In [5]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import math
from scipy.stats import variation
%matplotlib inline
warnings.filterwarnings('ignore')
sns.set_style("darkgrid")
import sklearn
from sklearn.metrics import mean_squared_error
```

In [6]:

```
data=pd.read_excel("Global Superstore.xls")
```

In [7]:

```
data.head()
```

Out[7]:

	Order Date	Segment	Market	Sales	Profit
0	2012-07-31	Consumer	US	2309.650	762.1845
1	2013-02-05	Corporate	APAC	3709.395	-288.7650
2	2013-10-17	Consumer	APAC	5175.171	919.9710
3	2013-01-28	Home Office	EU	2892.510	-96.5400
4	2013-11-05	Consumer	Africa	2832.960	311.5200

In [8]:

```
data.shape
```

Out[8]:

```
(51290, 5)
```

In [9]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   Order Date  51290 non-null  datetime64[ns]
 1   Segment     51290 non-null  object        
 2   Market      51290 non-null  object        
 3   Sales       51290 non-null  float64       
 4   Profit      51290 non-null  float64       
dtypes: datetime64[ns](1), float64(2), object(2)
memory usage: 2.0+ MB
```

In [10]:

```
data.describe() # statistics
```

Out[10]:

	Sales	Profit
count	51290.000000	51290.000000
mean	246.490581	28.610982
std	487.565361	174.340972
min	0.444000	-6599.978000
25%	30.758625	0.000000
50%	85.053000	9.240000
75%	251.053200	36.810000
max	22638.480000	8399.976000

In [11]:

```
data.isnull().sum() # missingvalues
```

Out[11]:

```
Order Date    0
Segment       0
Market        0
Sales         0
Profit        0
dtype: int64
```

In [12]:

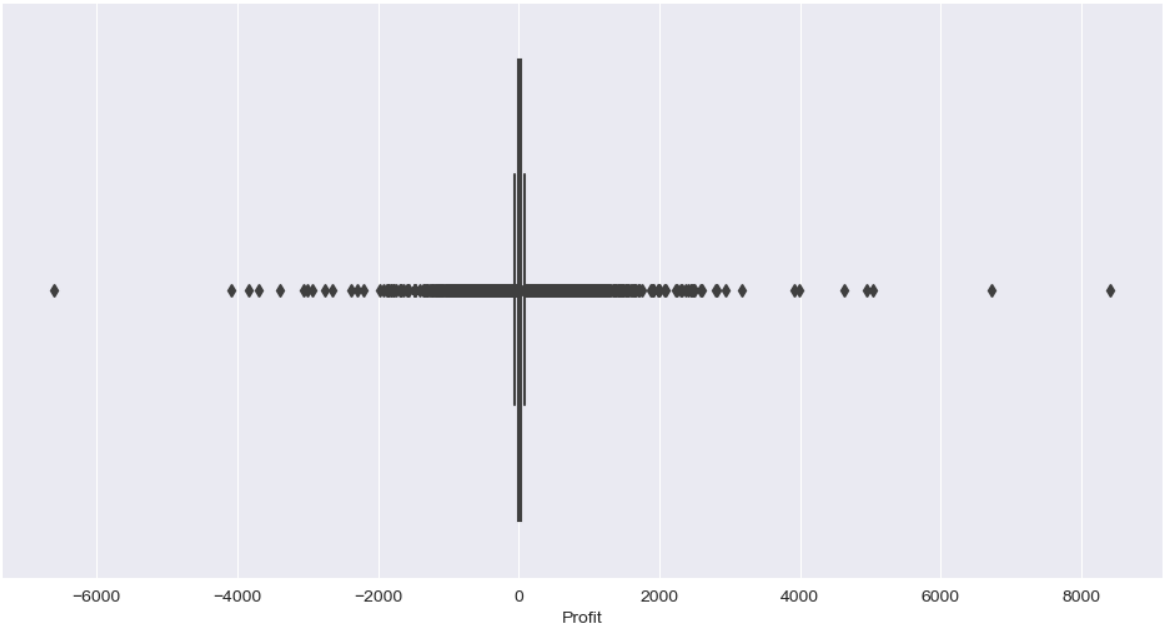
```
data.describe(percentiles=[.25,.5,.75,.90,.95,.99]) # outliers
```

Out[12]:

	Sales	Profit
count	51290.000000	51290.000000
mean	246.490581	28.610982
std	487.565361	174.340972
min	0.444000	-6599.978000
25%	30.758625	0.000000
50%	85.053000	9.240000
75%	251.053200	36.810000
90%	632.225736	112.680000
95%	1015.955640	211.500000
99%	2301.000000	587.359950
max	22638.480000	8399.976000

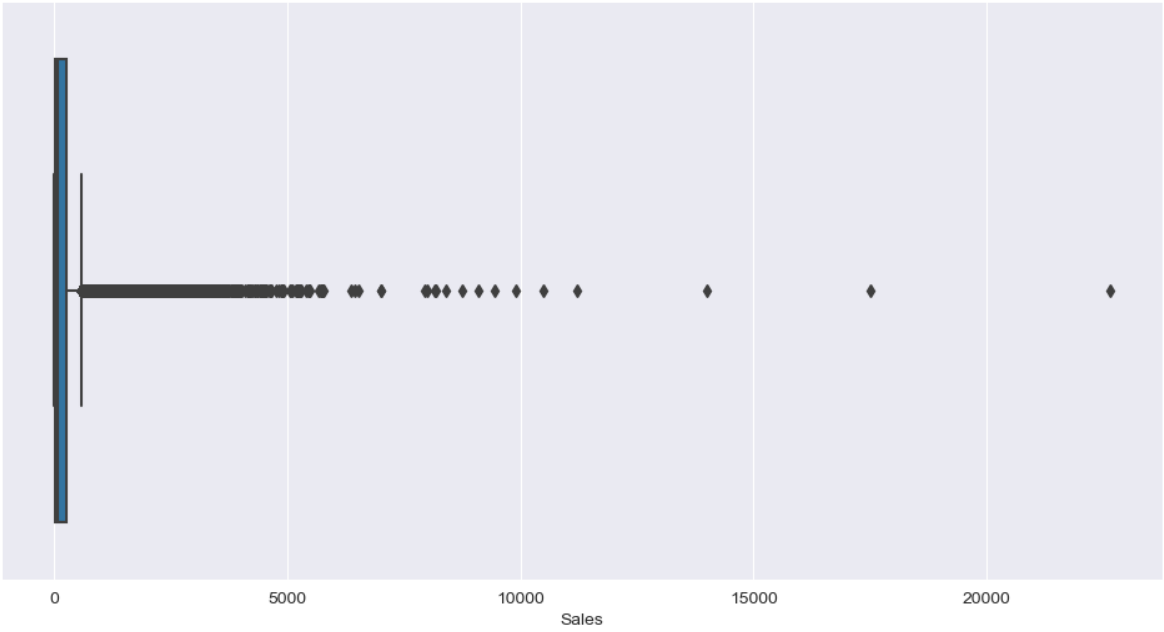
In [13]:

```
plt.figure(figsize=(12,6))
sns.boxplot(data['Profit'])
plt.show()
```



In [14]:

```
plt.figure(figsize=(12,6))
sns.boxplot(data['Sales'])
plt.show()
```



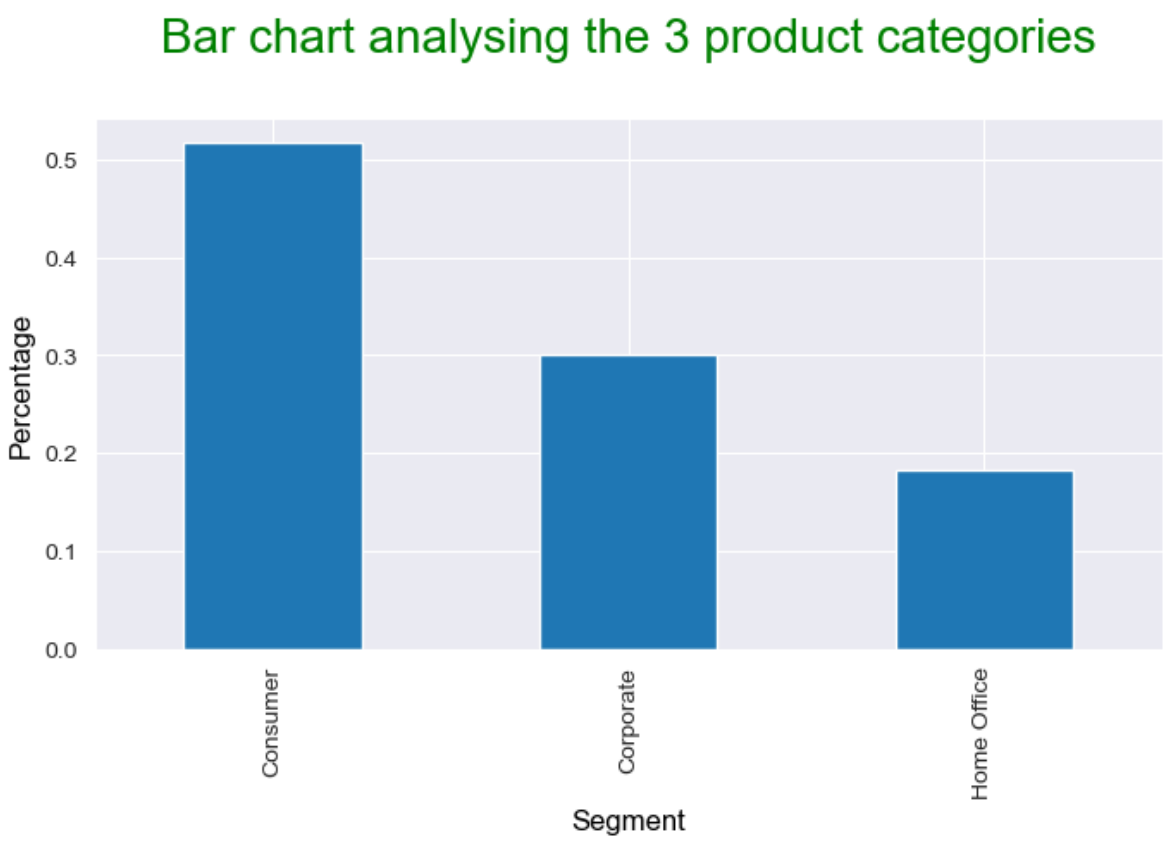
In [15]:

```
data.info() # Univariate Analysis
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Order Date  51290 non-null  datetime64[ns]
 1   Segment     51290 non-null  object
 2   Market      51290 non-null  object
 3   Sales       51290 non-null  float64
 4   Profit      51290 non-null  float64
dtypes: datetime64[ns](1), float64(2), object(2)
memory usage: 2.0+ MB
```

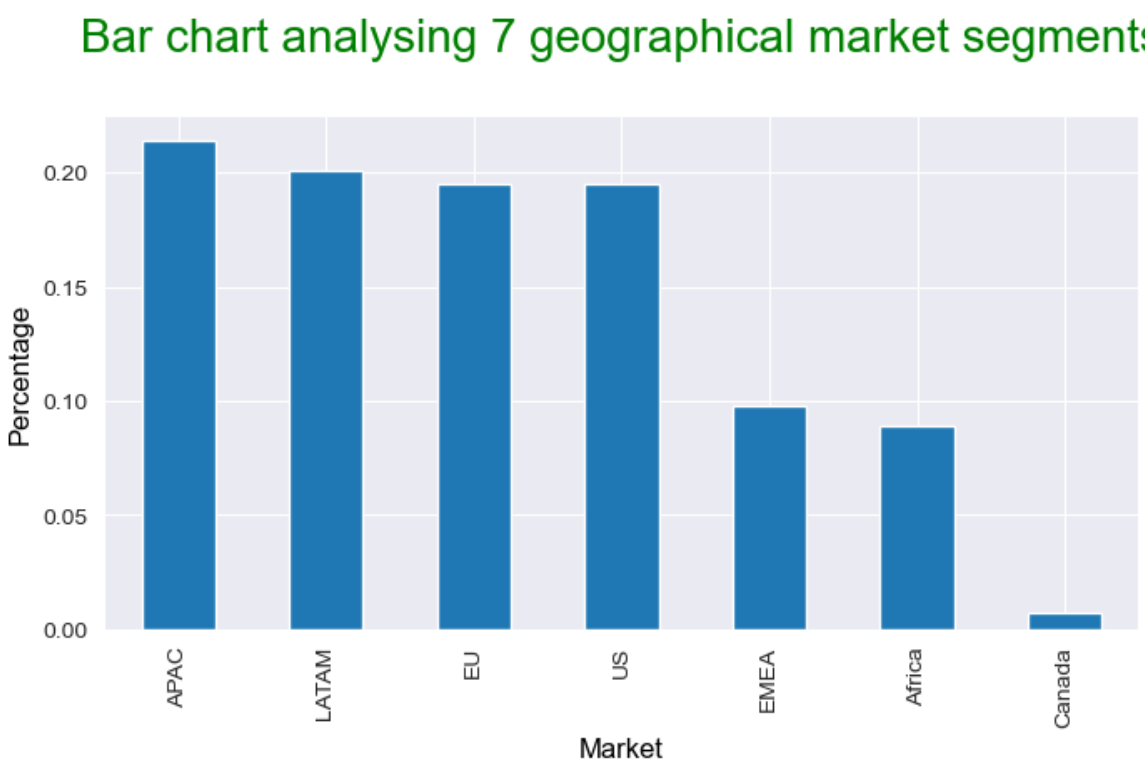
In [16]:

```
plt.figure(figsize= (8,4))
data["Segment"].value_counts(normalize=True).plot.bar()
plt.title("Bar chart analysing the 3 product categories\n", fontdict={'fontsize': 20, 'font
plt.xlabel("Segment", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black'})
plt.ylabel("Percentage", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black'} )
plt.show()
```



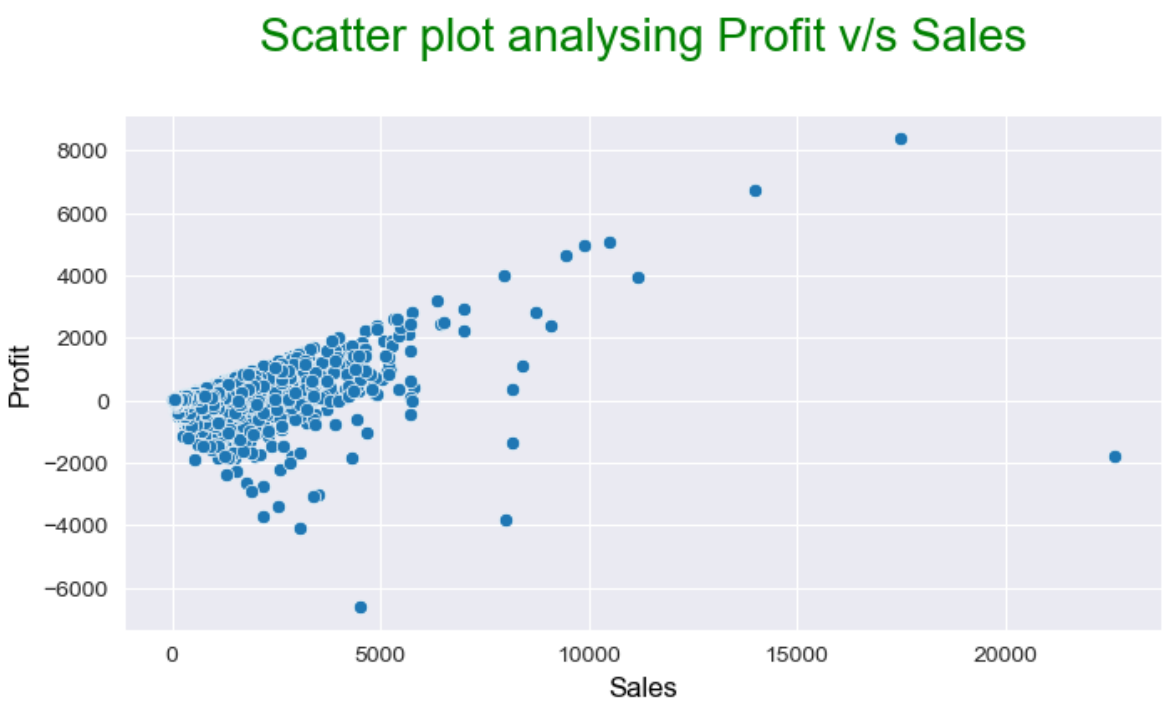
In [17]:

```
plt.figure(figsize= (8,4))
data["Market"].value_counts(normalize=True).plot.bar()
plt.title("Bar chart analysing 7 geographical market segments\n", fontdict={'fontsize': 20,
plt.xlabel("Market", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black'})
plt.ylabel("Percentage", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black'} )
plt.show()
```



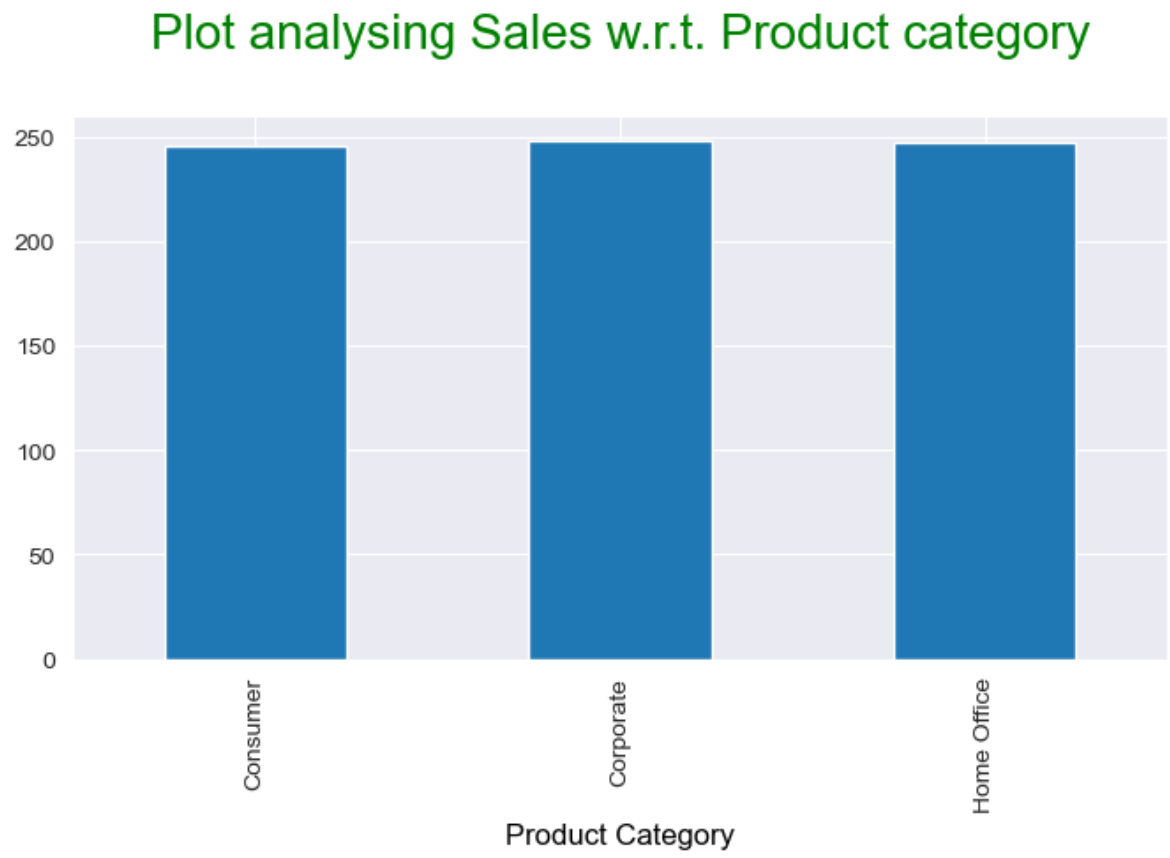
In [18]:

```
plt.figure(figsize= [8,4])
sns.scatterplot(data['Sales'], data['Profit'])
plt.title("Scatter plot analysing Profit v/s Sales\n", fontdict={'fontsize': 20, 'fontweigh
plt.xlabel("Sales", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black'})
plt.ylabel("Profit", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black'} )
plt.show()
```



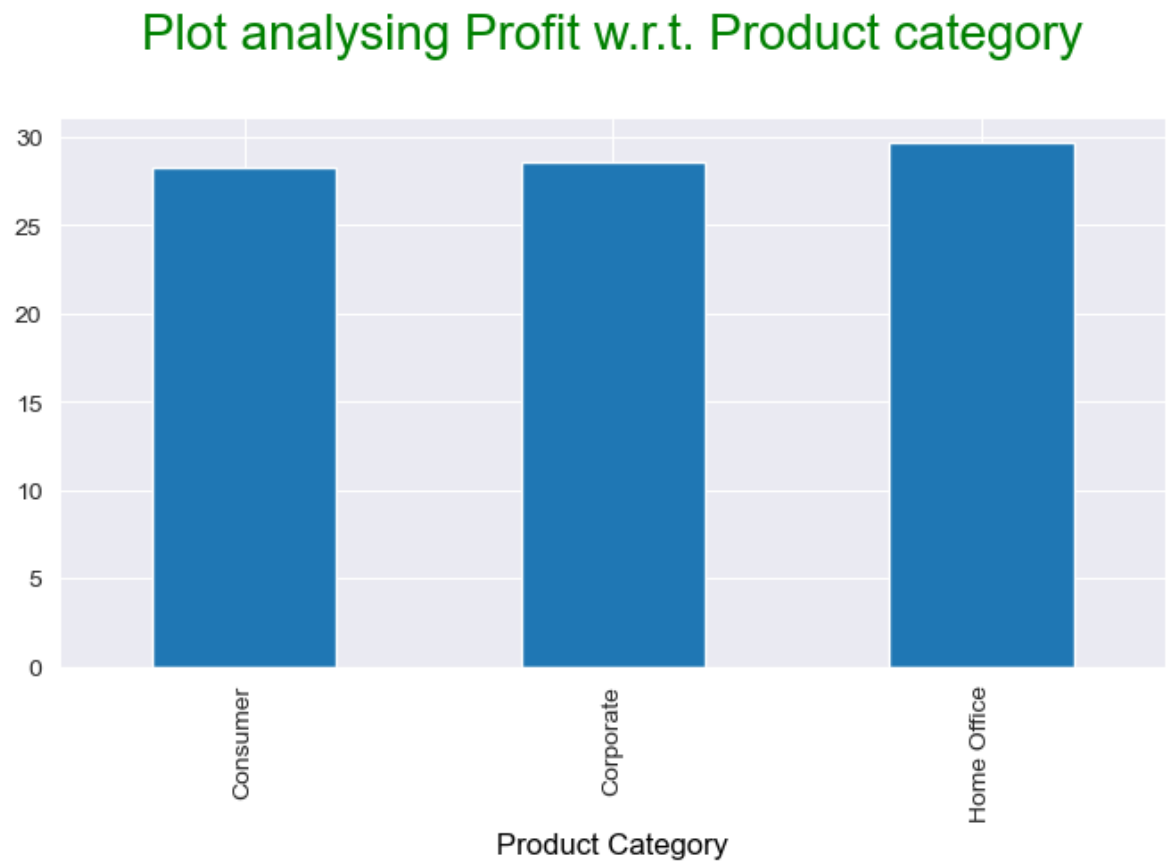
In [19]:

```
plt.figure(figsize= (8,4))
data.groupby("Segment")["Sales"].mean().plot.bar()
plt.title("Plot analysing Sales w.r.t. Product category\n", fontdict={'fontsize': 20, 'font
plt.xlabel("Product Category", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black
plt.show()
```



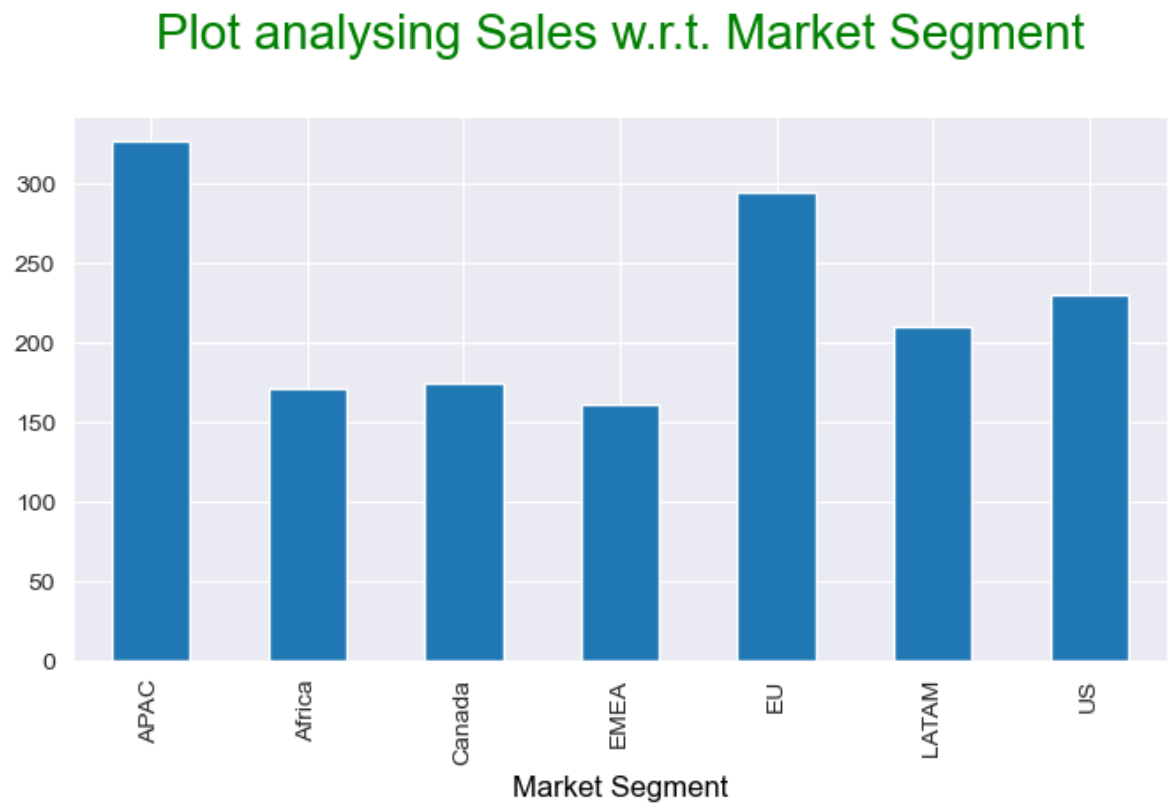
In [20]:

```
plt.figure(figsize= (8,4))
data.groupby("Segment")["Profit"].mean().plot.bar()
plt.title("Plot analysing Profit w.r.t. Product category\n", fontdict={'fontsize': 20, 'fon
plt.xlabel("Product Category", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black
plt.show()
```



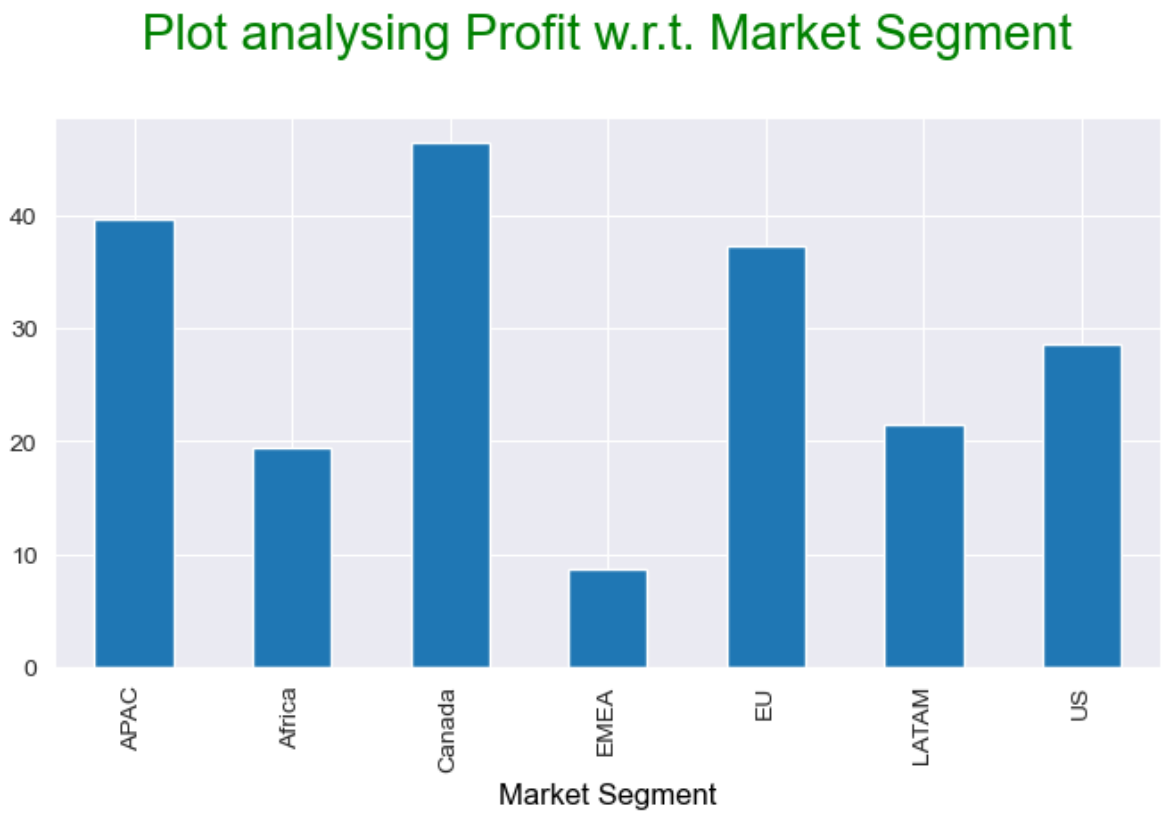
In [21]:

```
plt.figure(figsize= (8,4))
data.groupby("Market")["Sales"].mean().plot.bar()
plt.title("Plot analysing Sales w.r.t. Market Segment\n", fontdict={'fontsize': 20, 'fontwe
plt.xlabel("Market Segment", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black'}
plt.show()
```



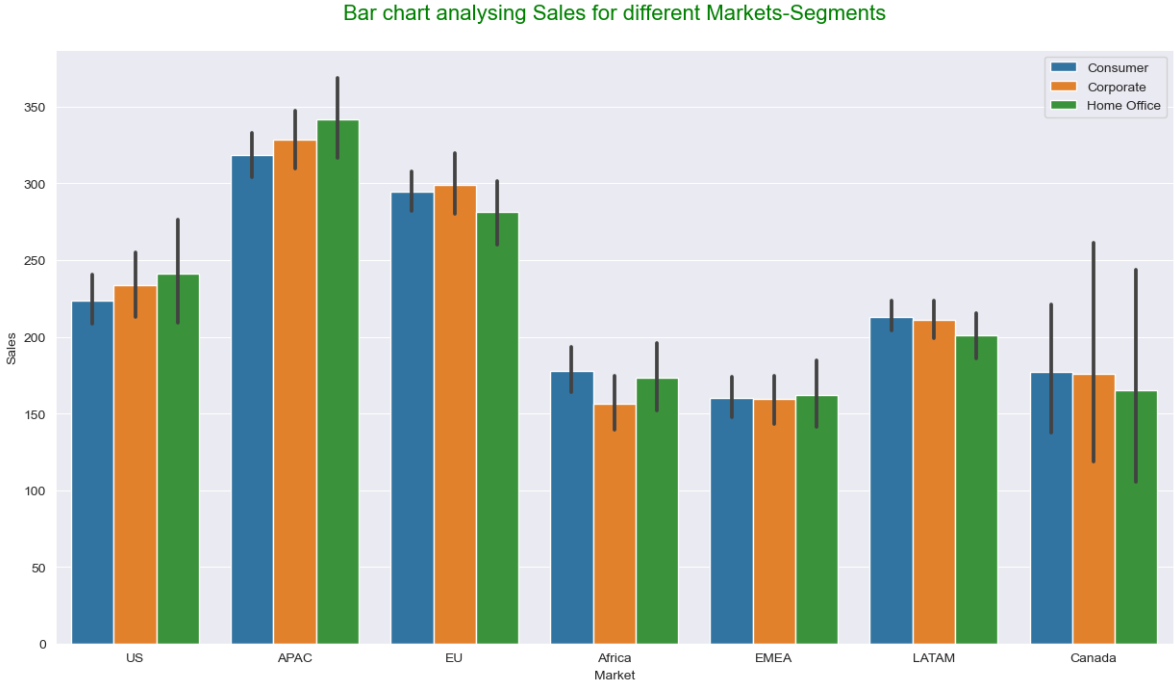
In [22]:

```
plt.figure(figsize= (8,4))
data.groupby("Market")["Profit"].mean().plot.bar()
plt.title("Plot analysing Profit w.r.t. Market Segment\n", fontdict={'fontsize': 20, 'fontw
plt.xlabel("Market Segment", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black'}
plt.show()
```



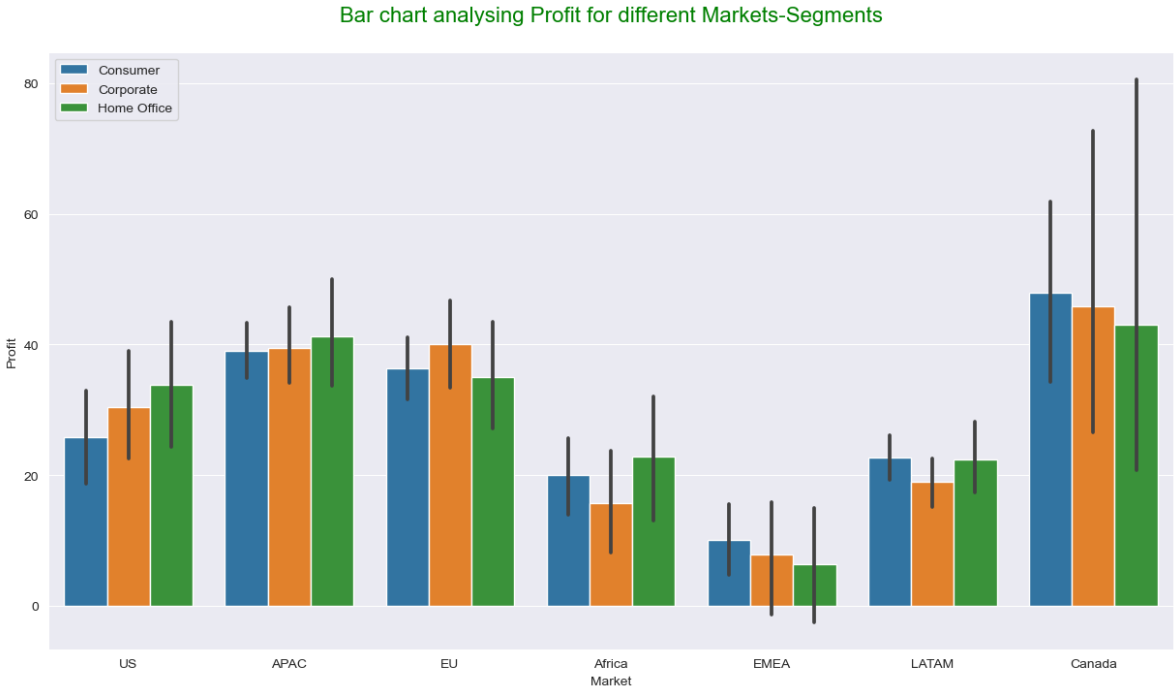
In [23]:

```
plt.figure(figsize=(15,8))
sns.barplot(data=data, x='Market', y= 'Sales', hue='Segment')
plt.title('Bar chart analysing Sales for different Markets-Segments\n', fontdict={'fontsize
plt.legend(loc = 'best')
plt.show()
```



In [24]:

```
plt.figure(figsize=(15,8))
sns.barplot(data=data, x='Market', y= 'Profit', hue='Segment')
plt.title('Bar chart analysing Profit for different Markets-Segments\n', fontdict={'fontsiz
plt.legend(loc = 'best')
plt.show()
```



In [25]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   Order Date  51290 non-null  datetime64[ns]
 1   Segment     51290 non-null  object        
 2   Market      51290 non-null  object        
 3   Sales       51290 non-null  float64       
 4   Profit      51290 non-null  float64       
dtypes: datetime64[ns](1), float64(2), object(2)
memory usage: 2.0+ MB
```

In [26]:

```
# Concatenate the Market and Segment column to get the Market Segment data
data['Market_Segment'] = data['Market']+'-'+data['Segment']
```

In [27]:

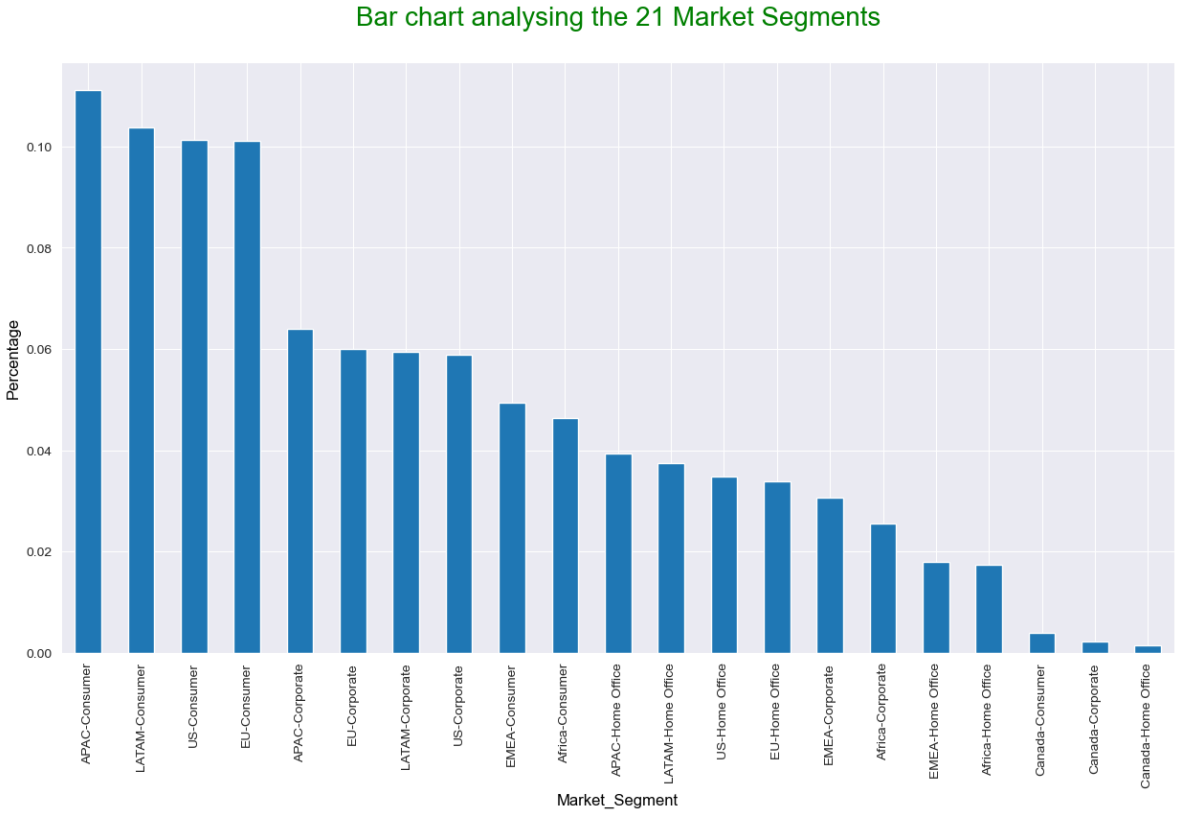
```
data.head()
```

Out[27]:

	Order Date	Segment	Market	Sales	Profit	Market_Segment
0	2012-07-31	Consumer	US	2309.650	762.1845	US-Consumer
1	2013-02-05	Corporate	APAC	3709.395	-288.7650	APAC-Corporate
2	2013-10-17	Consumer	APAC	5175.171	919.9710	APAC-Consumer
3	2013-01-28	Home Office	EU	2892.510	-96.5400	EU-Home Office
4	2013-11-05	Consumer	Africa	2832.960	311.5200	Africa-Consumer

In [28]:

```
plt.figure(figsize= (15,8))
data["Market_Segment"].value_counts(normalize=True).plot.bar()
plt.title("Bar chart analysing the 21 Market Segments\n", fontdict={'fontsize': 20, 'fontwe
plt.xlabel("Market_Segment", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black'}
plt.ylabel("Percentage", fontdict={'fontsize': 12, 'fontweight' : 5, 'color' : 'Black'} )
plt.show()
```



In [29]:

```
data.drop(['Segment', 'Market'], axis = 1, inplace = True)
```

In [30]:

```
data.head()
```

Out[30]:

	Order Date	Sales	Profit	Market_Segment
0	2012-07-31	2309.650	762.1845	US-Consumer
1	2013-02-05	3709.395	-288.7650	APAC-Corporate
2	2013-10-17	5175.171	919.9710	APAC-Consumer
3	2013-01-28	2892.510	-96.5400	EU-Home Office
4	2013-11-05	2832.960	311.5200	Africa-Consumer

In [31]:

```
data['Order Date'].head()
```

Out[31]:

```
0   2012-07-31
1   2013-02-05
2   2013-10-17
3   2013-01-28
4   2013-11-05
Name: Order Date, dtype: datetime64[ns]
```

In [32]:

```
data['Order Date'] = pd.to_datetime(data['Order Date']).dt.to_period('m')
```

In [33]:

```
data.head()
```

Out[33]:

	Order Date	Sales	Profit	Market_Segment
0	2012-07	2309.650	762.1845	US-Consumer
1	2013-02	3709.395	-288.7650	APAC-Corporate
2	2013-10	5175.171	919.9710	APAC-Consumer
3	2013-01	2892.510	-96.5400	EU-Home Office
4	2013-11	2832.960	311.5200	Africa-Consumer

In [34]:

```
data.groupby(['Market_Segment', 'Order Date']).sum()
```

Out[34]:

		Sales	Profit
Market_Segment	**Order Date**		
APAC-Consumer	**2011-01**	15711.7125	991.2825
	2011-02	12910.8588	1338.8688
	2011-03	19472.5632	3747.1632
	2011-04	15440.3046	3846.4746
	2011-05	24348.9723	3639.9423
...	**...**	...	...
US-Home Office	**2014-08**	6024.9360	1552.4576
	2014-09	20435.0080	2249.1033
	2014-10	29871.8930	4031.1877
	2014-11	21046.7080	2303.8764
	2014-12	16044.6550	3965.6265

965 rows × 2 columns

In [35]:

```
data_cov = pd.pivot_table(data = data, index = "Order Date", columns = "Market_Segment", va
data_cov # pivoting
```

Out[35]:

Market_Segment	APAC-Consumer	APAC-Corporate	APAC-Home Office	Africa-Consumer	Africa-Corporate	Africa-Home Office	Canada-Consumer
Order Date							
2011-01	991.2825	11.5998	86.4423	475.683	219.096	856.710	3.12
2011-02	1338.8688	4358.8254	-417.4128	1441.926	-490.551	820.302	23.31
2011-03	3747.1632	1213.3386	923.7492	322.140	-586.716	67.320	335.55
2011-04	3846.4746	71.0265	657.1080	292.122	776.691	500.136	55.08
2011-05	3639.9423	2534.1672	-272.1717	110.004	241.338	34.926	77.97
2011-06	4328.2596	1435.8294	3452.1018	-1290.639	-259.218	-774.801	7.50
2011-07	1258.9617	525.4647	-42.0498	621.168	134.847	-506.562	47.28
2011-08	775.8066	4070.5581	775.7616	232.917	915.885	1090.200	43.68
2011-09	5181.4449	1952.4675	623.3595	-86.163	950.766	1345.740	134.01
2011-10	6911.9970	5452.2429	1131.9597	612.942	-97.821	387.312	35.43
2011-11	221.5539	2154.5313	6574.6911	-221.124	608.700	639.633	135.66
2011-12	4004.3211	8126.9478	1384.9611	642.921	875.220	41.064	NaN
2012-01	4810.1535	2224.6740	709.7085	762.894	-192.171	35.760	NaN
2012-02	2967.1281	659.0961	335.0127	-352.278	131.235	44.823	NaN
2012-03	146.4261	1108.3668	1381.6839	774.156	498.090	-99.330	6.03
2012-04	2368.1721	1873.7997	-97.5930	347.982	306.720	467.070	27.00
2012-05	6114.3312	4751.5857	3500.2146	-342.684	-57.648	224.304	52.65
2012-06	4790.0052	4853.2323	124.5438	2357.850	-421.314	-144.864	1785.84
2012-07	606.0216	-131.9754	317.1483	971.247	42.210	4.749	601.44
2012-08	7600.9977	4136.7153	2078.2506	859.734	788.667	-107.661	338.79
2012-09	2401.1736	1440.7560	-181.9125	1617.435	-74.097	16.056	62.76
2012-10	7759.0233	1913.3238	3178.9056	2212.704	411.564	-539.322	31.68
2012-11	5676.1440	1308.7617	1234.4160	1154.934	742.626	-24.429	595.83
2012-12	3835.9467	3188.5167	428.0961	-427.119	-362.340	281.382	45.90
2013-01	3555.8820	2789.8638	293.7639	459.837	-571.128	1058.550	281.19
2013-02	3225.6978	203.9898	884.3511	2101.842	1127.961	2953.851	151.17
2013-03	5333.4909	2405.1768	1853.6781	397.629	-53.928	155.280	82.08
2013-04	2304.9297	576.3489	657.9576	360.393	590.757	149.253	NaN
2013-05	5004.9900	1080.3939	2248.8078	384.963	957.030	468.522	NaN
2013-06	12689.2389	5407.8054	2203.7766	669.594	458.568	264.882	193.11
2013-07	3417.3747	2706.1569	1828.2861	3180.492	89.532	-231.180	122.43

Market_Segment	APAC-Consumer	APAC-Corporate	APAC-Home Office	Africa-Consumer	Africa-Corporate	Africa-Home Office	Canada-Consumer
Order Date							
2013-08	3856.6239	5242.6287	2526.1674	938.934	1448.235	-855.510	423.00
2013-09	8307.3735	5859.4245	-72.5928	2074.941	1956.138	183.732	183.93
2013-10	6231.6432	1407.9099	2974.9104	128.148	1600.533	1081.650	468.30
2013-11	6932.1384	2917.8372	3621.8820	1874.955	900.804	920.370	338.37
2013-12	4783.6104	4462.6371	3378.9714	-792.147	-359.031	612.660	319.23
2014-01	4893.3396	1287.2886	-772.0530	2684.742	-129.156	308.880	NaN
2014-02	1151.9886	1895.7504	918.6291	-562.335	217.485	326.019	1014.06
2014-03	6598.9896	623.5428	2962.1511	-136.956	-120.711	1610.370	133.47
2014-04	2275.9362	2474.7813	1806.8562	-370.251	2858.532	-165.882	19.26
2014-05	4780.1628	2469.6885	2018.5770	4109.970	972.849	226.140	39.21
2014-06	6724.2411	4348.8798	701.9766	2958.252	846.897	257.985	67.11
2014-07	2732.4828	2874.3051	2064.9264	737.898	485.172	1249.146	14.76
2014-08	5050.6929	1722.2874	2211.4455	3934.101	-66.864	1067.970	321.18
2014-09	5758.2162	5392.7694	7290.0429	3345.822	-387.165	765.531	194.25
2014-10	11824.2486	4155.1648	5850.4515	2278.191	2678.727	-103.149	299.13
2014-11	12869.9883	5693.9367	5344.7532	3824.202	-217.344	3060.981	183.06
2014-12	7192.6810	2504.8152	2760.5631	98.130	301.293	385.998	382.89

48 rows × 21 columns

In [36]:

```
train_len = 42
train = data_cov[0 : train_len]
test = data_cov[train_len : ]
```

In [37]:

```
train.shape
```

Out[37]:

```
(42, 21)
```

In [38]:

```
test.shape
```

Out[38]:

```
(6, 21)
```

In [39]:

```
train_mean = np.mean(train)
train_mean # mean
```

Out[39]:

```
Market_Segment
APAC-Consumer         4223.553586
APAC-Corporate        2556.998957
APAC-Home Office      1379.120743
Africa-Consumer        798.898929
Africa-Corporate       426.027286
Africa-Home Office     333.002143
Canada-Consumer        230.067500
Canada-Corporate       110.377500
Canada-Home Office     138.247500
EMEA-Consumer          415.354786
EMEA-Corporate         172.274500
EMEA-Home Office       123.249214
EU-Consumer           3627.517036
EU-Corporate          2251.993036
EU-Home Office        1097.441500
LATAM-Consumer        2252.677529
LATAM-Corporate       1075.994223
LATAM-Home Office      788.531853
US-Consumer           2603.736252
US-Corporate          1853.568607
US-Home Office        1062.397424
dtype: float64
```

In [40]:

```
train_std = np.std(train)
train_std #standard deviation
```

Out[40]:

```
Market_Segment
APAC-Consumer         2518.944225
APAC-Corporate        1871.535073
APAC-Home Office      1446.445137
Africa-Consumer       1141.894252
Africa-Corporate       709.261893
Africa-Home Office     662.629728
Canada-Consumer        339.601099
Canada-Corporate       132.146175
Canada-Home Office     302.526945
EMEA-Consumer         1128.515779
EMEA-Corporate        1167.958953
EMEA-Home Office       747.714036
EU-Consumer           2348.762579
EU-Corporate          1552.403019
EU-Home Office        1223.296718
LATAM-Consumer        1533.362508
LATAM-Corporate        947.154491
LATAM-Home Office     1059.547064
US-Consumer           2851.858407
US-Corporate          1904.002356
US-Home Office        1293.079478
dtype: float64
```

In [41]:

```
train_CoV = train_std/train_mean
train_CoV # coefficient of varience
```

Out[41]:

```
Market_Segment
APAC-Consumer         0.596404
APAC-Corporate        0.731926
APAC-Home Office      1.048817
Africa-Consumer       1.429335
Africa-Corporate      1.664827
Africa-Home Office    1.989866
Canada-Consumer       1.476093
Canada-Corporate      1.197220
Canada-Home Office    2.188300
EMEA-Consumer         2.716992
EMEA-Corporate        6.779639
EMEA-Home Office      6.066684
EU-Consumer           0.647485
EU-Corporate          0.689346
EU-Home Office        1.114681
LATAM-Consumer        0.680684
LATAM-Corporate       0.880260
LATAM-Home Office     1.343696
US-Consumer           1.095295
US-Corporate          1.027209
US-Home Office        1.217133
dtype: float64
```

In [42]:

```
CoV = pd.DataFrame({'Mean' : train_mean, 'Std_Dev' : train_std, 'Coeff_of_Var' : train_CoV})
CoV
```

Out[42]:

Market_Segment	Mean	Std_Dev	Coeff_of_Var
APAC-Consumer	4223.553586	2518.944225	0.596404
APAC-Corporate	2556.998957	1871.535073	0.731926
APAC-Home Office	1379.120743	1446.445137	1.048817
Africa-Consumer	798.898929	1141.894252	1.429335
Africa-Corporate	426.027286	709.261893	1.664827
Africa-Home Office	333.002143	662.629728	1.989866
Canada-Consumer	230.067500	339.601099	1.476093
Canada-Corporate	110.377500	132.146175	1.197220
Canada-Home Office	138.247500	302.526945	2.188300
EMEA-Consumer	415.354786	1128.515779	2.716992
EMEA-Corporate	172.274500	1167.958953	6.779639
EMEA-Home Office	123.249214	747.714036	6.066684
EU-Consumer	3627.517036	2348.762579	0.647485
EU-Corporate	2251.993036	1552.403019	0.689346
EU-Home Office	1097.441500	1223.296718	1.114681
LATAM-Consumer	2252.677529	1533.362508	0.680684
LATAM-Corporate	1075.994223	947.154491	0.880260
LATAM-Home Office	788.531853	1059.547064	1.343696
US-Consumer	2603.736252	2851.858407	1.095295
US-Corporate	1853.568607	1904.002356	1.027209
US-Home Office	1062.397424	1293.079478	1.217133

In [43]:

```
CoV.sort_values('Coeff_of_Var')
```

Out[43]:

Market_Segment	Mean	Std_Dev	Coeff_of_Var
APAC-Consumer	4223.553586	2518.944225	0.596404
EU-Consumer	3627.517036	2348.762579	0.647485
LATAM-Consumer	2252.677529	1533.362508	0.680684
EU-Corporate	2251.993036	1552.403019	0.689346
APAC-Corporate	2556.998957	1871.535073	0.731926
LATAM-Corporate	1075.994223	947.154491	0.880260
US-Corporate	1853.568607	1904.002356	1.027209
APAC-Home Office	1379.120743	1446.445137	1.048817
US-Consumer	2603.736252	2851.858407	1.095295
EU-Home Office	1097.441500	1223.296718	1.114681
Canada-Corporate	110.377500	132.146175	1.197220
US-Home Office	1062.397424	1293.079478	1.217133
LATAM-Home Office	788.531853	1059.547064	1.343696
Africa-Consumer	798.898929	1141.894252	1.429335
Canada-Consumer	230.067500	339.601099	1.476093
Africa-Corporate	426.027286	709.261893	1.664827
Africa-Home Office	333.002143	662.629728	1.989866
Canada-Home Office	138.247500	302.526945	2.188300
EMEA-Consumer	415.354786	1128.515779	2.716992
EMEA-Home Office	123.249214	747.714036	6.066684
EMEA-Corporate	172.274500	1167.958953	6.779639

In [44]:

```
# Filter the main data frame such for APAC-Consumer market segment
data_apac_con = data[data['Market_Segment'] == 'APAC-Consumer']
data_apac_con
```

Out[44]:

	Order Date	Sales	Profit	Market_Segment
2	2013-10	5175.1710	919.9710	APAC-Consumer
6	2011-11	1822.0800	564.8400	APAC-Consumer
7	2012-04	5244.8400	996.4800	APAC-Consumer
14	2013-06	3701.5200	1036.0800	APAC-Consumer
29	2012-02	1878.7200	582.3600	APAC-Consumer
...	...	...	...	...
51215	2013-01	38.9709	-32.3991	APAC-Consumer
51234	2014-11	6.9000	-0.8400	APAC-Consumer
51235	2011-11	17.2800	-13.9200	APAC-Consumer
51236	2013-07	30.6180	1.0080	APAC-Consumer
51252	2013-08	4.5600	0.0000	APAC-Consumer

5699 rows × 4 columns

In [45]:

```
# Group the data by Order Date for sum of Sales
data1 = data_apac_con.groupby(['Order Date'])['Sales'].sum()
data1
```

Out[45]:

```
Order Date
2011-01    15711.7125
2011-02    12910.8588
2011-03    19472.5632
2011-04    15440.3046
2011-05    24348.9723
2011-06    27260.0196
2011-07    15842.8317
2011-08    22012.2366
2011-09    34613.1849
2011-10    36472.0470
2011-11    37722.6039
2011-12    37846.9911
2012-01    31280.8635
2012-02    24985.6881
2012-03    14241.1761
2012-04    20926.4721
2012-05    32608.6212
2012-06    39710.0352
2012-07     8389.7316
2012-08    48444.7977
2012-09    28193.2236
2012-10    56743.0833
2012-11    51967.0140
2012-12    47343.6267
2013-01    31328.2620
2013-02    20999.7378
2013-03    28414.6209
2013-04    20508.7197
2013-05    47979.6000
2013-06    70436.8089
2013-07    34596.8247
2013-08    45251.6439
2013-09    51513.2235
2013-10    53448.5532
2013-11    47812.7784
2013-12    51717.6204
2014-01    37924.3296
2014-02    21584.5086
2014-03    46340.1096
2014-04    32897.5062
2014-05    53544.8328
2014-06    51483.7011
2014-07    36524.3028
2014-08    63521.7729
2014-09    44477.2662
2014-10    77379.8286
2014-11    82286.3583
2014-12    60292.1310
Freq: M, Name: Sales, dtype: float64
```

In [46]:

```
# Time series Decomposition
```

In [47]:

```
data1 = pd.DataFrame(data1)
```

In [48]:

```
data1.index = data1.index.to_timestamp() # Converting the index to timestamp
```

In [49]:

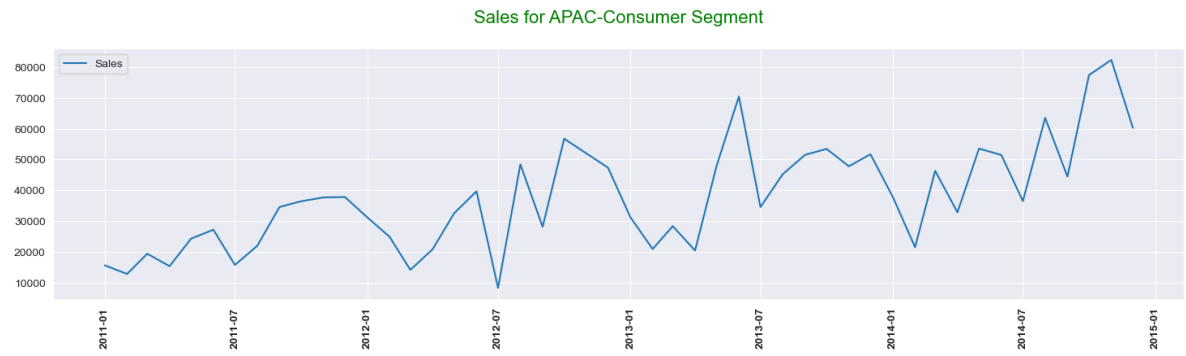
```
data1.index.dtype
```

Out[49]:

```
dtype('<M8[ns]')
```

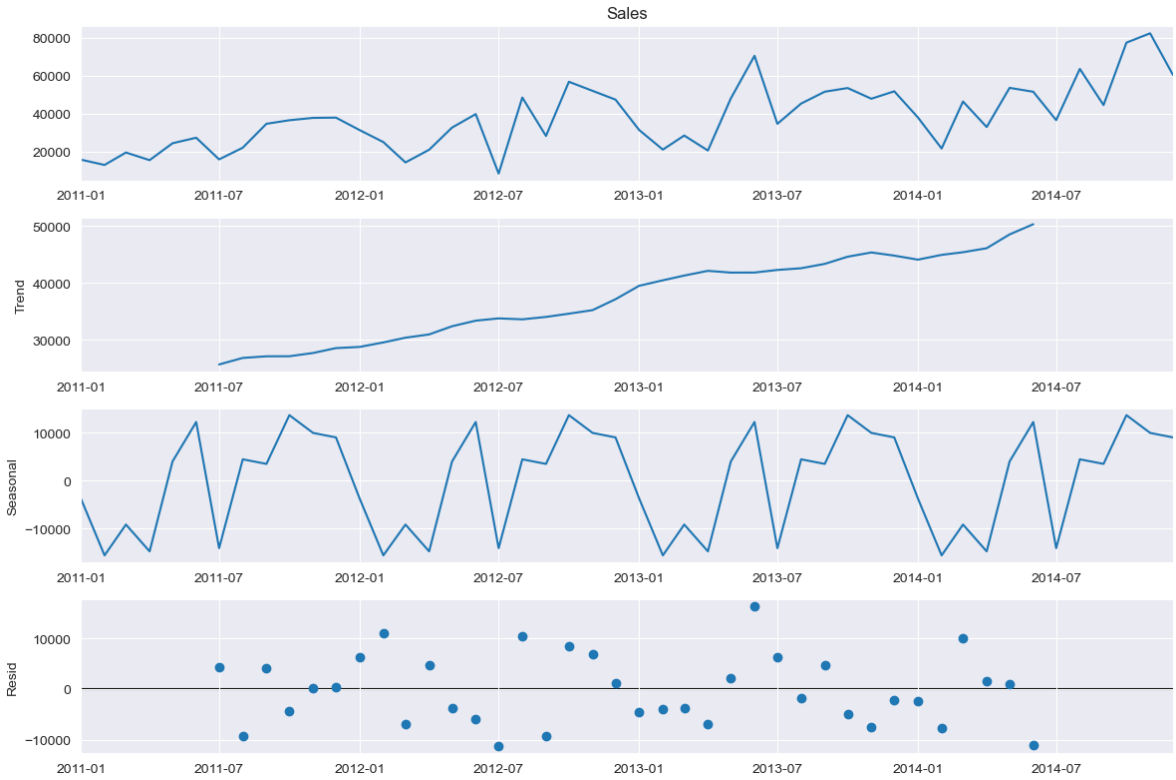
In [50]:

```
plt.figure(figsize=(18,4))
plt.plot(data1, label='Sales')
plt.legend(loc='best')
plt.title('Sales for APAC-Consumer Segment\n', fontdict={'fontsize': 16, 'fontweight' : 5,
plt.xticks(rotation = 90,fontweight="bold")
plt.show()
```



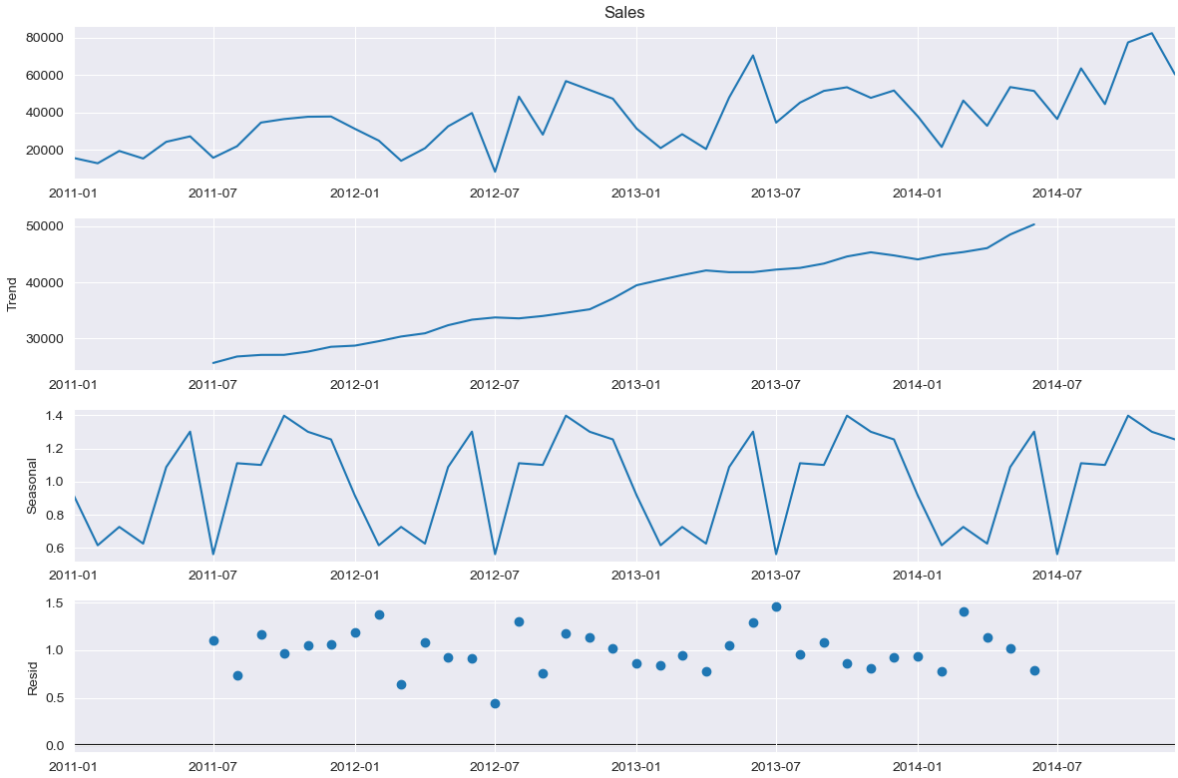
In [51]:

```
from pylab import rcParams
import statsmodels.api as sm
rcParams['figure.figsize'] = 12, 8
decomposition = sm.tsa.seasonal_decompose(data1.Sales, model='additive') # additive seasona
fig = decomposition.plot()
plt.show()
```



In [52]:

```
decomposition = sm.tsa.seasonal_decompose(data1.Sales, model='multiplicative') # multiplica
fig = decomposition.plot()
plt.show()
```



In [53]:

```
# Split the data into train and test sets.
```

In [54]:

```
train_len = 42
train = data1[0 : train_len]
test = data1[train_len : ]
```

In [55]:

```
train.head()
```

Out[55]:

Order Date	Sales
2011-01-01	15711.7125
2011-02-01	12910.8588
2011-03-01	19472.5632
2011-04-01	15440.3046
2011-05-01	24348.9723

In [56]:

```
test
```

Out[56]:

Order Date	Sales
2014-07-01	36524.3028
2014-08-01	63521.7729
2014-09-01	44477.2662
2014-10-01	77379.8286
2014-11-01	82286.3583
2014-12-01	60292.1310

In [57]:

```
# Naive method
```

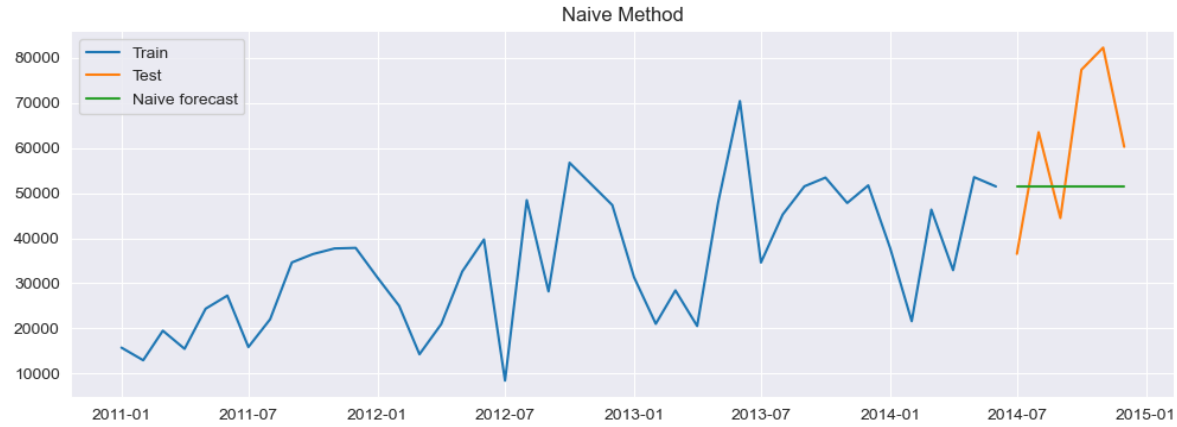
In [58]:

```
y_hat_naive = test.copy()
y_hat_naive['naive_forecast'] = train['Sales'][train_len-1]
```

In [59]:

```
# Plot train, test and forecast

plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
plt.plot(y_hat_naive['naive_forecast'], label='Naive forecast')
plt.legend(loc='best')
plt.title('Naive Method')
plt.show()
```



In [60]:

```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(test['Sales'], y_hat_naive['naive_forecast'])).round(2)
mape = np.round(np.mean(np.abs(test['Sales']-y_hat_naive['naive_forecast'])/test['Sales'])*

results = pd.DataFrame({'Method':['Naive method'], 'MAPE': [mape], 'RMSE': [rmse]})
results = results[['Method', 'RMSE', 'MAPE']]
results
```

Out[60]:

	Method	RMSE	MAPE
0	Naive method	18774.05	26.86

In [61]:

```
# Simple average method
```

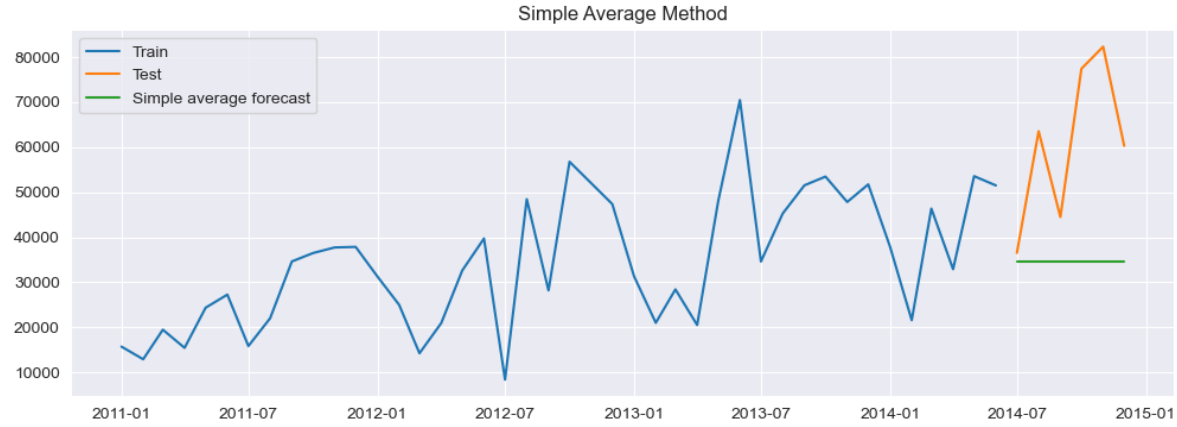
In [62]:

```
y_hat_avg = test.copy()
y_hat_avg['avg_forecast'] = train['Sales'].mean()
```

In [63]:

```
# Plot train, test and forecast

plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
plt.plot(y_hat_avg['avg_forecast'], label='Simple average forecast')
plt.legend(loc='best')
plt.title('Simple Average Method')
plt.show()
```



In [64]:

```
# Calculate RMSE and MAPE

rmse = np.sqrt(mean_squared_error(test['Sales'], y_hat_avg['avg_forecast'])).round(2)
mape = np.round(np.mean(np.abs(test['Sales']-y_hat_avg['avg_forecast'])/test['Sales'])*100,

tempResults = pd.DataFrame({'Method':['Simple average method'], 'RMSE': [rmse],'MAPE': [map
results = pd.concat([results, tempResults])
results = results[['Method', 'RMSE', 'MAPE']]
results
```

Out[64]:

	Method	RMSE	MAPE
0	Naive method	18774.05	26.86
0	Simple average method	30846.00	38.18

In [65]:

```
# Simple moving average method
```

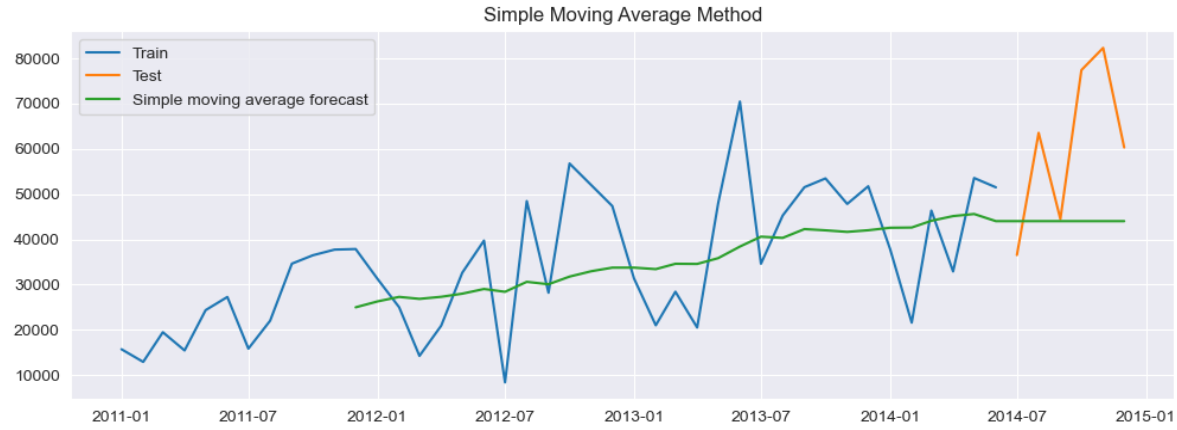
In [66]:

```
y_hat_sma = data1.copy()
ma_window = 12
y_hat_sma['sma_forecast'] = data1['Sales'].rolling(ma_window).mean()
y_hat_sma['sma_forecast'][train_len:] = y_hat_sma['sma_forecast'][train_len-1]
```

In [67]:

```
# Plot train, test and forecast

plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
plt.plot(y_hat_sma['sma_forecast'], label='Simple moving average forecast')
plt.legend(loc='best')
plt.title('Simple Moving Average Method')
plt.show()
```



In [68]:

```
# Calculate RMSE and MAPE

rmse = np.sqrt(mean_squared_error(test['Sales'], y_hat_sma['sma_forecast'][train_len:])).ro
mape = np.round(np.mean(np.abs(test['Sales']-y_hat_sma['sma_forecast'][train_len:])/test['S

tempResults = pd.DataFrame({'Method':['Simple moving average forecast'], 'RMSE': [rmse],'MA
results = pd.concat([results, tempResults])
results = results[['Method', 'RMSE', 'MAPE']]
results
```

Out[68]:

	Method	RMSE	MAPE
0	Naive method	18774.05	26.86
0	Simple average method	30846.00	38.18
0	Simple moving average forecast	23383.65	28.15

In [69]:

```
# Simple exponential smoothing
```

In [70]:

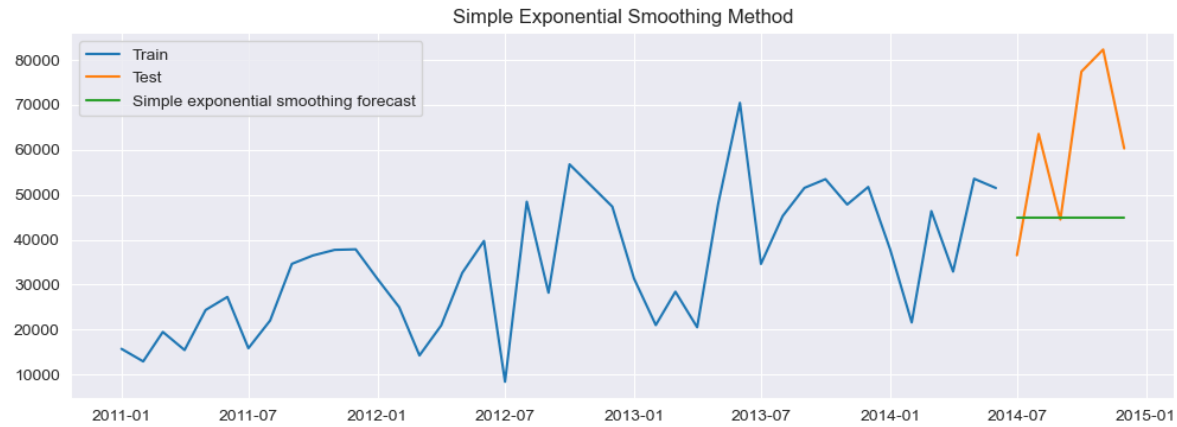
```
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
model = SimpleExpSmoothing(train['Sales'])
model_fit = model.fit(optimized=True)
model_fit.params
y_hat_ses = test.copy()
y_hat_ses['ses_forecast'] = model_fit.forecast(len(test))
```

```
C:\Users\hp\anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.p
y:915: ConvergenceWarning: Optimization failed to converge. Check mle_retval
s.
  warnings.warn(
```

In [71]:

```
# Plot train, test, forecast

plt.figure(figsize=(12,4))
plt.plot(train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
plt.plot(y_hat_ses['ses_forecast'], label='Simple exponential smoothing forecast')
plt.legend(loc='best')
plt.title('Simple Exponential Smoothing Method')
plt.show()
```



In [72]:

```
# Calculate RMSE and MAPE

rmse = np.sqrt(mean_squared_error(test['Sales'], y_hat_ses['ses_forecast'])).round(2)
mape = np.round(np.mean(np.abs(test['Sales']-y_hat_ses['ses_forecast'])/test['Sales'])*100,

tempResults = pd.DataFrame({'Method':['Simple exponential smoothing forecast'], 'RMSE': [rm
results = pd.concat([results, tempResults])
results
```

Out[72]:

	Method	RMSE	MAPE
0	Naive method	18774.05	26.86
0	Simple average method	30846.00	38.18
0	Simple moving average forecast	23383.65	28.15
0	Simple exponential smoothing forecast	22824.62	27.70

In [73]:

```
# Holt's method with trend
```

In [74]:

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
model = ExponentialSmoothing(np.asarray(train['Sales']) ,seasonal_periods=12 ,trend='additi
model_fit = model.fit(optimized=True)
print(model_fit.params)
y_hat_holt = test.copy()
y_hat_holt['holt_forecast'] = model_fit.forecast(len(test))
```

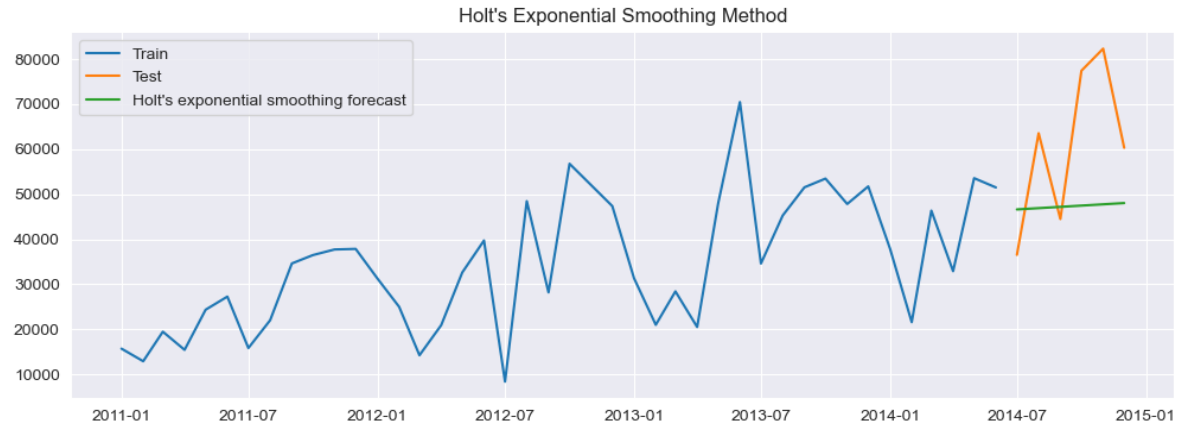
```
{'smoothing_level': 0.1464285714285714, 'smoothing_trend': 0.128125, 'smooth
ing_seasonal': nan, 'damping_trend': nan, 'initial_level': 10555.93015999999
6, 'initial_trend': 2155.007810909092, 'initial_seasons': array([], dtype=fl
oat64), 'use_boxcox': False, 'lamda': None, 'remove_bias': False}

C:\Users\hp\anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.p
y:915: ConvergenceWarning: Optimization failed to converge. Check mle_retval
s.
  warnings.warn(
```

In [75]:

```
# Plot train, test, forecast

plt.figure(figsize=(12,4))
plt.plot( train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
plt.plot(y_hat_holt['holt_forecast'], label='Holt\'s exponential smoothing forecast')
plt.legend(loc='best')
plt.title('Holt\'s Exponential Smoothing Method')
plt.show()
```



In [76]:

```
# Calculate RMSE and MAPE

rmse = np.sqrt(mean_squared_error(test['Sales'], y_hat_holt['holt_forecast'])).round(2)
mape = np.round(np.mean(np.abs(test['Sales']-y_hat_holt['holt_forecast'])/test['Sales'])*10

tempResults = pd.DataFrame({'Method':['Holt\'s exponential smoothing method'], 'RMSE': [rms
results = pd.concat([results, tempResults])
results = results[['Method', 'RMSE', 'MAPE']]
results
```

Out[76]:

	Method	RMSE	MAPE
0	Naive method	18774.05	26.86
0	Simple average method	30846.00	38.18
0	Simple moving average forecast	23383.65	28.15
0	Simple exponential smoothing forecast	22824.62	27.70
0	Holt's exponential smoothing method	20916.20	26.81

In [77]:

```
# Holt Winters' additive method with trend and seasonality
```

In [78]:

```
y_hat_hwa = test.copy()
model = ExponentialSmoothing(np.asarray(train['Sales']) ,seasonal_periods=12 ,trend='add',
model_fit = model.fit(optimized=True)
print(model_fit.params)
y_hat_hwa['hw_forecast'] = model_fit.forecast(len(test))
```

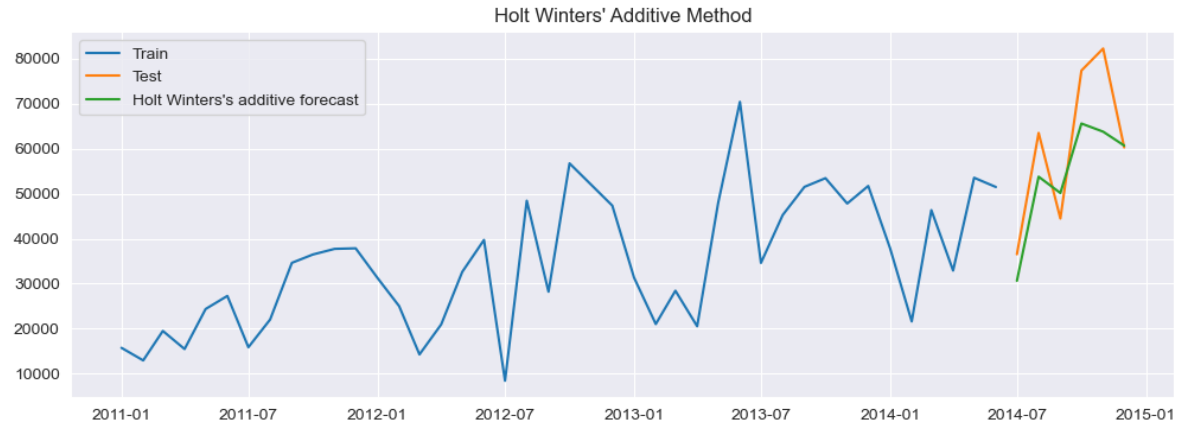
```
{'smoothing_level': 0.005, 'smoothing_trend': 0.005, 'smoothing_seasonal':
0.0001, 'damping_trend': nan, 'initial_level': 25191.878854999984, 'initial_
trend': 549.1511154545474, 'initial_seasons': array([ -3046.57535417, -1223
3.28766042, -14734.34056667, -16050.40451667,
         2962.22100208,  17242.34903958, -17814.35478542,   4804.01762083,
          623.89353958,  15536.36958958,  13175.71666458,   9534.39542708]),
'use_boxcox': False, 'lamda': None, 'remove_bias': False}

C:\Users\hp\anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.p
y:915: ConvergenceWarning: Optimization failed to converge. Check mle_retval
s.
  warnings.warn(
```

In [79]:

```
# Plot train, test, forecast

plt.figure(figsize=(12,4))
plt.plot( train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
plt.plot(y_hat_hwa['hw_forecast'], label='Holt Winters\'s additive forecast')
plt.legend(loc='best')
plt.title('Holt Winters\' Additive Method')
plt.show()
```



In [80]:

```
# Calculate RMSE and MAPE

rmse = np.sqrt(mean_squared_error(test['Sales'], y_hat_hwa['hw_forecast'])).round(2)
mape = np.round(np.mean(np.abs(test['Sales']-y_hat_hwa['hw_forecast'])/test['Sales'])*100,2

tempResults = pd.DataFrame({'Method':['Holt Winters\' additive method'], 'RMSE': [rmse],'MA
results = pd.concat([results, tempResults])
results = results[['Method', 'RMSE', 'MAPE']]
results
```

Out[80]:

	Method	RMSE	MAPE
0	Naive method	18774.05	26.86
0	Simple average method	30846.00	38.18
0	Simple moving average forecast	23383.65	28.15
0	Simple exponential smoothing forecast	22824.62	27.70
0	Holt's exponential smoothing method	20916.20	26.81
0	Holt Winters' additive method	10350.33	13.77

In [81]:

```
#Holt Winter's multiplicative method with trend and seasonality
```

In [82]:

```
y_hat_hwm = test.copy()
model = ExponentialSmoothing(np.asarray(train['Sales']) ,seasonal_periods=12 ,trend='add',
model_fit = model.fit(optimized=True)
print(model_fit.params)
y_hat_hwm['hw_forecast'] = model_fit.forecast(len(test))
```

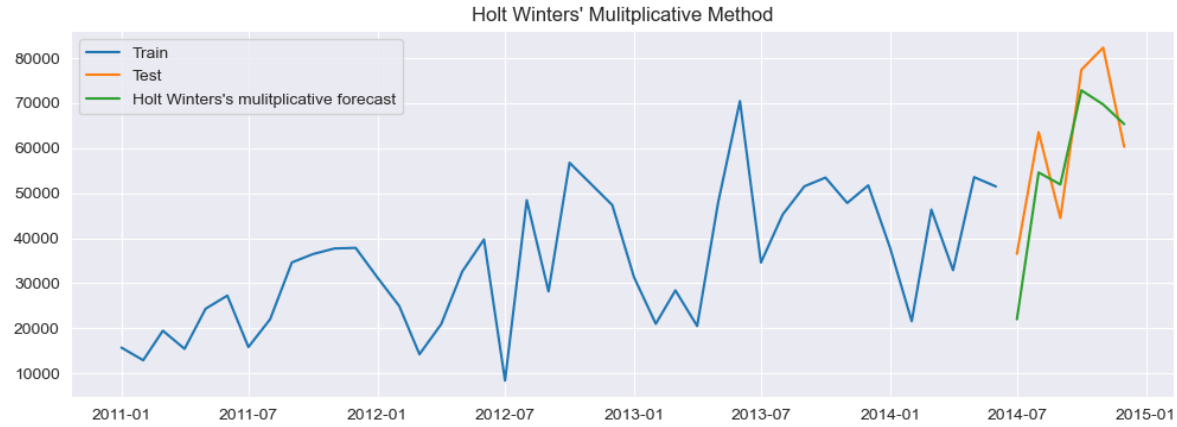
```
{'smoothing_level': 0.005, 'smoothing_trend': 0.005, 'smoothing_seasonal':
0.07107142857142858, 'damping_trend': nan, 'initial_level': 25191.8788549999
84, 'initial_trend': 549.1511154545474, 'initial_seasons': array([0.9307406
1, 0.67529696, 0.57227633, 0.57532186, 1.06552983,
       1.42137084, 0.42860295, 1.11971051, 1.04228985, 1.47741289,
       1.40453147, 1.28691591]), 'use_boxcox': False, 'lamda': None, 'remove
_bias': False}

C:\Users\hp\anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.p
y:915: ConvergenceWarning: Optimization failed to converge. Check mle_retval
s.
  warnings.warn(
```

In [83]:

```
# Plot train, test, forecast

plt.figure(figsize=(12,4))
plt.plot( train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
plt.plot(y_hat_hwm['hw_forecast'], label='Holt Winters\'s mulitplicative forecast')
plt.legend(loc='best')
plt.title('Holt Winters\' Mulitplicative Method')
plt.show()
```



In [84]:

```
rmse = np.sqrt(mean_squared_error(test['Sales'], y_hat_hwm['hw_forecast'])).round(2)
mape = np.round(np.mean(np.abs(test['Sales']-y_hat_hwm['hw_forecast'])/test['Sales'])*100,2

tempResults = pd.DataFrame({'Method':['Holt Winters\' multiplicative method'], 'RMSE': [rms
results = pd.concat([results, tempResults])
results = results[['Method', 'RMSE', 'MAPE']]
results
```

Out[84]:

	Method	RMSE	MAPE
0	Naive method	18774.05	26.86
0	Simple average method	30846.00	38.18
0	Simple moving average forecast	23383.65	28.15
0	Simple exponential smoothing forecast	22824.62	27.70
0	Holt's exponential smoothing method	20916.20	26.81
0	Holt Winters' additive method	10350.33	13.77
0	Holt Winters' multiplicative method	9585.23	16.69