Time Series Sales Prediction Analysis

specific way of analyzing a sequence of data points collected over an interval of time

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
#use for load dataset into google colab
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Superstore.xls to Superstore.xls
```

In [26]:
```
df=pd.read_excel('Superstore.xls')
df.head()
```

Out[26]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale

5 rows × 21 columns

In [27]:
```
df.describe()
```

Out[27]:

	Row ID	Order Date	Ship Date	Postal Code	Sales	C
count	9994.000000	9994	9994	9994.000000	9994.000000	9994
mean	4997.500000	2016-04-30 00:07:12.259355648	2016-05-03 23:06:58.571142912	55190.379428	229.858001	3
min	1.000000	2014-01-03 00:00:00	2014-01-07 00:00:00	1040.000000	0.444000	1
25%	2499.250000	2015-05-23 00:00:00	2015-05-27 00:00:00	23223.000000	17.280000	2
50%	4997.500000	2016-06-26 00:00:00	2016-06-29 00:00:00	56430.500000	54.490000	3
75%	7495.750000	2017-05-14 00:00:00	2017-05-18 00:00:00	90008.000000	209.940000	5
max	9994.000000	2017-12-30 00:00:00	2018-01-05 00:00:00	99301.000000	22638.480000	14
std	2885.163629	NaN	NaN	32063.693350	623.245101	2

In [9]:
```
#show data types of all columns
df.dtypes
```

Out[9]:

	0
Row ID	int64
Order ID	object
Order Date	datetime64[ns]
Ship Date	datetime64[ns]
Ship Mode	object
Customer ID	object
Customer Name	object
Segment	object
Country	object
City	object
State	object
Postal Code	int64
Region	object
Product ID	object
Category	object
Sub-Category	object
Product Name	object
Sales	float64
Quantity	int64
Discount	float64
Profit	float64

dtype: object

In [10]:
```
#Show some basic information of dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: datetime64[ns](2), float64(3), int64(3), object(13)
memory usage: 1.6+ MB
```

In [11]:
```
df.isnull().sum()
```

Out[11]:

	0
Row ID	0
Order ID	0
Order Date	0
Ship Date	0
Ship Mode	0
Customer ID	0
Customer Name	0
Segment	0
Country	0
City	0
State	0
Postal Code	0
Region	0
Product ID	0
Category	0
Sub-Category	0
Product Name	0
Sales	0
Quantity	0
Discount	0
Profit	0

dtype: int64

In [12]:
```
#show all the columns
df.columns
```

Out[12]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'Stat
e',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

In [28]:
```
#Count values according to particular category or columns
df['Category'].value_counts()
```

Out[28]:

Category	count
Office Supplies	6026
Furniture	2121
Technology	1847

dtype: int64

In [29]:
```
#show only technology dataset  == is for comparision for values
100==100
tech=df[df['Category']=='Technology']
tech.head()
```

Out[29]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	Ci
7	8	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	L Angel
11	12	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	L Angel
19	20	CA-2014-143336	2014-08-27	2014-09-01	Second Class	ZD-21925	Zuschuss Donatelli	Consumer	United States	S Francis
26	27	CA-2016-121755	2016-01-16	2016-01-20	Second Class	EH-13945	Eric Hoffmann	Consumer	United States	L Angel
35	36	CA-2016-117590	2016-12-08	2016-12-10	First Class	GH-14485	Gene Hale	Corporate	United States	Richards

5 rows × 21 columns

In [18]:
```
tech.columns
```

Out[18]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'Stat
e',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

In [30]:
```
col=['Row ID', 'Order ID', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City',
'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Quantity', 'Discount', 'Profit']
#dorp the columns, axis=1 use for columns, inplace true for accept the
changes permanently
tech.drop(col,axis=1,inplace=True)
tech.head()
```

Out[30]:

	Order Date	Sales
7	2014-06-09	907.152
11	2014-06-09	911.424
19	2014-08-27	213.480
26	2016-01-16	90.570
35	2016-12-08	1097.544

In [31]:
```
#sorting the values
tech.sort_values(by='Order Date',ascending=True,inplace=True)
tech.head()
```

Out[31]:

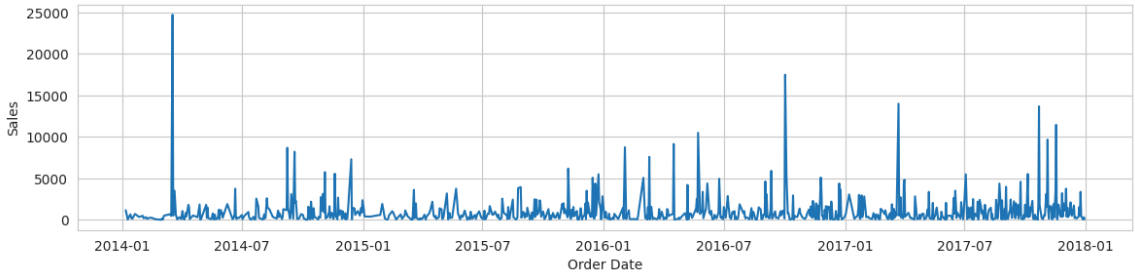
	Order Date	Sales
7478	2014-01-06	755.96
7477	2014-01-06	391.98
593	2014-01-09	31.20
765	2014-01-13	646.74
1913	2014-01-15	149.95

In [32]:
```
#total sales by order date and want to reset the index
tech_tot=tech.groupby(['Order Date'])['Sales'].sum().reset_index()
tech_tot.head()
```

Out[32]:

	Order Date	Sales
0	2014-01-06	1147.94
1	2014-01-09	31.20
2	2014-01-13	646.74
3	2014-01-15	149.95
4	2014-01-16	124.20

In [37]:
```
#show sales according to order date using seaborn chart
plt.figure(figsize=(14,3))  #Width and Height
sns.set_style('whitegrid')
#sns.set_style('darkgrid')
sns.lineplot(x='Order Date',y='Sales',data=tech_tot)
plt.show()
```



In [38]:
```
#sales prediction according to order date for next years using seaborn
chart
from statsmodels.tsa.holtwinters import ExponentialSmoothing

#Assuming tech_tot is your dataframe with order date and sales
tech_tot['Order Date'] = pd.to_datetime(tech_tot['Order Date'])
tech_tot.set_index('Order Date', inplace=True)
tech_tot
```

Out[38]:

Order Date	Sales
2014-01-06	1147.940
2014-01-09	31.200
2014-01-13	646.740
2014-01-15	149.950
2014-01-16	124.200
...	...
2017-12-25	401.208
2017-12-27	164.388
2017-12-28	14.850
2017-12-29	302.376
2017-12-30	90.930

824 rows × 1 columns

In [39]:
```
#Aggregate sales by year  Y=Year
annual_sales=tech_tot['Sales'].resample('Y').sum()
annual_sales
```

Out[39]:

Order Date	Sales
2014-12-31	175278.233
2015-12-31	162780.809
2016-12-31	226364.180
2017-12-31	271730.811

dtype: float64

In [42]:
```
#Fit the model -using Exponential smoothing for forecasting
model=ExponentialSmoothing(annual_sales,trend='add',seasonal=None,seasona
l_periods=None).fit()

#predict sales for the next year
future_sales=model.forecast(steps=1)
print(future_sales)
```

```
2018-12-31    323035.988404
Freq: A-DEC, dtype: float64

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/holtwinters/model.
py:918: ConvergenceWarning: Optimization failed to converge. Check mle_ret
vals.
  warnings.warn(
```

In [46]:
```
#plotting the historical and predicted sales
plt.figure(figsize=(10,6))
plt.plot(annual_sales,label='Historical Sales')
plt.plot(future_sales,label='Predicted Sales',linestyle='--',marker='o')
plt.xlabel('Year')
plt.ylabel('Sales')
plt.title('Historical and Predicted Sales')
plt.legend()
plt.show()
```