In [14]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [15]:
```
air = pd.read_csv("C:\\Users\\aman\\aman\\AirPassengers.csv1.csv")
air.head()
```

Out[15]:

	Week_num	Passengers	Promotion_Budget	Service_Quality_Score	Holiday_week	Delaye
0	1	37824	517356	4.00000	NO	
1	2	43936	646086	2.67466	NO	
2	3	42896	638330	3.29473	NO	
3	4	35792	506492	3.85684	NO	
4	5	38624	609658	3.90757	NO	

In [16]:
```
air1 = air[["Passengers","Promotion_Budget"]]
```

In [17]:
```
air1.head(2)
```

Out[17]:

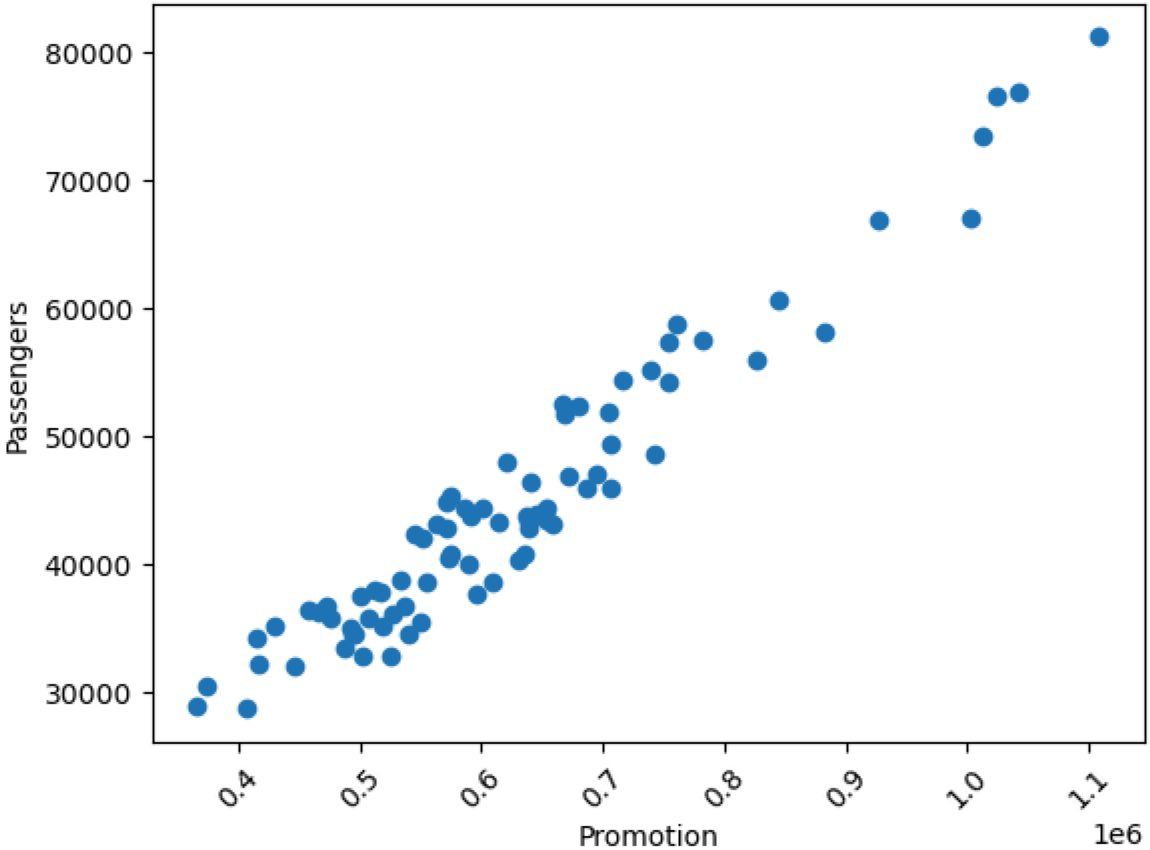
	Passengers	Promotion_Budget
0	37824	517356
1	43936	646086

objective: we would like to predict number of passengers with

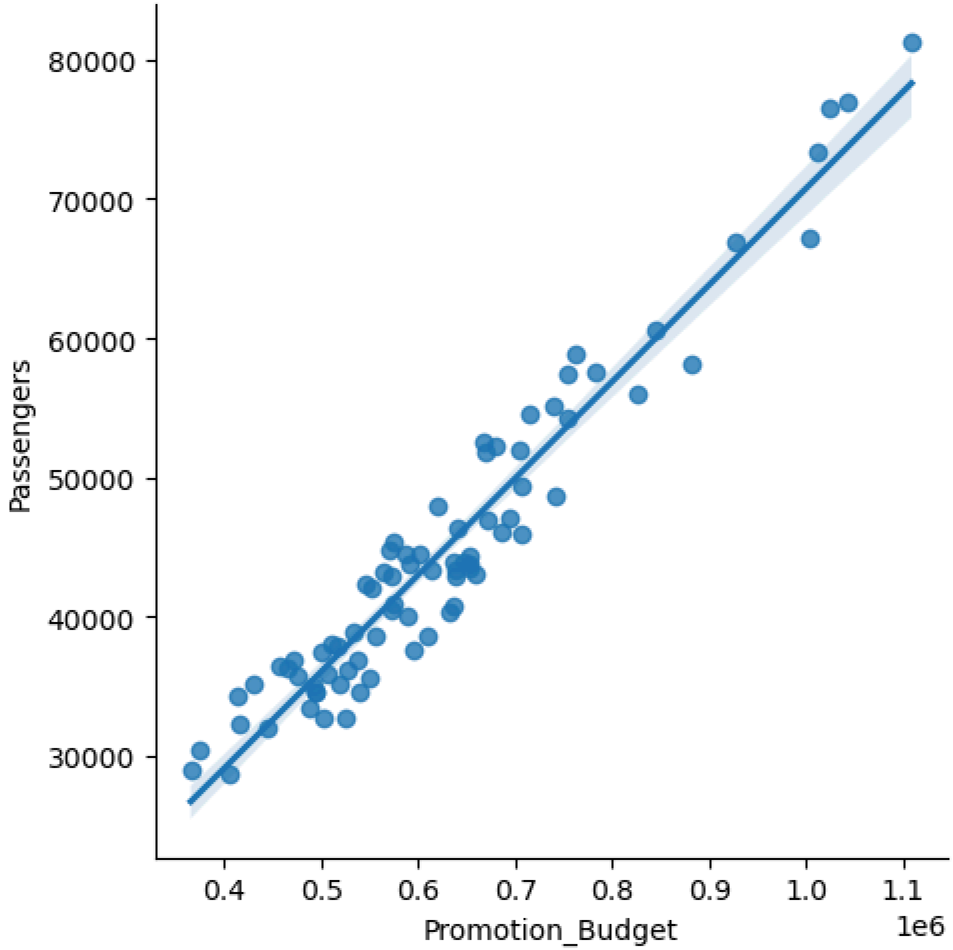
the help of promotional budget

In [18]:
```
# is passengers and promotional budget is in linear nature?
```

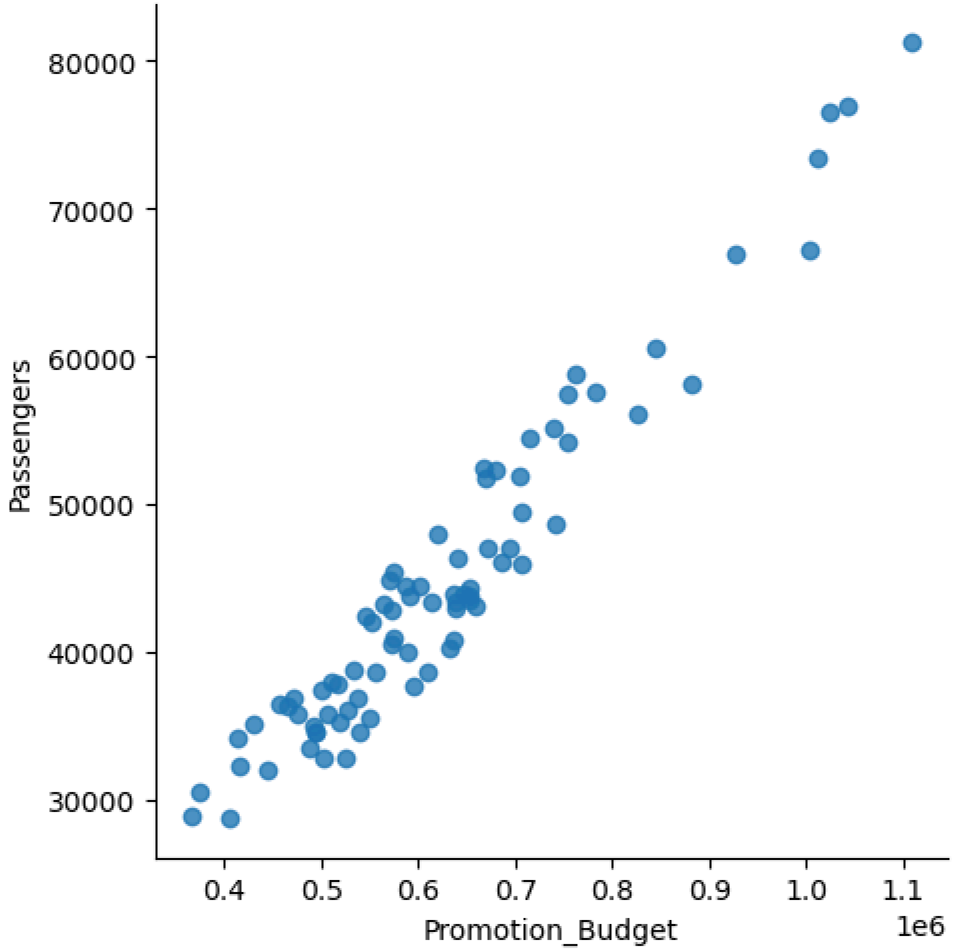
In [19]:
```
plt.scatter(air1["Promotion_Budget"], air1["Passengers"])
plt.xlabel("Promotion")
plt.ylabel("Passengers")
plt.xticks(rotation=45)
plt.show()
```



In [20]:
```
sns.lmplot(x = "Promotion_Budget", y = "Passengers", data = air)
plt.show()
```



In [21]:
```
sns.lmplot( x = "Promotion_Budget", y = "Passengers", data = air, fit_reg = False)
plt.show()
```



Slope of intercept : x and y

intercept : 1259.60583

slope : 0.06952969

In [22]:
```
# coefficent
```

summmation of x - x_ * (y-y_)/summation of x-x_**2

In [23]:
```
def coef_cal(x,y):
    a = sum((x-np.mean(x)) * (y - np.mean(y)))
    b = sum((x-np.mean(x)) ** 2)
    slope = a / b
    intercept = np.mean(y) - (slope * (np.mean(x)))
    print("Slope : ", slope)
    print("Intercept : ", intercept)
    return intercept, slope
```

In [24]:
```
coefs = coef_cal(x=air1['Promotion_Budget'], y = air1['Passengers'])
```

```
Slope :  0.06952968528865411
Intercept :  1259.6058320095253
```

Linear Regression Equation

$$Y = a + bx$$

$$a = \frac{[(\Sigma y)(\Sigma x^2) - (\Sigma y)(\Sigma xy)]}{[n(\Sigma x^2) - (\Sigma x)^2]}$$

$$b = \frac{[n(\Sigma xy) - (\Sigma x)(\Sigma y)]}{[n(\Sigma x^2) - (\Sigma x)^2]}$$

Linear Regression Equation

$$Y = a + bx$$

$$a = \frac{[(\Sigma y)(\Sigma x^2) - (\Sigma y)(\Sigma xy)]}{[n(\Sigma x^2) - (\Sigma x)^2]}$$

$$b = \frac{[n(\Sigma xy) - (\Sigma x)(\Sigma y)]}{[n(\Sigma x^2) - (\Sigma x)^2]}$$

In [25]:
```
coefs[0]
```

Out[25]: 1259.6058320095253

In [26]:
```
coefs[1]
```

Out[26]: 0.06952968528865411

In [27]:
```
#pd.set_option('display.float_format' , '{:.of}'.format)
pd.reset_option('^display',None)
```

In [28]:
```
air1['predict'] = coefs[0] + coefs[1] * air1['Promotion_Budget']
```

In [29]:
```
air1.head(10)
```

Out[29]:

	Passengers	Promotion_Budget	predict
0	37824	517356	37231.205694
1	43936	646086	46181.762081
2	42896	638330	45642.489842
3	35792	506492	36475.835193
4	38624	609658	43648.934706
5	35744	476084	34361.576523
6	40752	635978	45478.956023
7	34592	495152	35687.368562
8	35136	429800	31143.464569
9	43328	613326	43903.969591

In [30]:
```
air1['error'] = air1['Passengers'] - air1['predict']
```

In [31]:
```
air1.head()
```

Out[31]:

	Passengers	Promotion_Budget	predict	error
0	37824	517356	37231.205694	592.794306
1	43936	646086	46181.762081	-2245.762081
2	42896	638330	45642.489842	-2746.489842
3	35792	506492	36475.835193	-683.835193
4	38624	609658	43648.934706	-5024.934706

In [32]:
```
# sum of toal erros are zero because it is nullifying the positive and negative val
```

In [33]:
```
abs(round(sum(air1['error']),5))
```

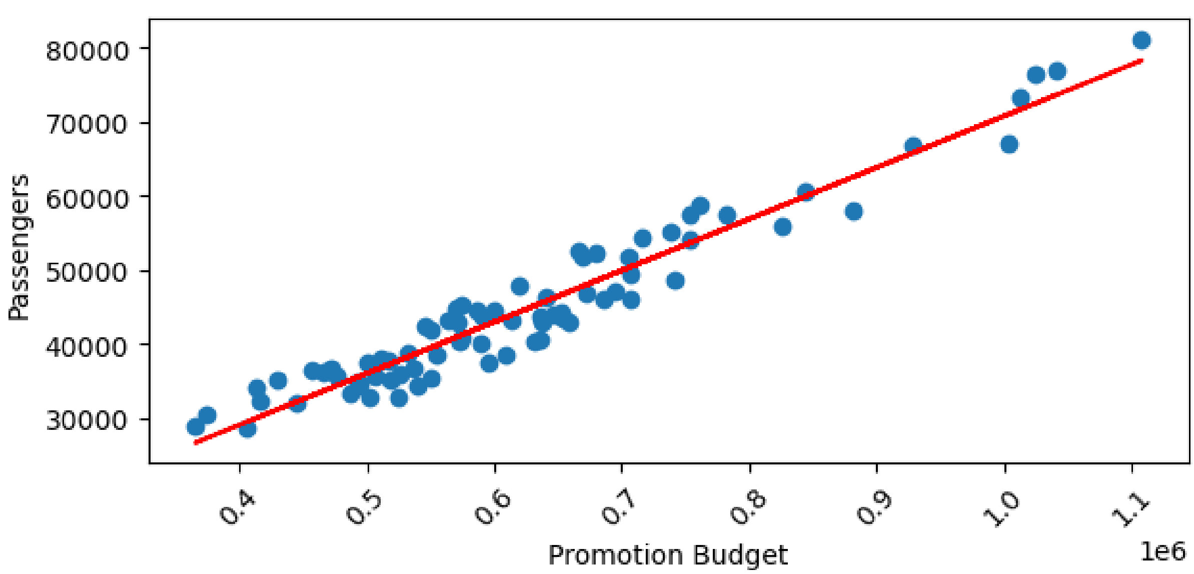
Out[33]: 0.0

In [34]:
```
air1.head(10)
```

Out[34]:

	Passengers	Promotion_Budget	predict	error
0	37824	517356	37231.205694	592.794306
1	43936	646086	46181.762081	-2245.762081
2	42896	638330	45642.489842	-2746.489842
3	35792	506492	36475.835193	-683.835193
4	38624	609658	43648.934706	-5024.934706
5	35744	476084	34361.576523	1382.423477
6	40752	635978	45478.956023	-4726.956023
7	34592	495152	35687.368562	-1095.368562
8	35136	429800	31143.464569	3992.535431
9	43328	613326	43903.969591	-575.969591

In [35]:
```
plt.figure(figsize=(7,3))
plt.scatter(air1['Promotion_Budget'], air1['Passengers'])
plt.plot(air1['Promotion_Budget'],air1['predict'], color = 'r')
plt.xlabel('Promotion Budget')
plt.ylabel('Passengers')
plt.xticks(rotation=45)
plt.show()
```



In [36]:
```
budget = 2000
pred_passengers = coefs[0] + coefs[1] * budget
```

In [37]:
```
pred_passengers
```

Out[37]:
```
1398.6652025868336
```

In [38]:
```
# how i can detect outliers
```

In [40]:
```
import statsmodels.formula.api as sm
air = pd.read_csv("C:\\Users\\aman\\aman\\AirPassengers.csv1.csv")
air1 = air[['Passengers','Promotion_Budget']].copy()
```

In [41]:
```
model = sm.ols(formula = 'Passengers~Promotion_Budget', data = air1).fit()
```

In [42]:
```
model.params
```

Out[42]:
```
Intercept           1259.605832
Promotion_Budget       0.069530
dtype: float64
```

In [43]:
```
model.summary()
```

Out[43]:

OLS Regression Results

Dep. Variable:	Passengers	**R-squared:**	0.933
Model:	OLS	**Adj. R-squared:**	0.932
Method:	Least Squares	**F-statistic:**	1084.
Date:	Sat, 01 Feb 2025	**Prob (F-statistic):**	1.66e-47
Time:	01:46:48	**Log-Likelihood:**	-751.34
No. Observations:	80	**AIC:**	1507.
Df Residuals:	78	**BIC:**	1511.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1259.6058	1361.071	0.925	0.358	-1450.078	3969.290
Promotion_Budget	0.0695	0.002	32.923	0.000	0.065	0.074

Omnibus:	26.624	**Durbin-Watson:**	1.831
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5.188
Skew:	-0.128	**Prob(JB):**	0.0747
Kurtosis:	1.779	**Cond. No.**	2.67e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.67e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

how to detect outlier?

In [44]:
```
import pandas as pd

def outlier_detection(data, x):
    q1 = data[x].quantile(0.25)  # First quartile (25th percentile)
    q3 = data[x].quantile(0.75)  # Third quartile (75th percentile)
    iqr = q3 - q1  # Interquartile range (IQR)

    lo = q1 - 1.5 * iqr  # Lower outlier boundary
    uo = q3 + 1.5 * iqr  # Upper outlier boundary

    print("Lower outlier value:", lo)
    print("Upper outlier value:", uo)
```

In [45]:
```
outlier_detection(data = air1, x = "Promotion_Budget")
```

```
Lower outlier value: 261838.5
Upper outlier value: 944646.5
```

In [46]:
```
air1.head(2)
```

Out[46]:

	Passengers	Promotion_Budget
0	37824	517356
1	43936	646086

In [47]:
```
air1.Promotion_Budget.min()
```

Out[47]: 365680

In [48]:
```
air1.Promotion_Budget.max()
```

Out[48]: 1108254

In [49]:
```
outlier_detection(data = air1, x = "Passengers")
```

```
Lower outlier value: 17764.0
Upper outlier value: 67524.0
```

In [50]:
```
air1.Passengers.min()
```

Out[50]: 28700

In [51]:
```
air1.Passengers.max()
```

Out[51]: 81228

In [52]:
```
air1['Passengers'] = np.where(air1['Passengers'] > 67524.0,67524.0,air1['Passengers
```

In [53]:
```
air1.Passengers.max()
```

Out[53]: 67524.0

In [54]:
```
air1['Promotion_Budget'] = np.where(air1['Promotion_Budget'] > 944646.5,944646.5,
                                    air1['Promotion_Budget'])
```

In [55]:
```
air1.Promotion_Budget.max()
```

Out[55]: 944646.5

linear Regression

- BO and B1 coefficient
- how to measure error and model
- what is the accuracy of model?
- what is the diffrence between parameters and its need of intercept

linear regression can be implimented into two ways:

- statemodel * import statsmodel.formula.api as sm * model = sm.ols('dependent~independent', data = dataset).fit() * model.summary() * what os ols? * ordinary least square method: in order to calc error,B0 AND B1 we need the OLS, It is used to estimate the parameters of linear regression model.

  ## points to remember:

  1. for simlpe linear regression data should be atleast 30 observation
  2. both variable(dependent and independent variables) should be numaric/ continuous.
  3. both variables are linear in nature
  4. data should not having missing and outlier value.

Points to remeber after buliding the model:

1. variance of error should be constant
2. residual error should be normally distributed.

-

In [56]:
```
model.summary()
```

Out[56]:

OLS Regression Results

Dep. Variable:	Passengers	**R-squared:**	0.933
Model:	OLS	**Adj. R-squared:**	0.932
Method:	Least Squares	**F-statistic:**	1084.
Date:	Sat, 01 Feb 2025	**Prob (F-statistic):**	1.66e-47
Time:	01:46:54	**Log-Likelihood:**	-751.34
No. Observations:	80	**AIC:**	1507.
Df Residuals:	78	**BIC:**	1511.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1259.6058	1361.071	0.925	0.358	-1450.078	3969.290
Promotion_Budget	0.0695	0.002	32.923	0.000	0.065	0.074

Omnibus:	26.624	**Durbin-Watson:**	1.831
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5.188
Skew:	-0.128	**Prob(JB):**	0.0747
Kurtosis:	1.779	**Cond. No.**	2.67e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.67e+06. This might indicate that there are strong multicollinearity or other numerical problems.

In [57]:
```
air1.shape
```

Out[57]:
```
(80, 2)
```

df_residuals : 78 (n-k-1)

df_residuals: how many data points which atre having to freely moves each other, or these are remmaing data points that you could generate new data set that look like your current dataset.

- N- number of observation
- k- independent variable

- 1- intercept

In [58]:
```
80 - 1 - 1
```

Out[58]:
```
78
```

In [59]:
```
# coveriance type: it is formula based method
```

R - Squared Means

- R2 = SSR/SST
- R2 = SSE/SST
- R2 = (SST-SSE)/SST
- R2 = SSR + SSE - SSE/ SST
- R2 = SSR/SST

In [60]:
```
model.params
```

Out[60]:
```
Intercept          1259.605832
Promotion_Budget      0.069530
dtype: float64
```

In [61]:
```
model.predict()
```

Out[61]:
```
array([37231.20569421, 46181.76208141, 45642.48984232, 36475.83519323,
       43648.93470572, 34361.57652297, 45478.95602252, 35687.36856206,
       31143.46456907, 43903.96959136, 35520.91449548, 43027.89555672,
       33031.89082151, 42010.67626095, 50264.26708282, 38594.96094146,
       52872.04745926, 36040.71842269, 40938.94569191, 58720.32834826,
       54174.47752409, 53647.85968771, 36213.01298284, 46722.00773611,
       50399.5718504 , 38291.25527612, 37280.8498895 , 38762.38842363,
       30053.23910375, 46685.99135913, 34089.99357224, 44374.12932328,
       48998.82681057, 37748.08937464, 39502.1842751 , 40439.58349217,
       45841.0666235 , 41012.92527706, 33623.72750269, 42637.55590351,
       48003.02265787, 41228.05012334, 27260.50976444, 26685.22114836,
       30209.95901439, 36827.23822268, 35117.92043954, 47059.78294724,
       45152.86179851, 41118.05416121, 42316.32875748, 49552.70028358,
       35568.61185958, 55612.21235649, 37843.48410286, 50421.96040906,
       39188.74445382, 39860.40121371, 29482.81756564, 52627.72014516,
       47620.47032941, 51007.95659667, 53714.05194811, 71632.68620321,
       70998.0192359 , 42249.16308149, 39561.56262634, 46640.24082621,
       73695.35384699, 62572.13385388, 48535.48098781, 72489.29192597,
       59982.84837373, 32229.79637202, 47784.9775648 , 65762.01675555,
       78316.1576719 , 45630.80885519, 45524.70655544, 41239.73111047])
```

In [62]:
```
air1['predict'] = model.predict()
```

In [63]:
```
air1.head()
```

Out[63]:

	Passengers	Promotion_Budget	predict
0	37824.0	517356.0	37231.205694
1	43936.0	646086.0	46181.762081
2	42896.0	638330.0	45642.489842
3	35792.0	506492.0	36475.835193
4	38624.0	609658.0	43648.934706

In [64]:
```
air1['Error_sq'] = np.square(air1['Passengers'] - air1['predict']) # SSE
```

In [65]:
```
air1.head()
```

Out[65]:

	Passengers	Promotion_Budget	predict	Error_sq
0	37824.0	517356.0	37231.205694	3.514051e+05
1	43936.0	646086.0	46181.762081	5.043447e+06
2	42896.0	638330.0	45642.489842	7.543206e+06
3	35792.0	506492.0	36475.835193	4.676306e+05
4	38624.0	609658.0	43648.934706	2.524997e+07

In [66]:
```
air1['SST'] = np.square(air1['Passengers'] - np.mean(air1['Passengers']))
```

In [67]:
```
air1.head()
```

Out[67]:

	Passengers	Promotion_Budget	predict	Error_sq	SST
0	37824.0	517356.0	37231.205694	3.514051e+05	4.153964e+07
1	43936.0	646086.0	46181.762081	5.043447e+06	1.109723e+05
2	42896.0	638330.0	45642.489842	7.543206e+06	1.885472e+06
3	35792.0	506492.0	36475.835193	4.676306e+05	7.186165e+07
4	38624.0	609658.0	43648.934706	2.524997e+07	3.186744e+07

In [68]:
```
#rsq = 1 - SSE / SST
```

In [69]:
```
air1['Error_sq'].sum() / air1['SST'].sum()
```

Out[69]: 0.10535999273399115

In [71]:
```
model.rsquared
```

Out[71]: 0.9328682090420465

CONCLUSION:

93% variance of promotional_budget is explaning / predicting the variance of passengers

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

In [72]:
```
def rsq_cal(x,y):
    a = sum((x-np.mean(x)) * (y-np.mean(y)))
    b = sum((x-np.mean(x)) ** 2)
    slope = a / b
    intercept = np.mean(y) - (slope * (np.mean(x)))
    pred = intercept + (slope * x)
    error = np.squared(y-pred)
    sst = np.squared(y-np.mean(y))
    error_sst = sum(error) / sum(sst)
    r_sq = 1 - error_sst

    print('rsquare value : ',round(r_sq,2))
    return round(r_sq,2)
```

In [73]:
```
import numpy as np

def rsq_cal(x, y):
    slope = np.cov(x, y, bias=True)[0, 1] / np.var(x)
    intercept = np.mean(y) - (slope * np.mean(x))
    pred = intercept + (slope * x)

    error = np.square(y - pred)  # Corrected line
    sst = np.square(y - np.mean(y))  # Corrected line

    error_sst = sum(error) / sum(sst)
    return 1 - error_sst  # R-squared formula
```

In [74]:
```
rsquared_value = rsq_cal(x=air1.Promotion_Budget, y=air1.Passengers)
print(rsquared_value)
```

```
0.9219349402645935
```

what is the adjusted R-Squared:

it measures the proportion of variance explained by only those independent variable that really help in explaning or helping in depending variable of a model generally Adjusted R2 not considerd in simple linear regression it is helpful when you are working on multiple linear regression

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

Where

R^2 Sample R-Squared

N Total Sample Size

p Number of independent variable

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

Where

R^2 Sample R-Squared

N Total Sample Size

p Number of independent variable

In [75]:
```
rsquared_value
```

Out[75]: 0.9219349402645935

In [76]:
```
adj_rsq = 1 - (((1 - rsquared_value) * (len(air1.Promotion_Budget) - 1)) / (78))
```

In [77]:
```
round(adj_rsq,2)
```

Out[77]: 0.92

F_Stat :

To measure the goodness fit for a model, we can not make a decision based on a single f-stat value

formula of F_Stat:

F = MRS/MSE

- MSR - mean square regression

- MSE - mean square error
- MSR = SSR/K-> K is the independent variable
- MSE = SSE / N - K - 1

In [78]:
```
air1['SSR'] = air1['SST'] - air1['Error_sq']
```

In [79]:
```
air1.head()
```

Out[79]:

	Passengers	Promotion_Budget	predict	Error_sq	SST	SSI
0	37824.0	517356.0	37231.205694	3.514051e+05	4.153964e+07	4.118823e+0
1	43936.0	646086.0	46181.762081	5.043447e+06	1.109723e+05	-4.932475e+0
2	42896.0	638330.0	45642.489842	7.543206e+06	1.885472e+06	-5.657734e+0
3	35792.0	506492.0	36475.835193	4.676306e+05	7.186165e+07	7.139402e+0
4	38624.0	609658.0	43648.934706	2.524997e+07	3.186744e+07	6.617467e+0

In [80]:
```
MSR = sum(air1['SSR'])
```

In [81]:
```
MSR
```

Out[81]:
```
7052664294.553731
```

In [82]:
```
MSE = sum(air1['Error_sq'] / (78))
```

In [83]:
```
MSE
```

Out[83]:
```
10648440.92559319
```

In [84]:
```
F = round(MSR / MSE,0)
```

In [85]:
```
F
```

Out[85]:
```
662.0
```

SUMMARY:

Fstat,adj_R2 : metrics to considered while interpreting the model.

for simple linear regression model:

- r2
- jb test
- skew

- Coefs
- P-value
- Durbin-weston test note: in simple linear regression F-stat, adj-R2 is not useful.

In [86]:
```
model.summary()
```

Out[86]:

OLS Regression Results

Dep. Variable:	Passengers	**R-squared:**	0.933
Model:	OLS	**Adj. R-squared:**	0.932
Method:	Least Squares	**F-statistic:**	1084.
Date:	Sat, 01 Feb 2025	**Prob (F-statistic):**	1.66e-47
Time:	01:47:11	**Log-Likelihood:**	-751.34
No. Observations:	80	**AIC:**	1507.
Df Residuals:	78	**BIC:**	1511.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1259.6058	1361.071	0.925	0.358	-1450.078	3969.290
Promotion_Budget	0.0695	0.002	32.923	0.000	0.065	0.074

Omnibus:	26.624	**Durbin-Watson:**	1.831
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5.188
Skew:	-0.128	**Prob(JB):**	0.0747
Kurtosis:	1.779	**Cond. No.**	2.67e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.67e+06. This might indicate that there are strong multicollinearity or other numerical problems.

F-statistic : p value:

- p <= 0.05 : reject the null hypothesis
- null hypothesis (HO) : model is not good fit
- Alternative hypothesis(HA) : model is good fit.

AIC : Akaiye information critaria

BIC : Bayseain information critaria

stand alone aic and bic does not make any sence when we comparing multiple models, we use F-stat along with we can use the aic and bic. AIC Formula is = -2 * LL +2K(HERE , K IS COUNT OF independent variable + count of intercept of given data)

- LL: LOG- liklihood AIC : how much amount of information we are loosing from the models. lesseer the AIC scores explain better model

In [87]:
```
AIC = -2 * -751.34 + (2*2)
```

In [88]:
```
AIC
```

Out[88]: 1506.68

In [89]:
```
round(AIC,0)
```

Out[89]: 1507.0

- AIC scores useful only when it is used to compare the model:
- examples : we have two models, such as : k1 and k2 :(K1 : AIC1 : K2 : AIC2)
- K1 : AIC1 < AIC2 => i.e. Model 1 is better than model 2.

BIC

IT is also used for model selection among a finite set of model. The value of BIC is very close to AIC value. BIC = -2 * LL(n) *(k)

- Ln: Natural log
- n : number of ovservation
- k : Number od independent + intercept count
-

In [90]:
```
round(-2 * -751.34 + np.log(80) * 2,0)
```

Out[90]: 1511.0

Intercept 1259.605832 Promotion_Budget 0.069530 dtype: float64 y^ = Wo+W1x

W1 : Slope : 0.069 : Passengers are expected to be increase by 0.069 for each unit of promotiaonl budget.

W0 : Intercept: 1259 : Average number of passengers are expected to be 1259, when there is no promotional budget or promotional budget is zero.

In [91]:
```
model.summary()
```

Out[91]:

OLS Regression Results

Dep. Variable:	Passengers	**R-squared:**	0.933
Model:	OLS	**Adj. R-squared:**	0.932
Method:	Least Squares	**F-statistic:**	1084.
Date:	Sat, 01 Feb 2025	**Prob (F-statistic):**	1.66e-47
Time:	01:48:03	**Log-Likelihood:**	-751.34
No. Observations:	80	**AIC:**	1507.
Df Residuals:	78	**BIC:**	1511.
Df Model:	1		
Covariance Type:	nonrobust		

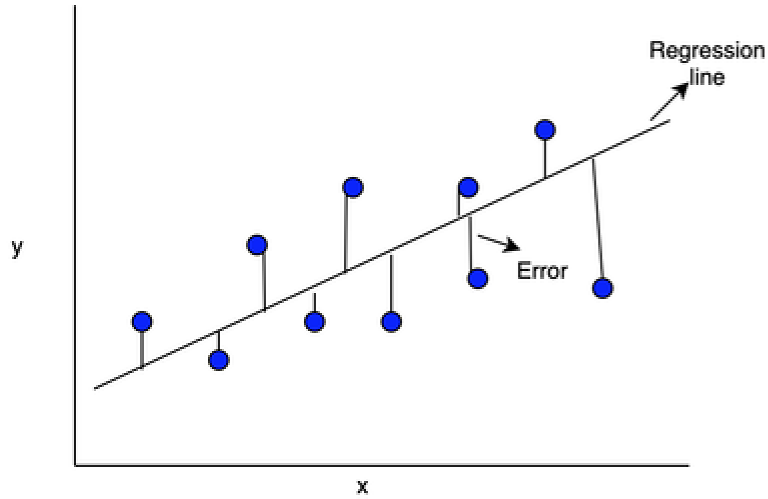
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1259.6058	1361.071	0.925	0.358	-1450.078	3969.290
Promotion_Budget	0.0695	0.002	32.923	0.000	0.065	0.074

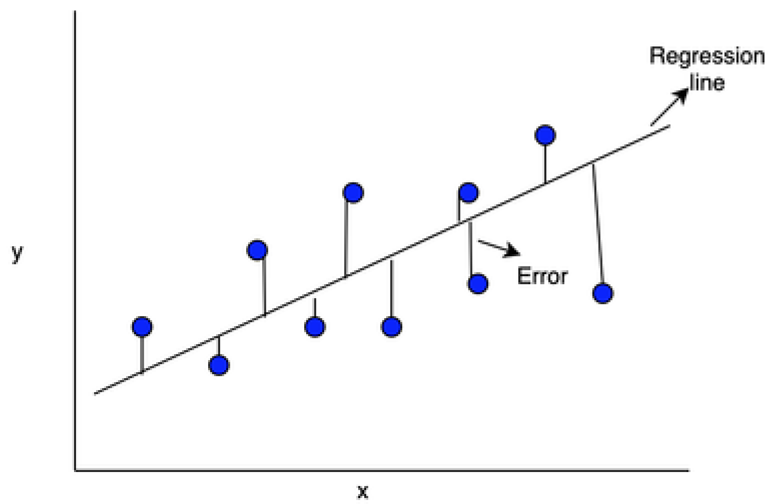
Omnibus:	26.624	**Durbin-Watson:**	1.831
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5.188
Skew:	-0.128	**Prob(JB):**	0.0747
Kurtosis:	1.779	**Cond. No.**	2.67e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.67e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

standard error:

it represents the average distance that the observed (actual) value fall from the regression. distance between line to actual data point that repersent the standard error.





In [92]:
```
1259.6058/1361.071
```

Out[92]: 0.9254519418898794

In [93]:
```
0.0695/0.002
```

Out[93]: 34.75

Omnibus: 26.624 Durbin-Watson: 1.831 Prob(Omnibus): 0.000 Jarque-Bera (JB): 5.188

Skewness

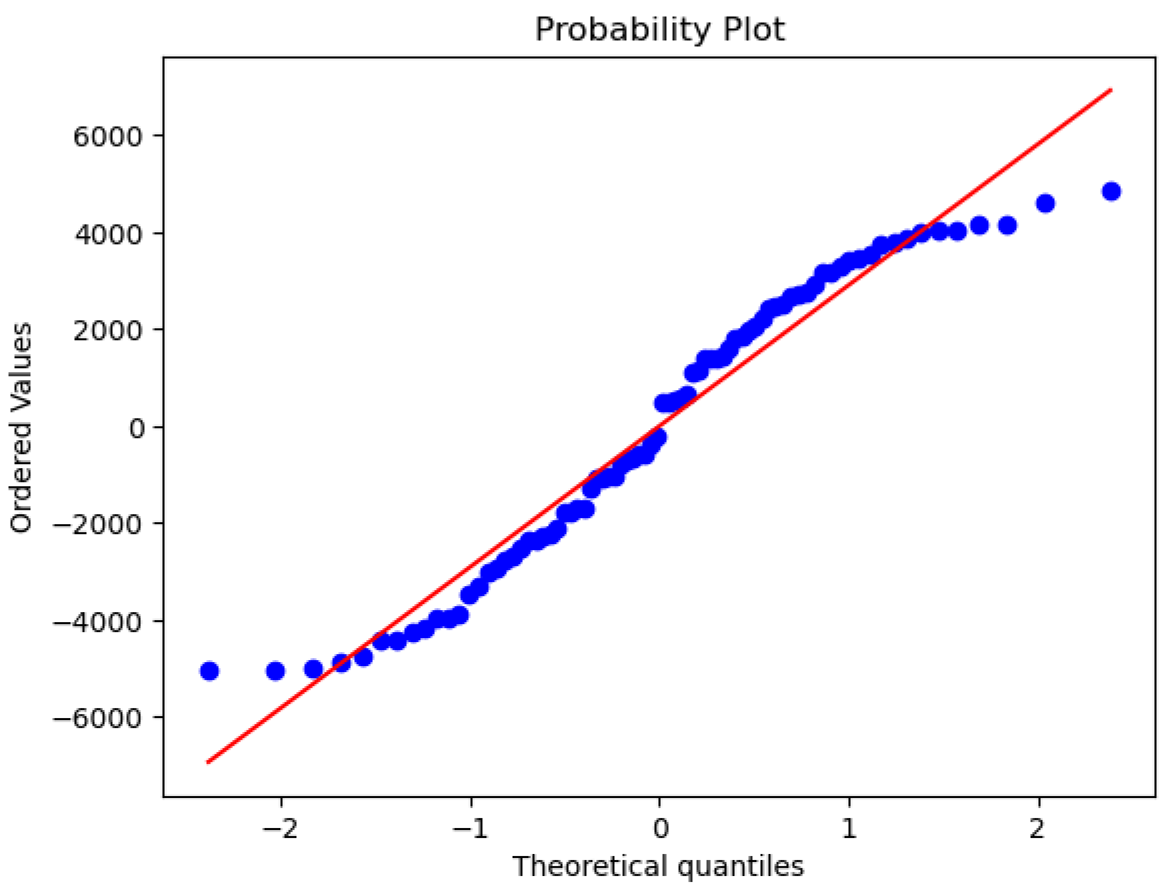
Weather the reidual error are normally distributed or not

-0.5 to 0.5 : Fairly symmetrical data +>1 or <-1 : Highly skewed data

In [94]:
```
model.resid
```

Out[94]:
```
0      592.794306
1    -2245.762081
2    -2746.489842
3     -683.835193
4    -5024.934706
         ...
75    1171.983244
76    2911.842328
77   -2342.808855
78   -1690.706555
79    -387.731110
Length: 80, dtype: float64
```

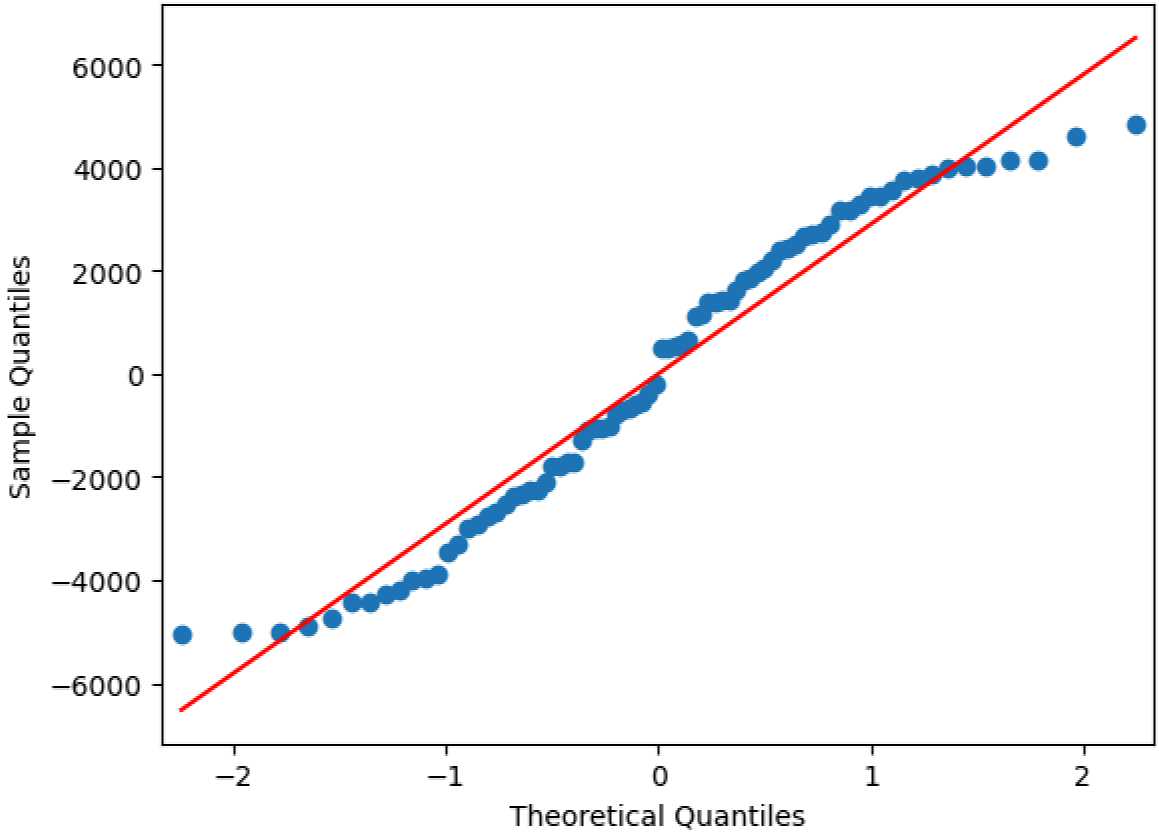
In [97]:
```
from scipy import stats
stats.probplot(model.resid, plot = plt)
plt.show()
```

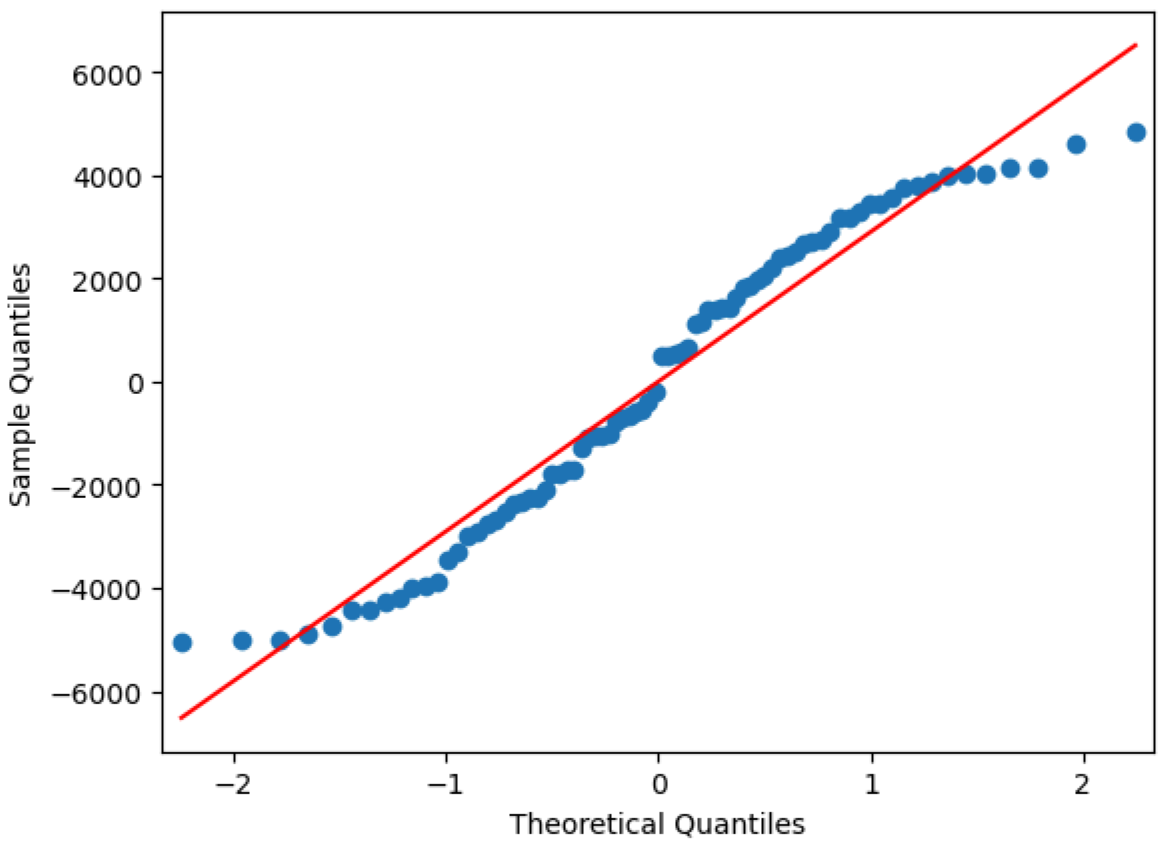


In [98]:
```
from statsmodels.graphics.gofplots import qqplot
```

In [99]:
```
qqplot(model.resid,line = 's') # s: strate line of the give data
```

Out[99]:

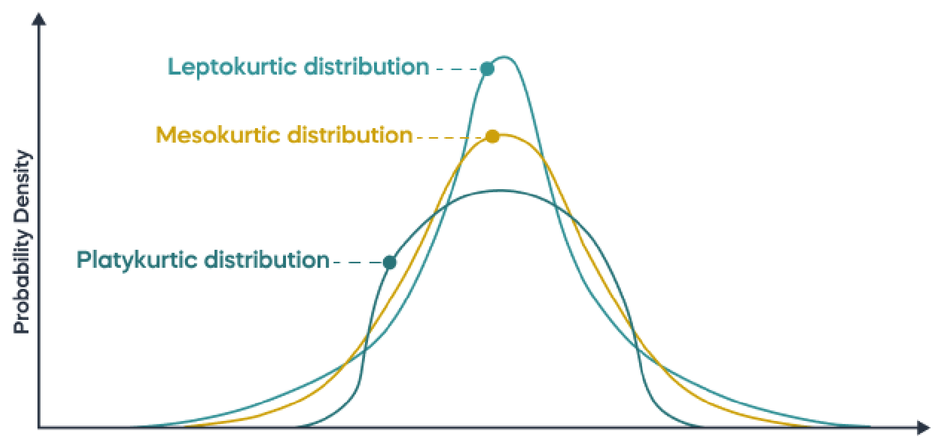


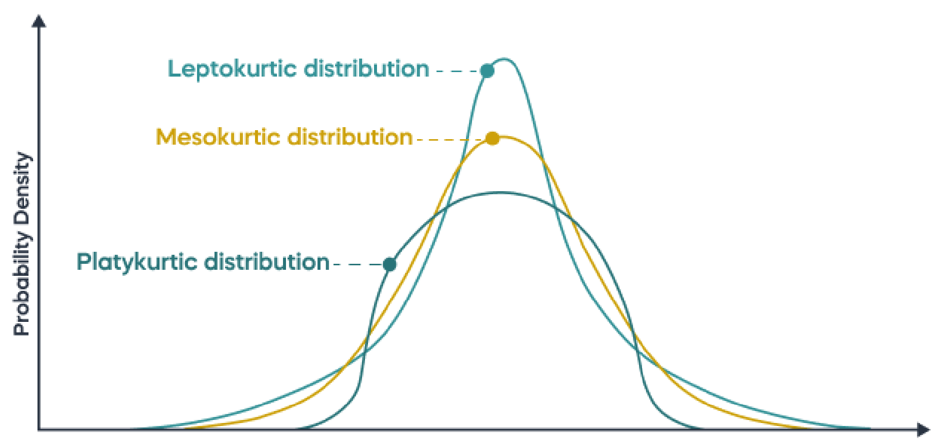


What is the meaning of Kurtosis:

It is a measure of the tailness of the probability distribution.

K = 3 : Normal distribution : Mesokurtic K < 3 : The tail are faraway from the mean value and highly likely chance outlier will be residual error. Paltykurtic K > 3 : Flat data and seems most of the data points normal distributed. Leptokurtic Range of Kurtosis for normally distributed: 1.5 to 3.5 (In book : 2





to 4).

Jarque_Bera Statistics test:

JB also check the normal distribution.

A JB close to 0 means that data is close to normal, and larger value indicates a deviation for normamlity. Typically, A higher test statistic means that the data is more likely to be non-normal

In [100…
```
from scipy import stats
```

In [101…
```
stats.jarque_bera(model.resid)
```

Out[101…

```
SignificanceResult(statistic=5.188189975966197, pvalue=0.07471346206775015)
```

In [102…

```
model.summary()
```

Out[102…

OLS Regression Results

Dep. Variable:	Passengers	**R-squared:**	0.933
Model:	OLS	**Adj. R-squared:**	0.932
Method:	Least Squares	**F-statistic:**	1084.
Date:	Sat, 01 Feb 2025	**Prob (F-statistic):**	1.66e-47
Time:	02:00:15	**Log-Likelihood:**	-751.34
No. Observations:	80	**AIC:**	1507.
Df Residuals:	78	**BIC:**	1511.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1259.6058	1361.071	0.925	0.358	-1450.078	3969.290
Promotion_Budget	0.0695	0.002	32.923	0.000	0.065	0.074

Omnibus:	26.624	**Durbin-Watson:**	1.831
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5.188
Skew:	-0.128	**Prob(JB):**	0.0747
Kurtosis:	1.779	**Cond. No.**	2.67e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.67e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]: