform t-test to check whether the number of bike rentals depending on working days or not.For this we will use two sample t-test

Two sample t-test is used to check whether the means of two samples(group) are different or same.We want to check whether number of bikes rented on working days are different then number of bikes rented on non-working days

Let's check mean of bikes rented on working days or not

In [14]:

```
data.groupby('workingday')['count'].describe()
```

Out[14]:

	count	mean	std	min	25%	50%	75%	max
workingday								
0	3469.0	188.772557	173.707726	1.0	44.0	128.0	304.0	783.0
1	7396.0	193.421714	184.502349	1.0	42.0	151.0	278.0	977.0

Here, Zero indicates non-working day and 1 indicates working day. We can see that mean of rented bike is not same on working days and non-working days.

Steps for performing hypothesis testing:

1. setup Null Hypothesis Testing(H0)
2. State the alternate Hypothesis(H1)
3. Set a Significance Level(alpha)
4. calculate Test Statistics
5. Decision to accept or reject null hypothesis

create for two samples for working days and one for non-working days

In [15]:

```
sample_01 = data[data['workingday']==1]
sample_02 = data[data['workingday']==0]
```

In [16]:

```
print(sample_01.shape)
print(sample_02.shape)
```

```
(7396, 10)
(3469, 10)
```

sample_01 have 7396 observation and sample_02 have 3469 observation.We have to take equal observation for both of scenarios in both sample

In [20]:

```
sample_01 = sample_01.sample(3469)
```

In [21]:

```
print(sample_01.shape)
print(sample_02.shape)
```

```
(3469, 10)
(3469, 10)
```

Assumption for T-Test

1. The variance of the 2 samples are not equal(We will use Lavene's test to check this assumption')
2. The distribution of the residuals b/w two group follow the normal distribution.We can plot histogram and see whether distribution follows normal distribution or not. We can do Q-Q plot

Levene's test to check whether variance of two group are same

H0 : variance are same , accept the null hypothesis

H1 : variacne are not same

Alpha = 0.05%

if P-value>alpha(Cannot reject)

if P-value<=alpha(we reject the null hypothesis)

In [22]:

```
alpha = 0.05
Stats,Pvalue = stats.levene(sample_01['count'],sample_02['count'])
print(f'Test statistics:{Stats} \n Pvalue : {Pvalue} \n Alpha:{alpha}')
```

```
Test statistics:0.0035127556886156357
 Pvalue : 0.9527399537909739
 Alpha:0.05
```

In [23]:

```
if(Pvalue>alpha):
    print("We fail to reject Null Hypothesis or varince are same accept Null Hypothesis")
else:
    print("Variance are not same reject the Null Hypothesis")
```

```
We fail to reject Null Hypothesis or varince are same accept Null Hypothesis
```

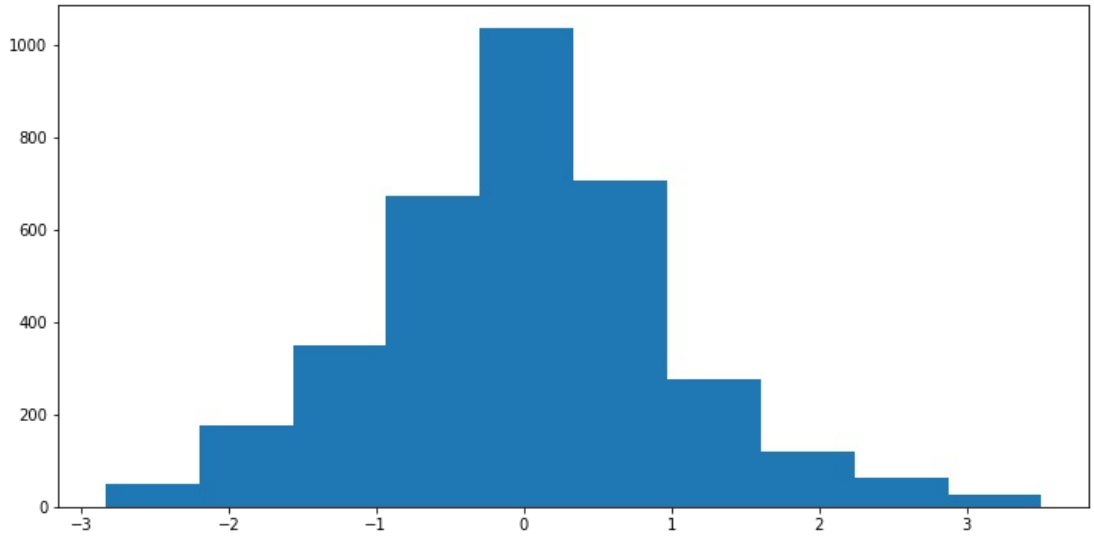
we will take difference between sample_01 and sample_02 and plot a histogram to check normality

we will scale the difference

In [26]:

```
diff = scale((np.array(sample_01['count']))-np.array(sample_02['count']))
plt.figure(figsize=(12,6))
plt.hist(diff)
plt.show()
```

```
C:\Users\Admin\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning: D
ata with input dtype int64 was converted to float64 by the scale function.
  warnings.warn(msg, DataConversionWarning)
```



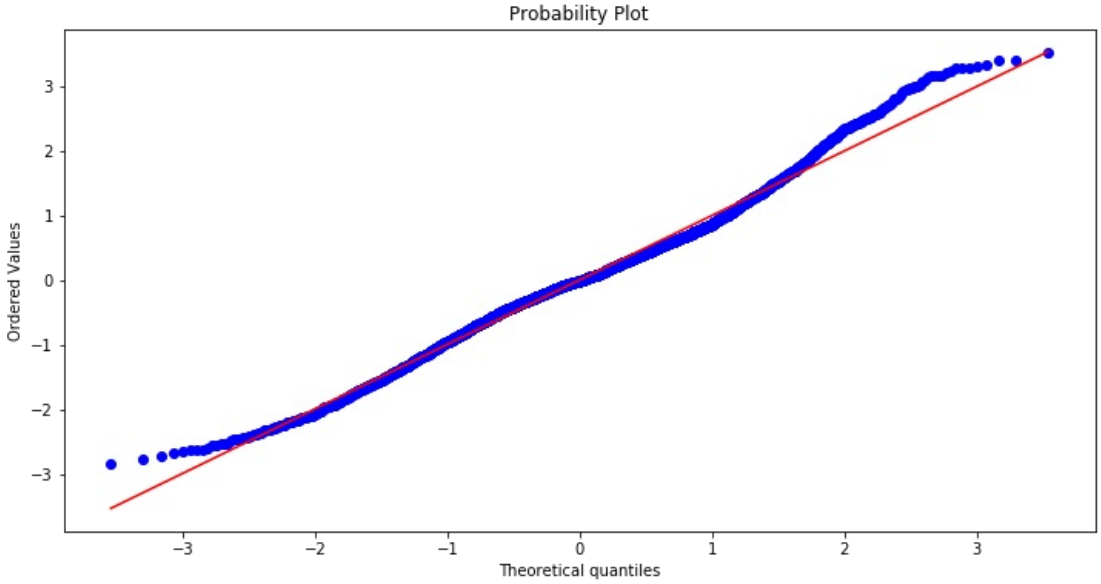
The distributions seems very close to normal distribution. Let's check other method to check normality

Q-Q plot

Q-Q plot generates the probabilty of sample data against the quantiles of theoritical distribution

In [28]:

```
plt.figure(figsize=(12,6))
stats.probplot(diff,plot=plt,dist='norm')
plt.show()
```



After 2 standard deviation the points are started scattered from the redline.They doesn't follow redline.We saw that most of data close to red line so we accept the assumption of normality