Import libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing # SES
from statsmodels.tsa.holtwinters import Holt # Holts Exponential Smoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_excel('CocaCola_Sales_Rawdata.xlsx')
data
```

Out[2]:

	Quarter	Sales
0	Q1_86	1734.827000
1	Q2_86	2244.960999
2	Q3_86	2533.804993
3	Q4_86	2154.962997
4	Q1_87	1547.818996
5	Q2_87	2104.411995
6	Q3_87	2014.362999
7	Q4_87	1991.746998
8	Q1_88	1869.049999
9	Q2_88	2313.631996
10	Q3_88	2128.320000
11	Q4_88	2026.828999
12	Q1_89	1910.603996
13	Q2_89	2331.164993
14	Q3_89	2206.549995
15	Q4_89	2173.967995
16	Q1_90	2148.278000
17	Q2_90	2739.307999
18	Q3_90	2792.753998
19	Q4_90	2556.009995
20	Q1_91	2480.973999
21	Q2_91	3039.522995
22	Q3_91	3172.115997
23	Q4_91	2879.000999
24	Q1_92	2772.000000
25	Q2_92	3550.000000
26	Q3_92	3508.000000
27	Q4_92	3243.859993
28	Q1_93	3056.000000
29	Q2_93	3899.000000
30	Q3_93	3629.000000
31	Q4_93	3373.000000
32	Q1_94	3352.000000

	Quarter	Sales
33	Q2_94	4342.000000
34	Q3_94	4461.000000
35	Q4_94	4017.000000
36	Q1_95	3854.000000
37	Q2_95	4936.000000
38	Q3_95	4895.000000
39	Q4_95	4333.000000
40	Q1_96	4194.000000
41	Q2_96	5253.000000

In [3]:
```
data.shape
```

Out[3]: (42, 2)

In [4]:
```
# We will extract Quarter values and Year values separately from 'Quarter column
data['Quarters'] = 0
data['Year'] = 0
for i in range(42):
    p =data["Quarter"][i]
    data['Quarters'][i]= p[0:2]
    data['Year'][i]= p[3:5]
```

In [5]:
```
data.head()
```

Out[5]:

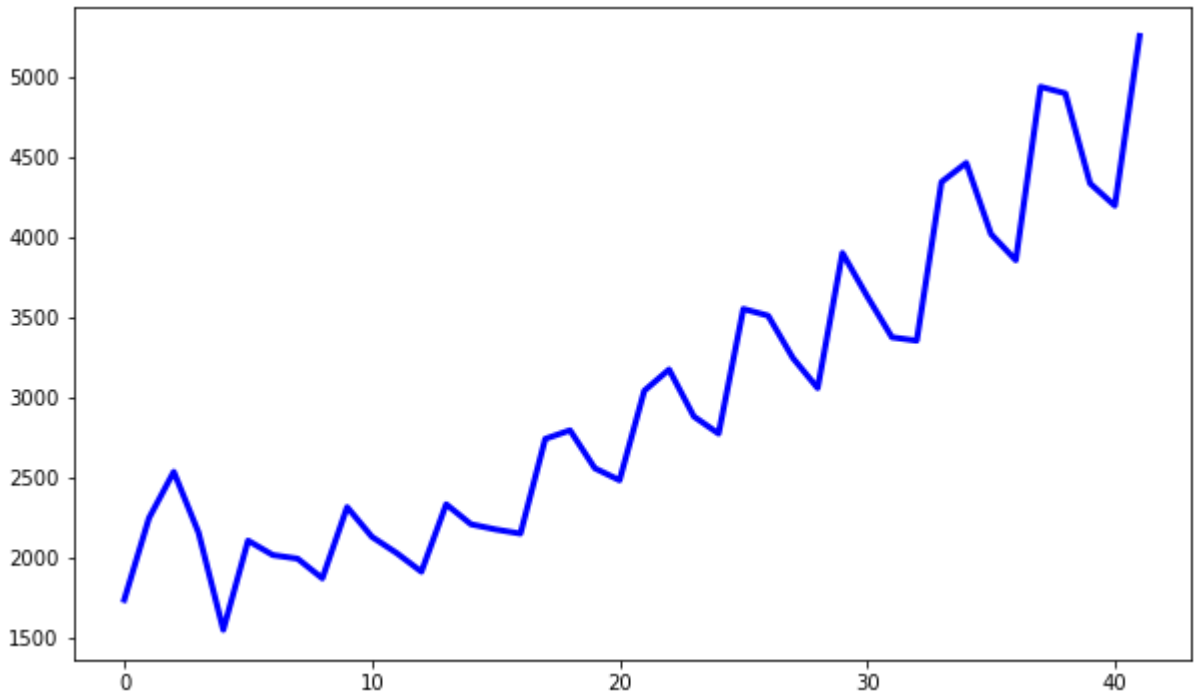
	Quarter	Sales	Quarters	Year
0	Q1_86	1734.827000	Q1	86
1	Q2_86	2244.960999	Q2	86
2	Q3_86	2533.804993	Q3	86
3	Q4_86	2154.962997	Q4	86
4	Q1_87	1547.818996	Q1	87

In [6]:
```
#getting dummy varibles for quarters Q1,Q2,Q3
Quarters_Dummies = pd.DataFrame(pd.get_dummies(data['Quarters']))
data =pd.concat([data,Quarters_Dummies],axis = 1)
data.head()
```

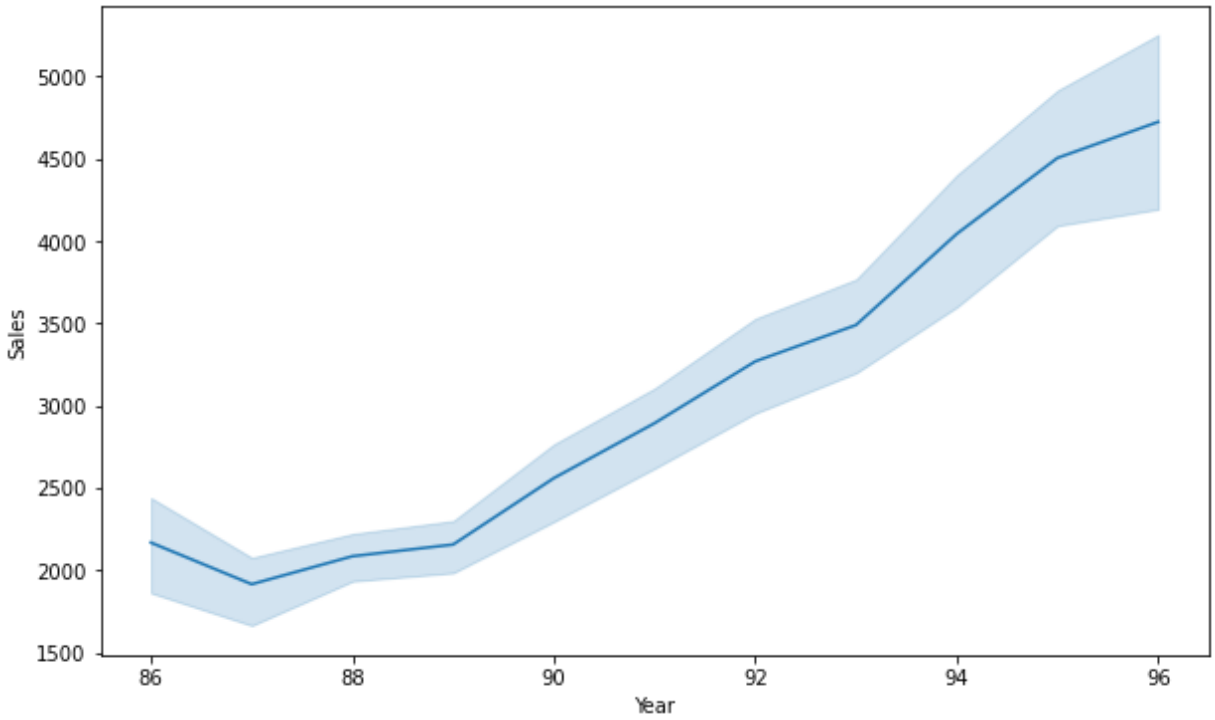
Out[6]:

	Quarter	Sales	Quarters	Year	Q1	Q2	Q3	Q4
0	Q1_86	1734.827000	Q1	86	1	0	0	0
1	Q2_86	2244.960999	Q2	86	0	1	0	0
2	Q3_86	2533.804993	Q3	86	0	0	1	0
3	Q4_86	2154.962997	Q4	86	0	0	0	1
4	Q1_87	1547.818996	Q1	87	1	0	0	0

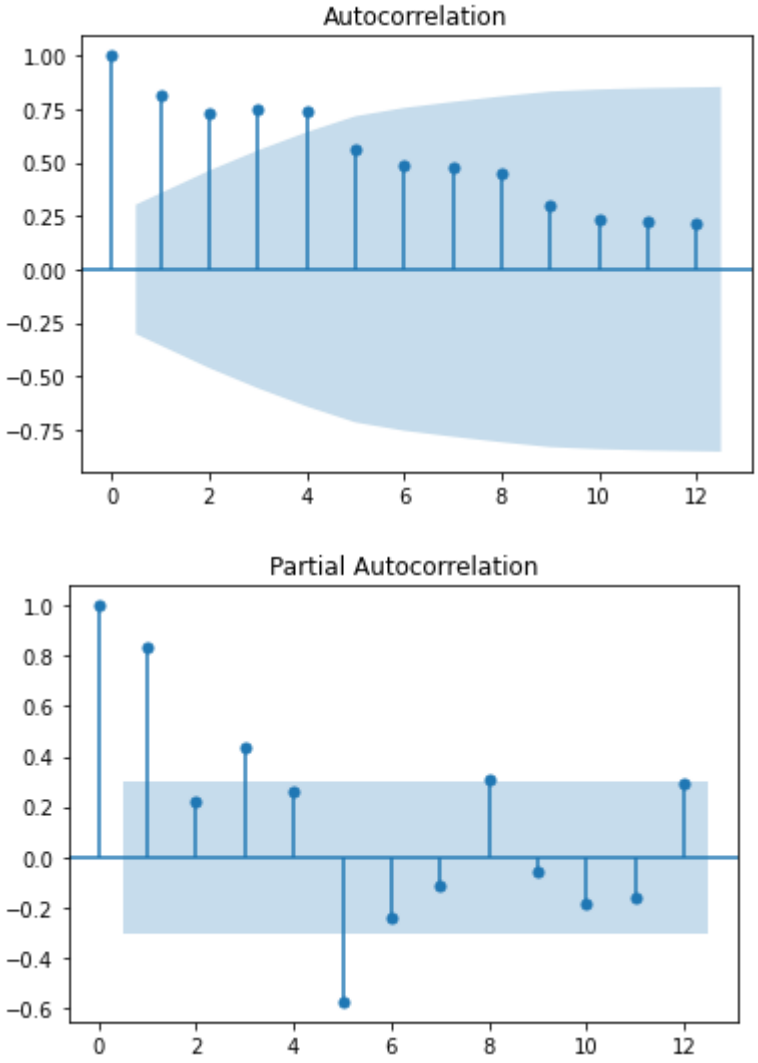
In [7]:
```
#Lineplotfor sales of CocaCla
plt.figure(figsize=(10,6))
plt.plot(data['Sales'],color = 'blue', linewidth = 3 )
plt.show()
```



In [8]:
```
plt.figure(figsize=(10,6))
sns.lineplot(x = "Year" , y="Sales",data=data)
None
```



In [9]:
```
import statsmodels.graphics.tsaplots as tsa_plots
tsa_plots.plot_acf(data.Sales,lags = 12)
tsa_plots.plot_pacf(data.Sales,lags = 12)
plt.show()
```



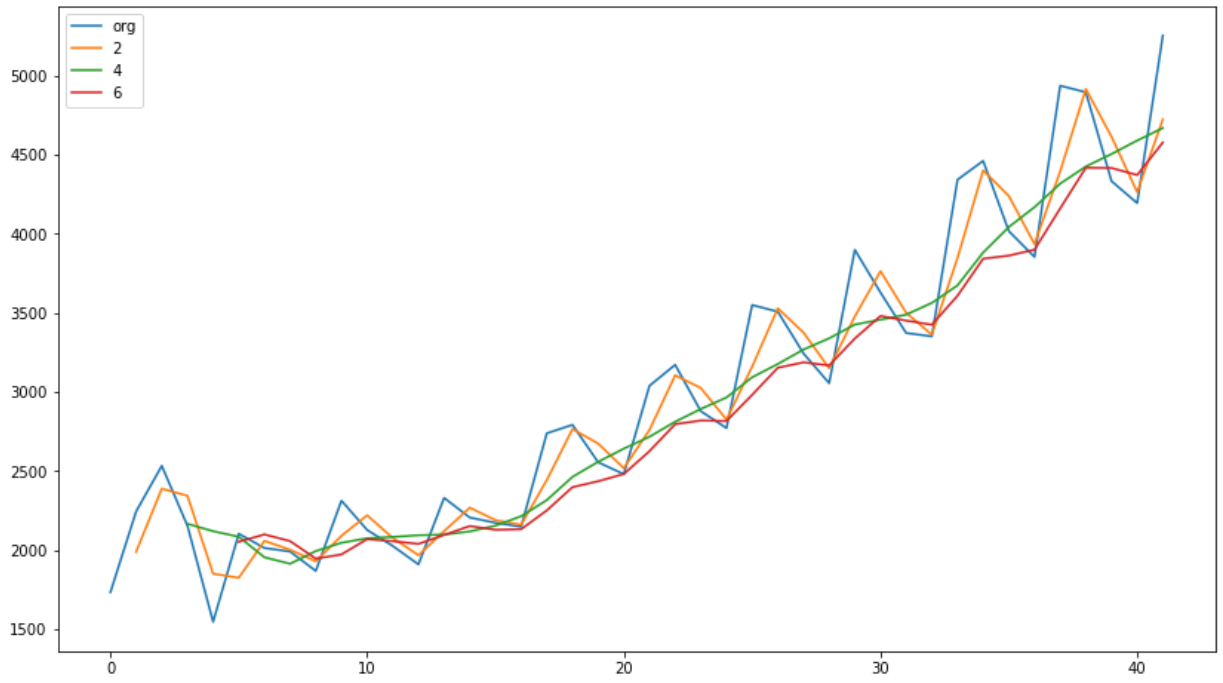
Data Driven Forecasting Methods

In [10]:
```
from statsmodels.tsa.holtwinters import SimpleExpSmoothing #SES
from statsmodels.tsa.holtwinters import Holt # holts Exponential Smoothing
from statsmodels.tsa import exponential_smoothing
```

In [11]:
```
#Splitting data into train and test
Train = data.head(32)
Test = data.tail(10)
```

Moving Average Method

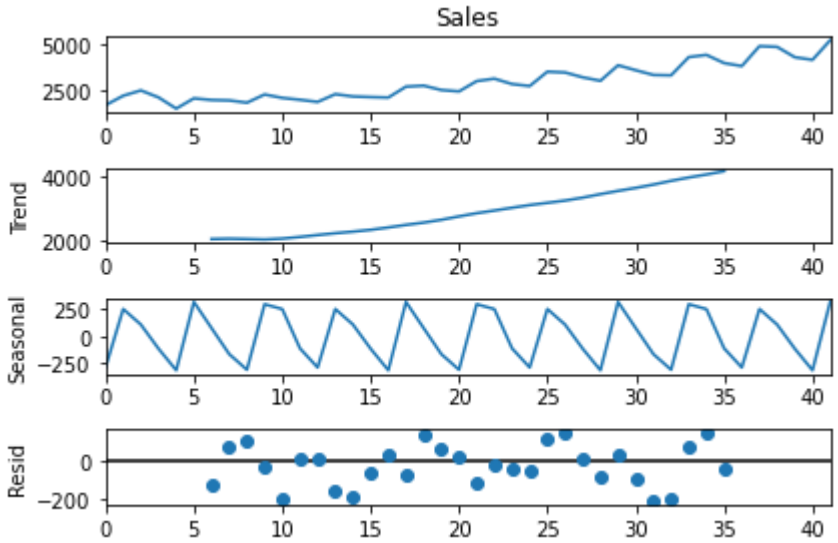
In [12]:
```
plt.figure(figsize=(14,8))
data.Sales.plot(label="org")
for i in range(2,8,2):
    data["Sales"].rolling(i).mean().plot(label=str(i))
plt.legend(loc='best')
plt.show()
```



Time series decomposition plot

In [13]:
```
from statsmodels.tsa.seasonal import seasonal_decompose


decompose_ts_add = seasonal_decompose(data.Sales,period=12)
decompose_ts_add.plot()
plt.show()
```



Evaluation metric RMSE

In [14]:
```
def RMSE(org, pred):
    rmse = np.sqrt(np.mean((np.array(org)-np.array(pred))**2))
    return rmse
```

Simple expontial method

In [15]:
```
ses_model = SimpleExpSmoothing(Train["Sales"]).fit()
pred_ses = ses_model.predict(start = Test.index[0],end = Test.index[-1])
rmse_ses = RMSE(Test.Sales, pred_ses)
rmse_ses
```

Out[15]: 1038.0146097085883

Holt method

In [16]:
```
hw_model = Holt(Train["Sales"]).fit()
pred_hw = hw_model.predict(start = Test.index[0],end = Test.index[-1])
rmse_hw = RMSE(Test.Sales,pred_hw)
rmse_hw
```

Out[16]: 924.9635257496326

Holts winter exponential smoothing with multiplicative seasonality and additive trend

In [17]:
```
hwe_model_mul_add = ExponentialSmoothing(Train["Sales"],seasonal="mul",trend="add
pred_hwe_mul_add = hwe_model_mul_add.predict(start = Test.index[0],end = Test.ind
rmse_hwe_mul_add = RMSE(Test.Sales, pred_hwe_mul_add)
rmse_hwe_mul_add
```

Out[17]: 587.207529421941

Model based Forecasting Methods

In [18]:
```
#Data preprocessing for models
data["t"] = np.arange(1,43)
data["t_squared"] = data["t"]*data["t"]

data["log_sales"] = np.log(data["Sales"])

data.head()
```

Out[18]:

	Quarter	Sales	Quarters	Year	Q1	Q2	Q3	Q4	t	t_squared	log_sales
0	Q1_86	1734.827000	Q1	86	1	0	0	0	1	1	7.458663
1	Q2_86	2244.960999	Q2	86	0	1	0	0	2	4	7.716443
2	Q3_86	2533.804993	Q3	86	0	0	1	0	3	9	7.837477
3	Q4_86	2154.962997	Q4	86	0	0	0	1	4	16	7.675529
4	Q1_87	1547.818996	Q1	87	1	0	0	0	5	25	7.344602

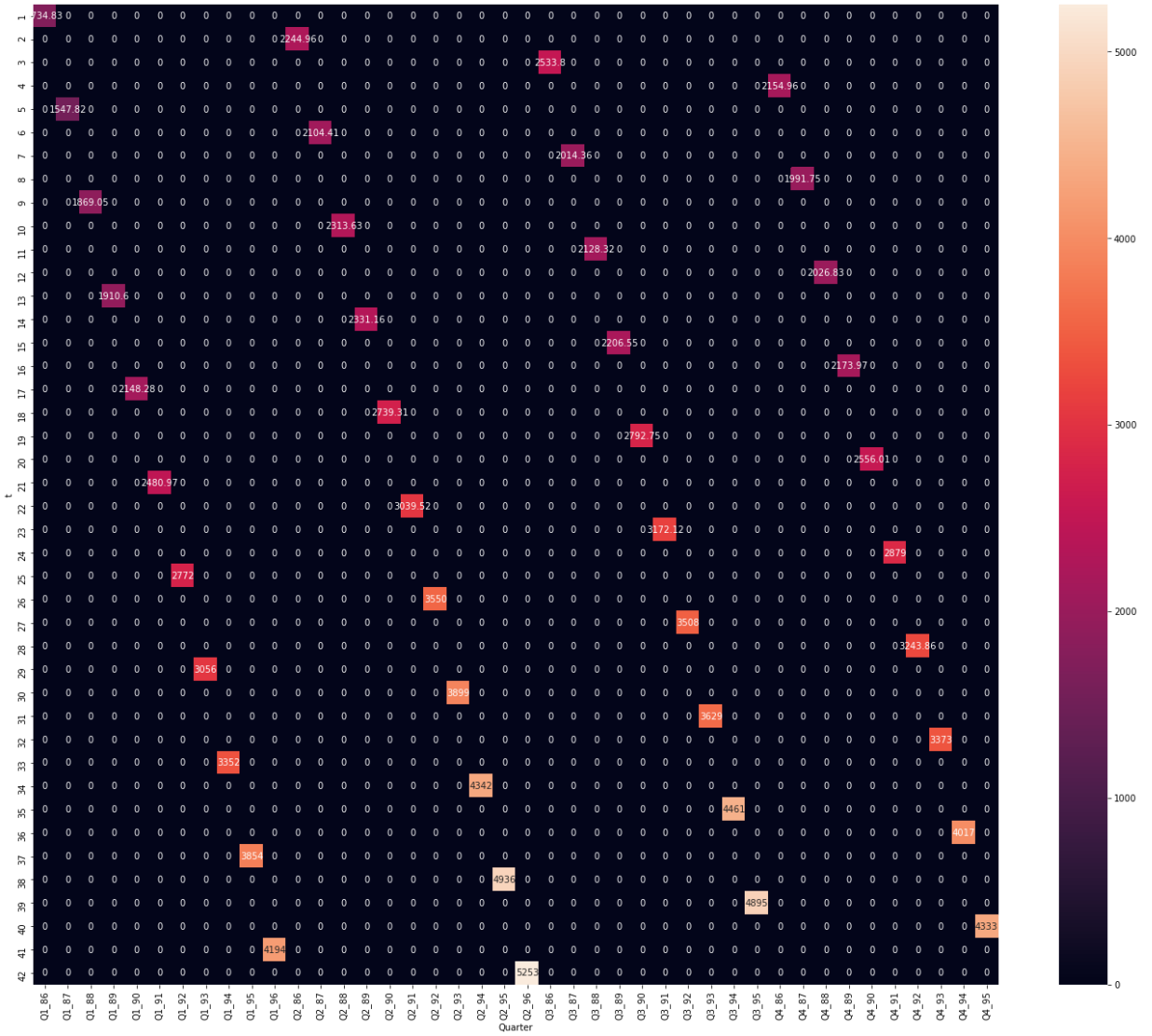
In [19]:
```
#splitting dat into train and test
Train = data.head(32)
Test = data.tail(10)
```

In [20]:
```
Train.head()
```

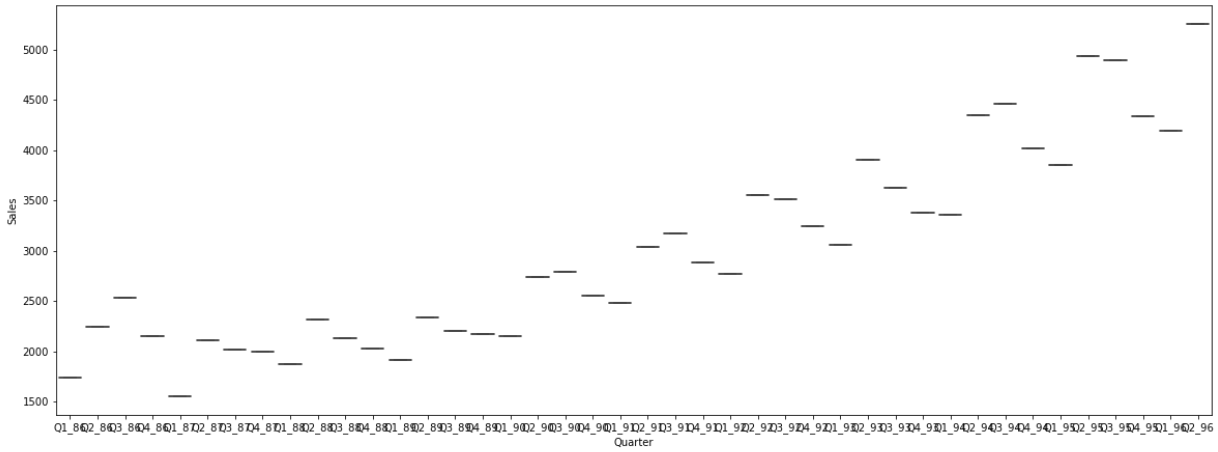
Out[20]:

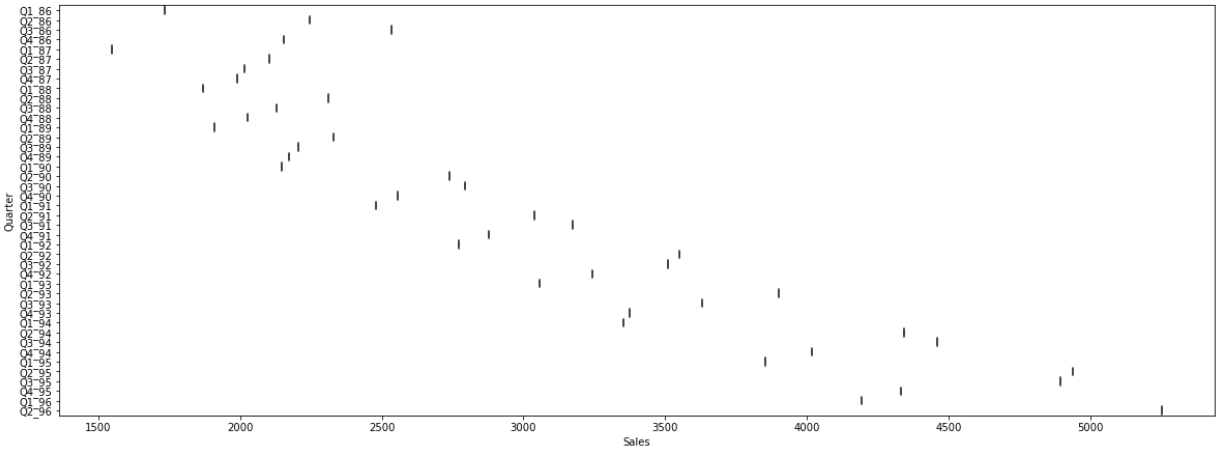
	Quarter	Sales	Quarters	Year	Q1	Q2	Q3	Q4	t	t_squared	log_sales
0	Q1_86	1734.827000	Q1	86	1	0	0	0	1	1	7.458663
1	Q2_86	2244.960999	Q2	86	0	1	0	0	2	4	7.716443
2	Q3_86	2533.804993	Q3	86	0	0	1	0	3	9	7.837477
3	Q4_86	2154.962997	Q4	86	0	0	0	1	4	16	7.675529
4	Q1_87	1547.818996	Q1	87	1	0	0	0	5	25	7.344602

In [21]:
```
plt.figure(figsize=(24,20))
heatmap_y_month = pd.pivot_table(data=data,values="Sales",index="t",columns="Quar
sns.heatmap(heatmap_y_month,annot=True,fmt="g")
None
```

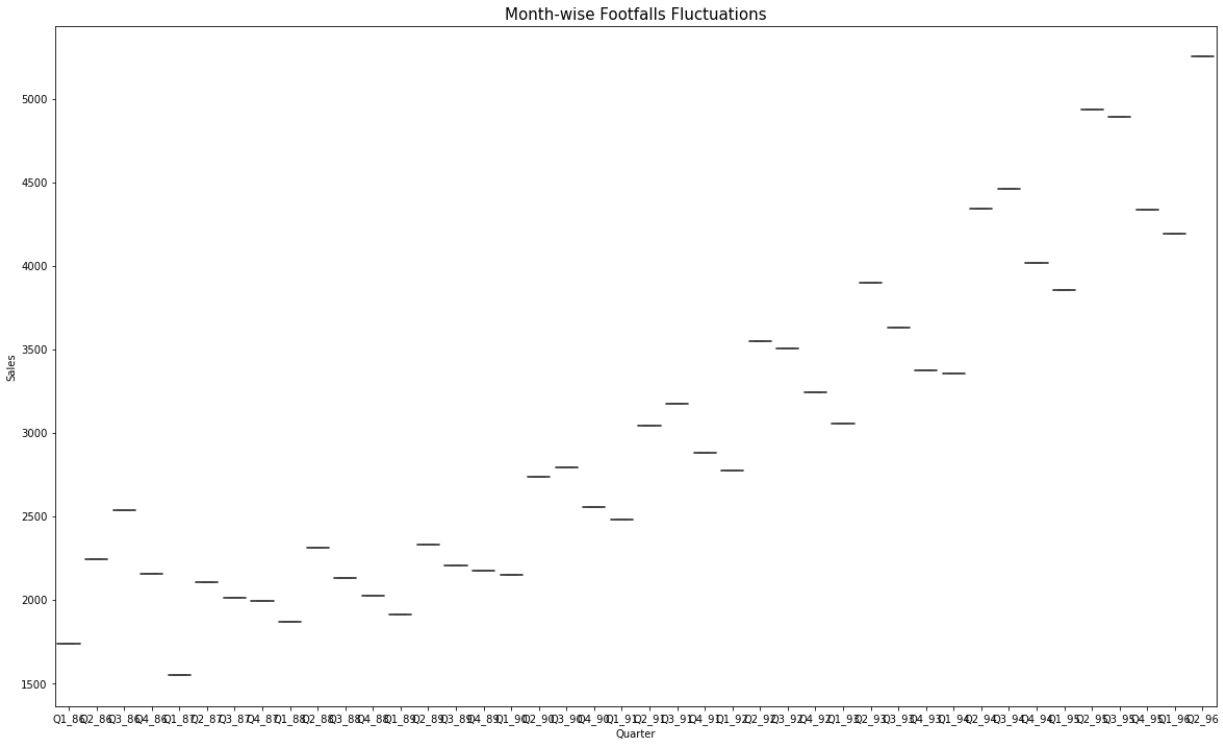


In [22]:
```
# Boxplot for ever
plt.figure(figsize=(20,16))
plt.subplot(2,1,1)
sns.boxplot(x="Quarter",y="Sales",data=data)
plt.subplot(2,1,2)
sns.boxplot(x="Sales",y="Quarter",data=data)
None
```





In [23]:
```
plt.figure(figsize=(20,12))
sns.boxplot(x="Quarter",y="Sales",data=data)
plt.title('Month-wise Footfalls Fluctuations',size = 15)
plt.show()
```



In [ ]:
```
```

In [24]:
```
Test.head()
```

Out[24]:

	Quarter	Sales	Quarters	Year	Q1	Q2	Q3	Q4	t	t_squared	log_sales
32	Q1_94	3352.0	Q1	94	1	0	0	0	33	1089	8.117312
33	Q2_94	4342.0	Q2	94	0	1	0	0	34	1156	8.376090
34	Q3_94	4461.0	Q3	94	0	0	1	0	35	1225	8.403128
35	Q4_94	4017.0	Q4	94	0	0	0	1	36	1296	8.298291
36	Q1_95	3854.0	Q1	95	1	0	0	0	37	1369	8.256867

Linear Model

In [25]:
```
import statsmodels.formula.api as smf

linear_model = smf.ols('Sales~t',data=Train).fit()
pred_linear =  pd.Series(linear_model.predict(pd.DataFrame(Test['t'])))
rmse_linear_model = RMSE(Test['Sales'], pred_linear)
rmse_linear_model
```

Out[25]: 752.9233932767113

Exponential Model

In [26]:
```
Exp = smf.ols('log_sales~t', data=Train).fit()
pred_Exp =pd.Series(Exp.predict(pd.DataFrame(Test['t'])))
rmse_exp = RMSE(Test['Sales'],np.exp(pred_Exp))
rmse_exp
```

Out[26]: 590.3316432076557

Quadratic Model

In [27]:
```
Quad = smf.ols('Sales~t+t_squared',data=Train).fit()
pred_Quad = pd.Series(Quad.predict(Test[["t","t_squared"]]))
rmse_quad_model = RMSE(Test['Sales'], pred_Quad)
rmse_quad_model
```

Out[27]: 457.73573554073965

Additive Seasonality model

In [28]:
```
add_sea = smf.ols('Sales~Q1+Q2+Q3',data=Train).fit()
pred_add_sea = pd.Series(add_sea.predict(Test[['Q1', 'Q2', 'Q3']]))
rmse_add_sea = RMSE(Test['Sales'], pred_add_sea)
rmse_add_sea
```

Out[28]: 1850.466546185835

Additive Seasonality Quadratic model

In [29]:
```
add_sea_Quad = smf.ols('Sales~t+t_squared+Q1+Q2+Q3',data=Train).fit()
pred_add_sea_quad = pd.Series(add_sea_Quad.predict(Test[['Q1','Q2','Q3','t','t_sq
rmse_add_sea_quad = RMSE(Test['Sales'], pred_add_sea_quad)
rmse_add_sea_quad
```

Out[29]: 277.35107711287395

Multiplicative Seasonality model

In [30]:
```
Mul_Add_sea = smf.ols('log_sales~t+Q1+Q2+Q3',data = Train).fit()
pred_Mul_add_sea = pd.Series(Mul_Add_sea.predict(Test))
rmse_Mul_add_sea = RMSE(Test['Sales'], np.exp(pred_Mul_add_sea))
rmse_Mul_add_sea
```

Out[30]: 448.86781826915234

In [31]:
```
list = [['Simple Exponential Method',rmse_ses], ['Holt method',rmse_hw],
          ['HW exp smoothing add',rmse_hwe_mul_add],['HW exp smoothing mult',rmse
          ['Linear Mode',rmse_hwe_mul_add],['Exp model',rmse_exp],['Quad model',r
          ['add seasonality',rmse_add_sea],['Quad add seasonality',rmse_add_sea_q
          ['Mult Seasonality',Mul_Add_sea],['Mult add seasonality',rmse_Mul_add_s
```

In [32]:
```
df = pd.DataFrame(list, columns =['Model', 'RMSE_Value'])
df
```

Out[32]:

	Model	RMSE_Value
0	Simple Exponential Method	1038.01461
1	Holt method	924.963526
2	HW exp smoothing add	587.207529
3	HW exp smoothing mult	587.207529
4	Linear Mode	587.207529
5	Exp model	590.331643
6	Quad model	457.735736
7	add seasonality	1850.466546
8	Quad add seasonality	277.351077
9	Mult Seasonality	<statsmodels.regression.linear_model.Regressio...
10	Mult add seasonality	448.867818

Building final model with least RMSE value

In [33]:
```
data.head()
```

Out[33]:

	Quarter	Sales	Quarters	Year	Q1	Q2	Q3	Q4	t	t_squared	log_sales
0	Q1_86	1734.827000	Q1	86	1	0	0	0	1	1	7.458663
1	Q2_86	2244.960999	Q2	86	0	1	0	0	2	4	7.716443
2	Q3_86	2533.804993	Q3	86	0	0	1	0	3	9	7.837477
3	Q4_86	2154.962997	Q4	86	0	0	0	1	4	16	7.675529
4	Q1_87	1547.818996	Q1	87	1	0	0	0	5	25	7.344602

In [34]:
```
final_model = smf.ols('Sales~t+t_squared+Q1+Q2+Q3',data=data).fit()
pred_final = pd.Series(final_model.predict(data[['Q1','Q2','Q3','t','t_squared']]
rmse_final_model = RMSE(data['Sales'], pred_final)
rmse_final_model
```

Out[34]: 159.05522576522475

In [35]:
```
pred_df = pd.DataFrame({'Actual' : data.Sales, 'Predicted' : pred_final})
pred_df
```

Out[35]:

	Actual	Predicted
0	1734.827000	1626.592727
1	2244.960999	2268.402339
2	2533.804993	2189.685821
3	2154.962997	1867.242917
4	1547.818996	1658.311143
5	2104.411995	2314.185178
6	2014.362999	2249.533083
7	1991.746998	1941.154603
8	1869.049999	1746.287252
9	2313.631996	2416.225709
10	2128.320000	2365.638037
11	2026.828999	2071.323979
12	1910.603996	1890.521051
13	2331.164993	2574.523931
14	2206.549995	2538.000682
15	2173.967995	2257.751047
16	2148.278000	2091.012542
17	2739.307999	2789.079845
18	2792.753998	2766.621019
19	2556.009995	2500.435807
20	2480.973999	2347.761724
21	3039.522995	3059.893450
22	3172.115997	3051.499047
23	2879.000999	2799.378257
24	2772.000000	2660.768598
25	3550.000000	3386.964746
26	3508.000000	3392.634766
27	3243.859993	3154.578400
28	3056.000000	3030.033163
29	3899.000000	3770.293734
30	3629.000000	3790.028177
31	3373.000000	3566.036233
32	3352.000000	3455.555419

	Actual	Predicted
33	4342.000000	4209.880414
34	4461.000000	4243.679279
35	4017.000000	4033.751758
36	3854.000000	3937.335367
37	4936.000000	4705.724784
38	4895.000000	4753.588072
39	4333.000000	4557.724974
40	4194.000000	4475.373006
41	5253.000000	5257.826846