RAMAKRISHNA MISSION
VIVEKANANDA EDUCATIONAL AND RESEARCH INSTITUTE
(Accredited by NAAC with 'A++' Grade)
Coimbatore – 641 020

SCHOOL OF MATHEMATICAL SCIENCE

DECEMBER– 2022

DEPARTMENT OF COMPUTER SCIENCE
CENTRE OF DATA SCIENCE

NAME	**: JAGHAN T**
REG. NO	**: H21MSDS006**
PROGRAMME	**: M.Sc. (Data Science)**
SEMESTER	**: IIInd Semester**
PROJECT TITLE	**: Sales Forecasting Time Series Analysis**
COURSE CODE	**: 21DS2P06**

RAMAKRISHNA MISSION
VIVEKANANDA EDUCATIONAL AND RESEARCH INSTITUTE
(Accredited by NAAC with ‘A++’ Grade)
Coimbatore – 641 020

SCHOOL OF MATHEMATICAL SCIENCE

DECEMBER – 2022

DEPARTMENT OF COMPUTER SCIENCE
CENTRE OF DATA SCIENCE

Bonafide Certificate

This is to certify that the project work done by **_ JAGHAN T___** (**H21MSDS006**) entitled "**Sales Forecasting Time Series Analysis**" in his Second Semester during the academic year 2021-2022. Submitted for the Semester viva-voice Examination held on **11-12-2022**

Staff In-Charge   Head of the Department

Internal Examiner   External Examiner

DECLARATION

I hereby declare that this project entitled "**Sales Forecasting Time Series Analysis**" submitted to Ramakrishna Mission Vivekananda Educational and Research Institute, Coimbatore-20. is partial fulfillment of the requirement of the degree in M.Sc., (Data Science) is a record of the original project done by me under the guidance of Sir.V.Dineshkumar, M.Sc., M.Phil., (Ph.D)., Assistant Professor, Department of Computer Science, Ramakrishna Mission Vivekananda Educational And Research Institute, Coimbatore - 641 020.

Place : Coimbatore

Date : 11-12-2022

Signature of the Candidate

JAGHAN T

(H21MSDS006)

ACKNOWLEDGEMENT

My pranams to **Rev. Swami Garishthananda**, Secretary, Ramakrishna Mission Vidyalaya and **Rev. Swami Anapekshanada**, Asst. Administrative Head, Department of Computer Science, Ramakrishna Mission Vivekananda Educational and Research Institute, Coimbatore - 20 for providing me facilities and encouragement to complete the project work.

I express my profound gratitude to **Dr. R.Sridhar,** Professor & Head, Department of Computer Science, RKMVERI, Coimbatore-20 for his whole hearted encouragement and timely help.

I placed on record my heartfelt thanks and gratitude to **Sri. V.Dineshkumar,** Assistant Professor, Department of Computer Science, RKMVERI, Coimbatore-20 under whose guidance the work has been done. It is a matter of joy that without his valuable suggestions, able guidance, scholarly touch and piercing insight that he offered me in each and every stage of this study, coupled with his unreserved, sympathetic and encouraging attitude, this thesis could not have been presented in this manner.

I would like to thank everyone who helped and motivated to complete this project work.

SYNOPSIS

Forecasts aren't just for meteorologists. Governments forecast economic growth. Scientists attempt to predict the future population. And businesses forecast product demand—a common task of professional data scientists. Forecasts are especially relevant to brick-and-mortar grocery stores, which must dance delicately with how much inventory to buy. Predict a little over, and grocers are stuck with overstocked, perishable goods. Guess a little under, and popular items quickly sell out, leading to lost revenue and upset customers. More accurate forecasting, thanks to machine learning, could help ensure retailers please customers by having just enough of the right products at the right time.

Current subjective forecasting methods for retail have little data to back them up and are unlikely to be automated. The problem becomes even more complex as retailers add new locations with unique needs, new products, ever-transitioning seasonal tastes, and unpredictable product marketing.
Time series analysis deals with time seriesbased data to extract patterns for predictions and other characteristics of the data. It uses a model for forecasting future values in a smalltime frame based on previous observations. It is widely used for non-stationary data, such as economic data, weather data, stock prices, and retail sales forecasting.

About the Superstore data

Today we are working with 4 years of sales data in the United States.

Data dictionary of the dataset is appended below.

Field	Description
Row ID	Unique row ID
Order **ID**	Unique identifier of each order
Order Date	Date of order
Ship Date	Date that product was shipped
Ship Mode	Shipping types (for e.g., Second Class, Standard Class, First Class, Same Day)
Customer ID	Unique identification of customer
Customer Name	Name of customer
Segment	Customer segment (for e.g., Consumer, Corporate, Home office
Country	Country where customer ordered from
City	City where customer is from
State	State where customer is from
Postal Code	Postal code where customer is from
Region	Region where customer is from
Product ID	Unique identifier of each product
Category	Overall category that product fits into (for e.g., Furniture, Office Supplies, Technology)
Sub-Category	Sub-category of category (for e.g., Bookcases, Chairs, Labels)
Product Name	Name of product
Sales	Gross sales per order

Loading Libraries and Dataset

```
## Loading Libraries

import numpy as np
import pandas as pd
from math import sqrt

## For visualization
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

import warnings
warnings.filterwarnings('ignore')

# set theme
sns.set_style('whitegrid')
plt.rc('font', size=14)
plt.style.use('tableau-colorblind10')


# Load the dataset
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dataset/Stor
esales.csv')
sales = data
sales.head()
```

Data Analysis

```
# see the null values
sales.isna().sum()
```

```
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     11
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
dtype: int64
```

```
# summary statistics
sales.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Row ID	9800.0	4900.500000	2829.160653	1.000	2450.750	4900.50	7350.250	9800.00
Postal Code	9789.0	55273.322403	32041.223413	1040.000	23223.000	58103.00	90008.000	99301.00
Sales	9800.0	230.769059	626.651875	0.444	17.248	54.49	210.605	22638.48

```
# shape
sales.shape
```

```
(9800, 18)
```

```
# data types
sales.dtypes
```

```
Row ID              int64
Order ID           object
Order Date         object
Ship Date          object
Ship Mode          object
Customer ID        object
Customer Name      object
Segment            object
Country            object
City               object
State              object
Postal Code       float64
Region             object
Product ID         object
Category           object
Sub-Category       object
Product Name       object
Sales             float64
dtype: object
```

- We can see that Data columns is of Object types, Let's change it into Date datatype.

```
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9800 entries, 0 to 9799
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9800 non-null   int64
 1   Order ID       9800 non-null   object
 2   Order Date     9800 non-null   object
 3   Ship Date      9800 non-null   object
 4   Ship Mode      9800 non-null   object
 5   Customer ID    9800 non-null   object
 6   Customer Name  9800 non-null   object
 7   Segment        9800 non-null   object
 8   Country        9800 non-null   object
 9   City           9800 non-null   object
 10  State          9800 non-null   object
 11  Postal Code    9789 non-null   float64
 12  Region         9800 non-null   object
 13  Product ID     9800 non-null   object
 14  Category       9800 non-null   object
 15  Sub-Category   9800 non-null   object
 16  Product Name   9800 non-null   object
 17  Sales          9800 non-null   float64
dtypes: float64(2), int64(1), object(15)
memory usage: 1.3+ MB
```

```
# change datatype of Date columns .
sales["order_date"] = pd.to_datetime(sales["Order Date"], dayfirst=True
)
sales["ship_date"] = pd.to_datetime(sales["Ship Date"], dayfirst=True)
```

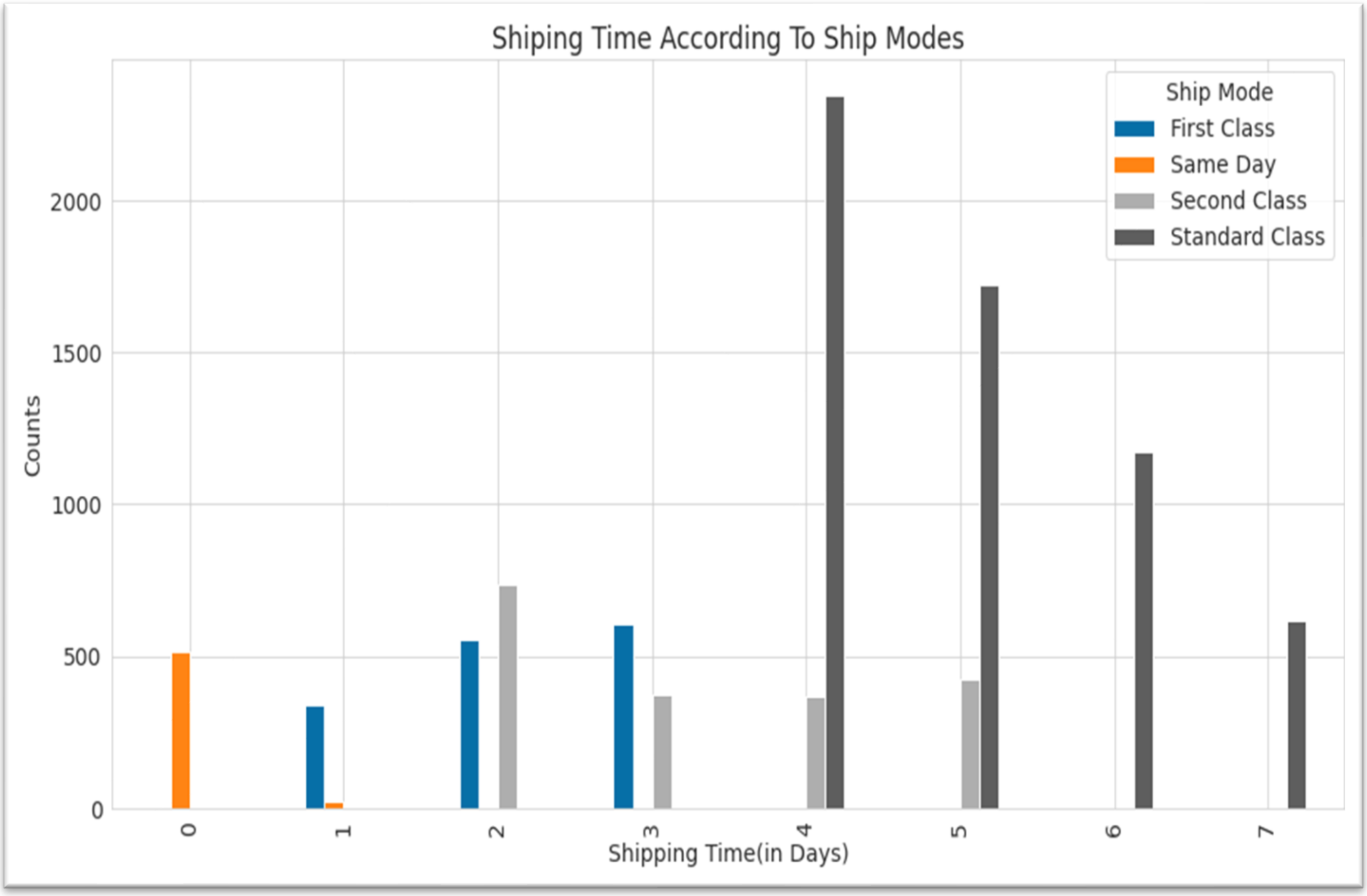
```
# Create 'Year-Month',. 'year' and 'month' columns .
sales["YearMonth"] = sales["order_date"].apply(lambda x: x.strftime("%Y
-%m"))
sales["year"] = sales["order_date"].dt.year
sales["month"] = sales["order_date"].dt.month_name()

# Create a column for Number of Days require to ship the product
sales["shipInDays"] = (sales["ship_date"] - sales["order_date"]).dt.day
s

# Viewing first five rows of data .
sales.head()
```

index	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID
0	1	CA-2017-152156	08/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South	FUR-BO-10001798
1	2	CA-2017-152156	08/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South	FUR-CH-10000454
2	3	CA-2017-138688	12/06/2017	16/06/2017	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036.0	West	OFF-LA-10000240
3	4	US-2016-108966	11/10/2016	18/10/2016	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South	FUR-TA-10000577
4	5	US-2016-108966	11/10/2016	18/10/2016	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South	OFF-ST-10000760

5	6	CA-2015-115812	09/06/2015	14/06/2015	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	Furniture	Furnishings	Eldon Expressions Wood and Plastic Desk Access...
6	7	CA-2015-115812	09/06/2015	14/06/2015	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	Office Supplies	Art	Newell 322
7	8	CA-2015-115812	09/06/2015	14/06/2015	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	Technology	Phones	Mitel 5320 IP Phone VoIP phone
8	9	CA-2015-115812	09/06/2015	14/06/2015	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	Office Supplies	Binders	DXL Angle-View Binders with Locking Rings by S...
9	10	CA-2015-115812	09/06/2015	14/06/2015	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	Office Supplies	Appliances	Belkin F5C206VTEL 6 Outlet Surge
10	11	CA-2015-115812	09/06/2015	14/06/2015	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	Furniture	Tables	Chromcraft Rectangular Conference Tables
11	12	CA-2015-115812	09/06/2015	14/06/2015	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	Technology	Phones	Konftel 250 Conference phone - Charcoal black

12	13	CA-2018-114412	15/04/2018	20/04/2018	Standard Class	AA-10480	Andrew Allen	Consumer	United States	Concord	...	Office Supplies	Paper	Xerox 1967	15.5520
13	14	CA-2017-161389	05/12/2017	10/12/2017	Standard Class	IM-15070	Irene Maddox	Consumer	United States	Seattle	...	Office Supplies	Binders	Fellowes PB200 Plastic Comb Binding Machine	407.9760
14	15	US-2016-118983	22/11/2016	26/11/2016	Standard Class	HP-14815	Harold Pawlan	Home Office	United States	Fort Worth	...	Office Supplies	Appliances	Holmes Replacement Filter for HEPA Air Cleaner...	68.8100
15	16	US-2016-118983	22/11/2016	26/11/2016	Standard Class	HP-14815	Harold Pawlan	Home Office	United States	Fort Worth	...	Office Supplies	Binders	Storex DuraTech Recycled Plastic Frosted Binders	2.5440
16	17	CA-2015-105893	11/11/2015	18/11/2015	Standard Class	PK-19075	Pete Kriz	Consumer	United States	Madison	...	Office Supplies	Storage	Stur-D-Stor Shelving, Vertical 5-Shelf: 72"H x...	665.8800
17	18	CA-2015-167164	13/05/2015	15/05/2015	Second Class	AG-10270	Alejandro Grove	Consumer	United States	West Jordan	...	Office Supplies	Storage	Fellowes Super Stor/Drawer	55.5000
18	19	CA-2015-143336	27/08/2015	01/09/2015	Second Class	ZD-21925	Zuschuss Donatelli	Consumer	United States	San Francisco	...	Office Supplies	Art	Newell 341	8.5600

```python
# Shippingh Time observation accourding to ShipModes.

a = df.groupby(by = ["shipInDays", "Ship Mode"]).count()["Order ID"]

# plot
a.unstack().plot(kind="bar", figsize=(15,8))

plt.xlabel("Shipping Time(in Days)")
plt.ylabel("Counts")
plt.title("Shiping Time According To Ship Modes")


plt.show()
```



```python
# Let's consider standard shipping days is 4. if shipping time is greater then 4 days then consider it as delayed.

sales["is_delayed"] = sales.shipInDays > 4

# Feature columns
features = ["Order ID", "Customer ID", "Product ID", "order_date", "ship_date", "Product Name", "Country", "Region", 'State', "City",
            "Segment", "Category", "Sub-Category", "Ship Mode", "YearMonth", "year", "month", "shipInDays", "is_delayed", "Sales"]

df = sales[features]
df.head()
```

	Order ID	Customer ID	Product ID	order_date	ship_date	Product Name	Country	Region	State	City	Segment	Category	Sub-Category	Ship Mode	YearMonth
0	CA-2017-152156	CG-12520	FUR-BO-10001798	2017-11-08	2017-11-11	Bush Somerset Collection Bookcase	United States	South	Kentucky	Henderson	Consumer	Furniture	Bookcases	Second Class	2017-11
1	CA-2017-152156	CG-12520	FUR-CH-10000454	2017-11-08	2017-11-11	Hon Deluxe Fabric Upholstered Stacking Chairs,...	United States	South	Kentucky	Henderson	Consumer	Furniture	Chairs	Second Class	2017-11
2	CA-2017-138688	DV-13045	OFF-LA-10000240	2017-06-12	2017-06-16	Self-Adhesive Address Labels for Typewriters b...	United States	West	California	Los Angeles	Corporate	Office Supplies	Labels	Second Class	2017-06
3	US-2016-108966	SO-20335	FUR-TA-10000577	2016-10-11	2016-10-18	Bretford CR4500 Series Slim Rectangular Table	United States	South	Florida	Fort Lauderdale	Consumer	Furniture	Tables	Standard Class	2016-10
4	US-2016-108966	SO-20335	OFF-ST-10000760	2016-10-11	2016-10-18	Eldon Fold 'N Roll Cart System	United States	South	Florida	Fort Lauderdale	Consumer	Office Supplies	Storage	Standard Class	2016-10

5	CA-2015-115812	BH-11710	FUR-FU-10001487	2015-06-09	2015-06-14	Eldon Expressions Wood and Plastic Desk Access...	United States	West	California	Los Angeles	Consumer	Furniture	Furnishings	Standard Class	2015-06
6	CA-2015-115812	BH-11710	OFF-AR-10002833	2015-06-09	2015-06-14	Newell 322	United States	West	California	Los Angeles	Consumer	Office Supplies	Art	Standard Class	2015-06
7	CA-2015-115812	BH-11710	TEC-PH-10002275	2015-06-09	2015-06-14	Mitel 5320 IP Phone VoIP phone	United States	West	California	Los Angeles	Consumer	Technology	Phones	Standard Class	2015-06
8	CA-2015-115812	BH-11710	OFF-BI-10003910	2015-06-09	2015-06-14	DXL Angle-View Binders with Locking Rings by S...	United States	West	California	Los Angeles	Consumer	Office Supplies	Binders	Standard Class	2015-06
9	CA-2015-115812	BH-11710	OFF-AP-10002892	2015-06-09	2015-06-14	Belkin F5C206VTEL 6 Outlet Surge	United States	West	California	Los Angeles	Consumer	Office Supplies	Appliances	Standard Class	2015-06

```python
# create a column for days of week .
df["day_of_week"] = df.order_date.dt.weekday
```

```python
# Check Missing data
df.isnull().sum()
```

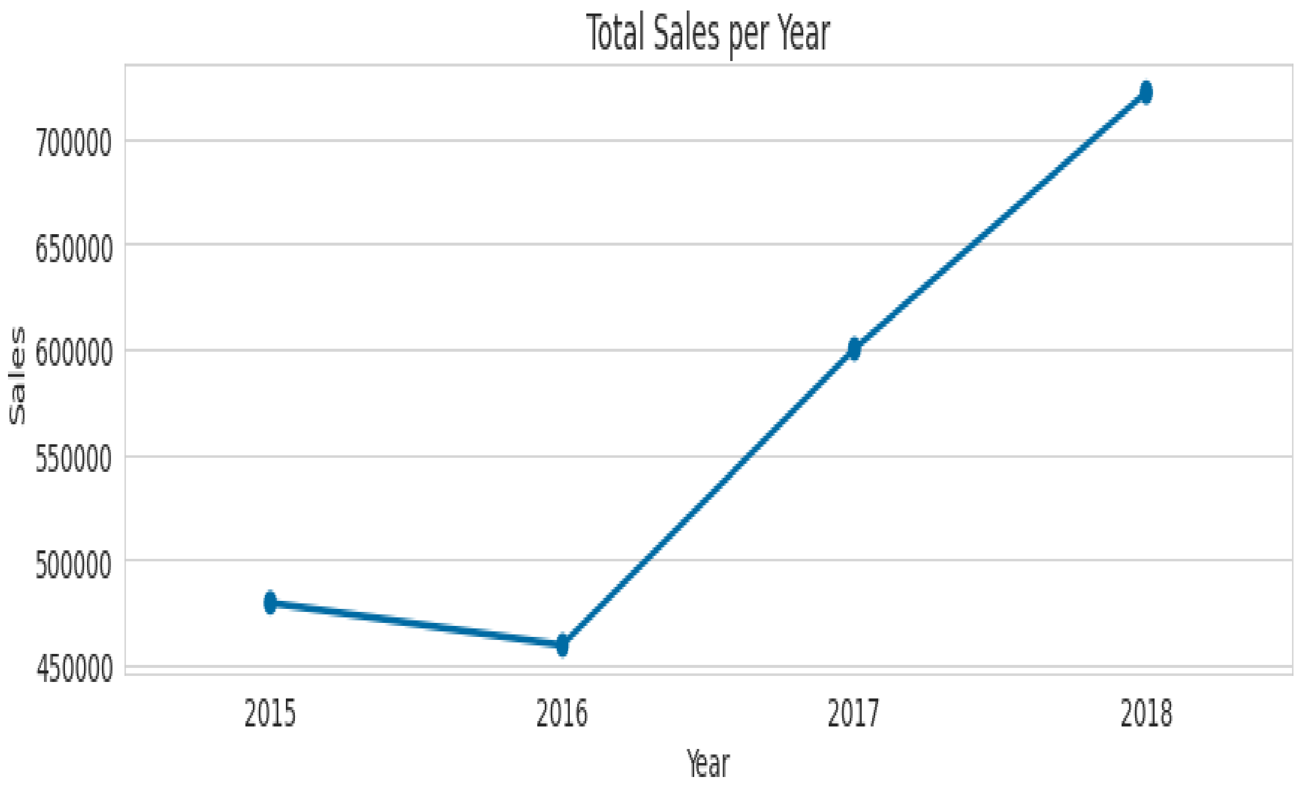
```
Order ID        0
Customer ID     0
Product ID      0
order_date      0
ship_date       0
Product Name    0
Country         0
Region          0
State           0
City            0
Segment         0
Category        0
Sub-Category    0
Ship Mode       0
YearMonth       0
year            0
month           0
shipInDays      0
is_delayed      0
Sales           0
day_of_week     0
dtype: int64
```

```
# check for duplicate rows .
if df.duplicated().sum() >0:
  df = df.drop_duplicates()
```

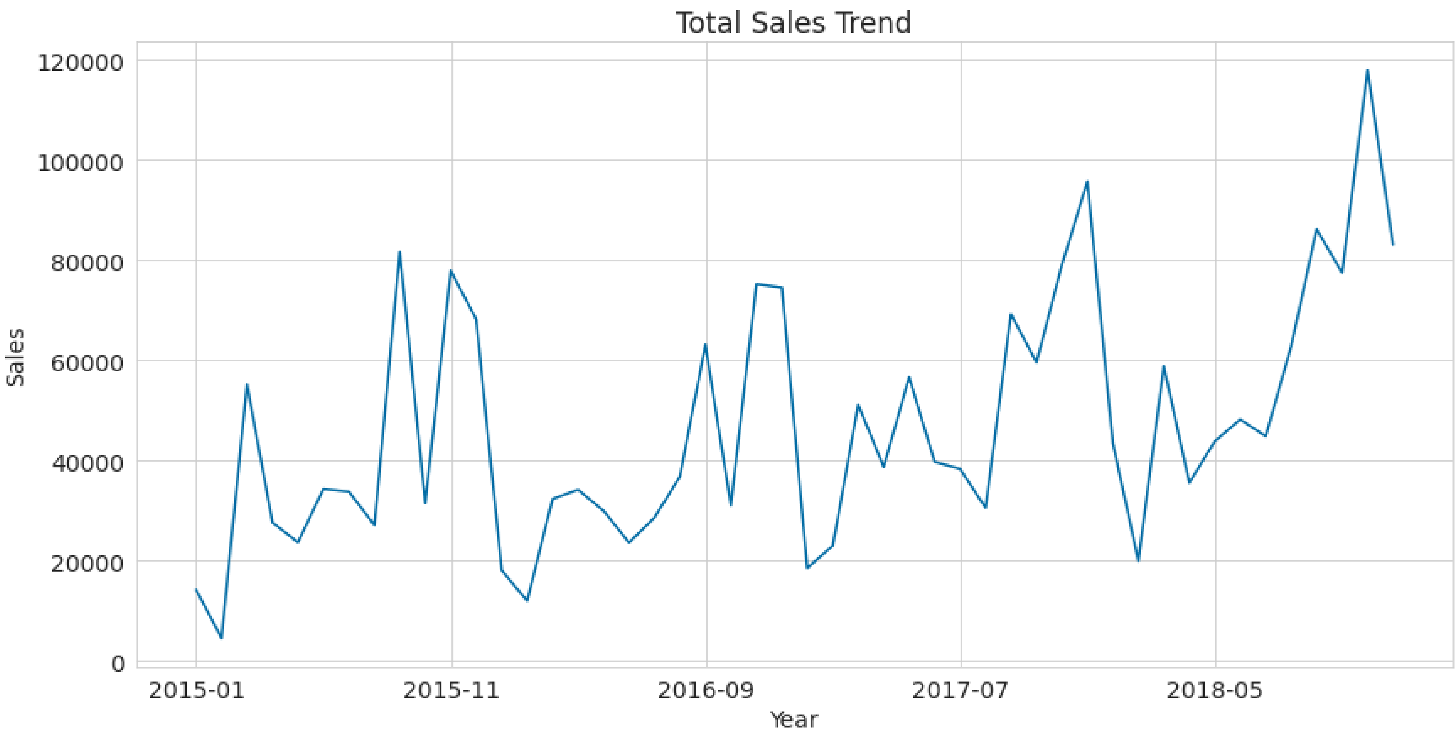
1. Trends and Seasonality

Line Plot

```
# Let's examine Sales over time .
a = pd.DataFrame(df.groupby(by=["year"]).sum())

plt.figure(figsize=(14, 4))
sns.pointplot(x=a.index, y="Sales", data=a)

plt.xlabel("Year")
plt.ylabel("Sales")
plt.title("Total Sales per Year")
plt.show()
```



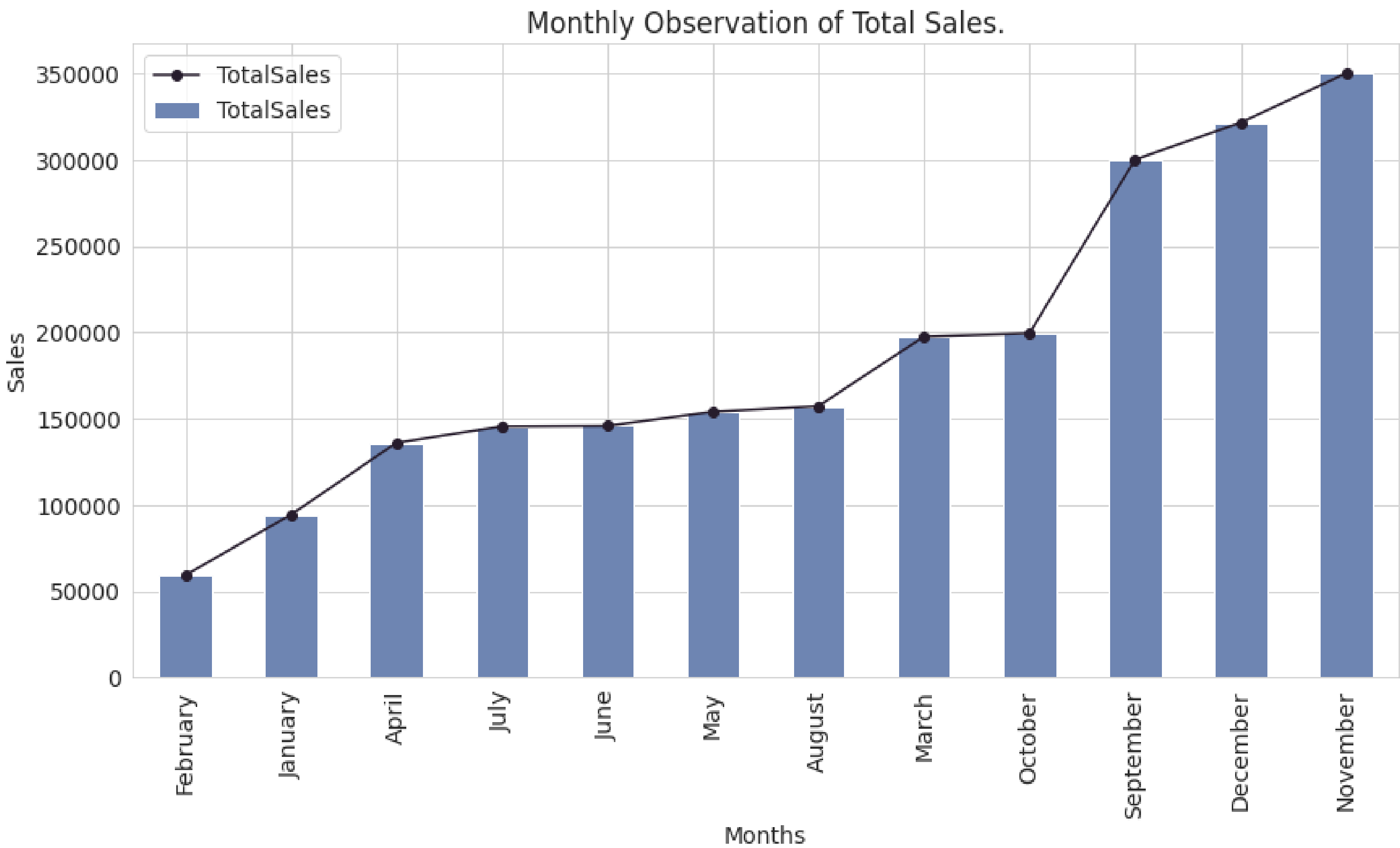
```python
# Year-to-Year observation of TotaloSales .
a = pd.DataFrame(df.groupby(by=["YearMonth"]).sum())["Sales"]

plt.figure(figsize=(14 ,7))
a.plot(kind="line")
plt.xlabel('Year')
plt.ylabel('Sales')
plt.title("Total Sales Trend")
plt.show()
```



- There is increasing trends or growth in Sales over time. There may be seasonality to the sales for each year

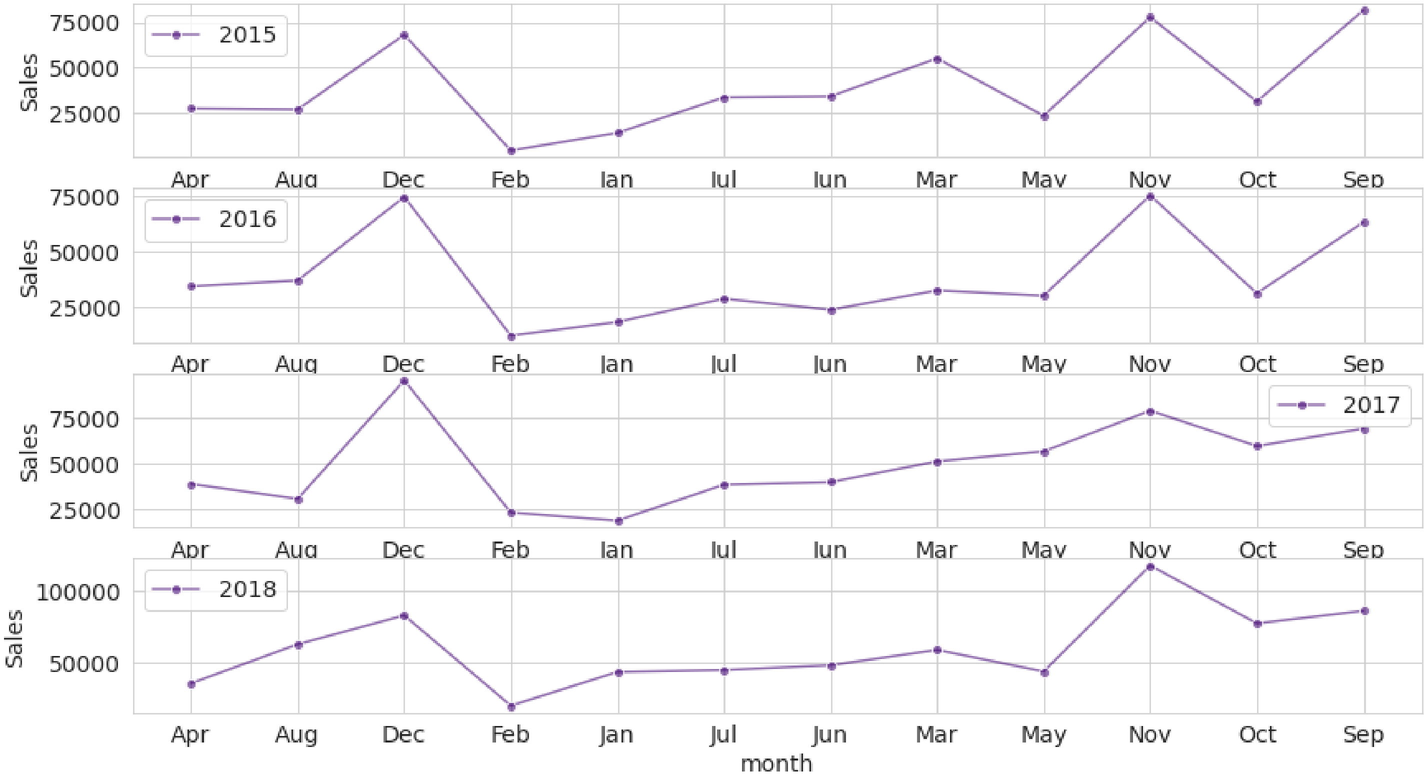
Bar Plot

```
# Monthly observation of Sales Pattern.
monthSales_data = df.groupby(by='month').sum()['Sales']
monthSales_data = monthSales_data.sort_values()

# plot
monthSales_data.plot(kind='line', figsize=(14, 7), color="#261C2C", mar
ker='o', label='TotalSales')
monthSales_data.plot(kind='bar', figsize=(14, 7), color="#6E85B2", labe
l='TotalSales')

plt.xlabel('Months')
plt.ylabel("Sales")
plt.title("Monthly Observation of Total Sales.")

plt.legend()
plt.show()
```



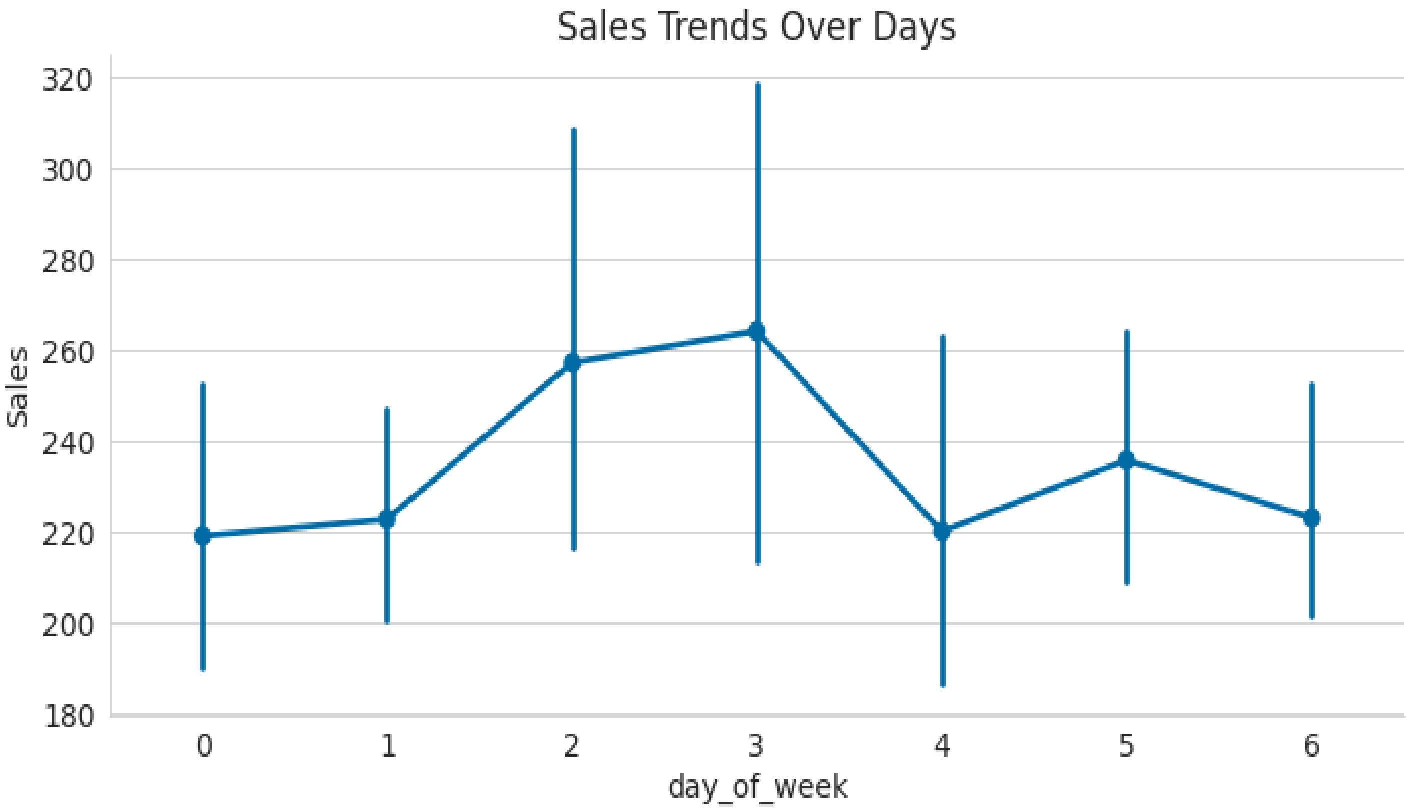
- From above bar plot, we can see that, overall growth in sales obseved in Months of September, December, November. Let's examine, if the same sales pattern observed in each year.

```
# Monthly Year-to-Year observation of Sales Pattern.
monthSales_data = df.groupby(by=['year', 'month']).sum()

a = monthSales_data.reset_index()
a['month'] = a.month.apply(lambda x:x[:3])
monthSales_data = a.groupby(by=['year', 'month']).sum()['Sales']

# plot
fig, ax = plt.subplots(nrows=4, ncols=1, figsize=(14, 8))

yrs = [2015, 2016, 2017, 2018]
for i in range(4):
    yr = yrs[i]
    a = monthSales_data.loc[yr]
    ax[i] = sns.lineplot(x= a.index, y=a.values, data=a, ax=ax[i], labe
l=yr, marker="o", color="#3F007190")
    ax[i].set_ylabel('Sales')

plt.show()
```



➢ We can see that,There is rise in months of December, November, and September. The same pattern observed in each year, however it appears at the different levels.

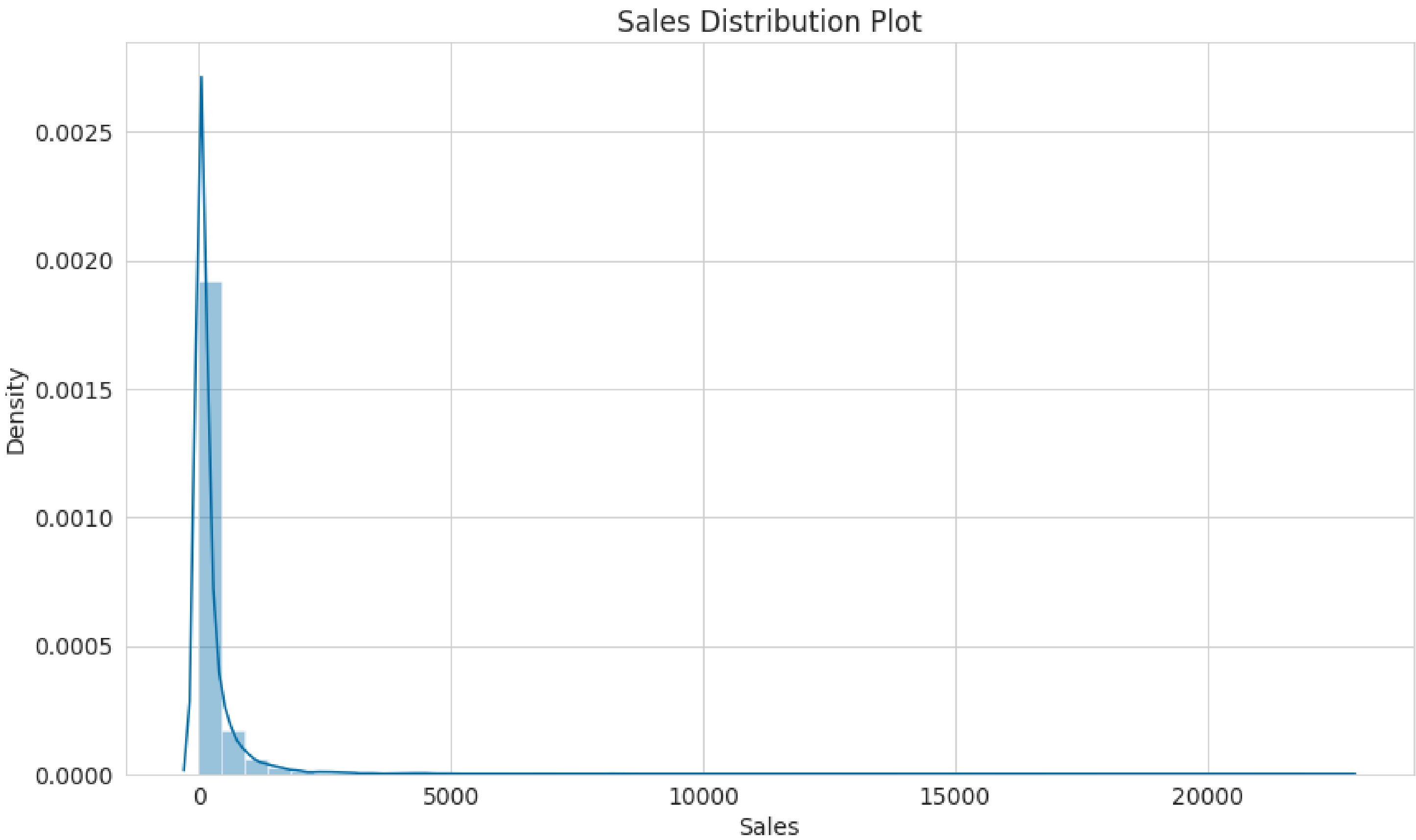
```
# Sales trends over days.
sns.catplot(data=df, x='day_of_week', y='Sales', kind='point', aspect=2
)

plt.title("Sales Trends Over Days")
plt.show()
```



- We can see that, there is maximum sales on Wednesday and Thursday.

Densety Plot

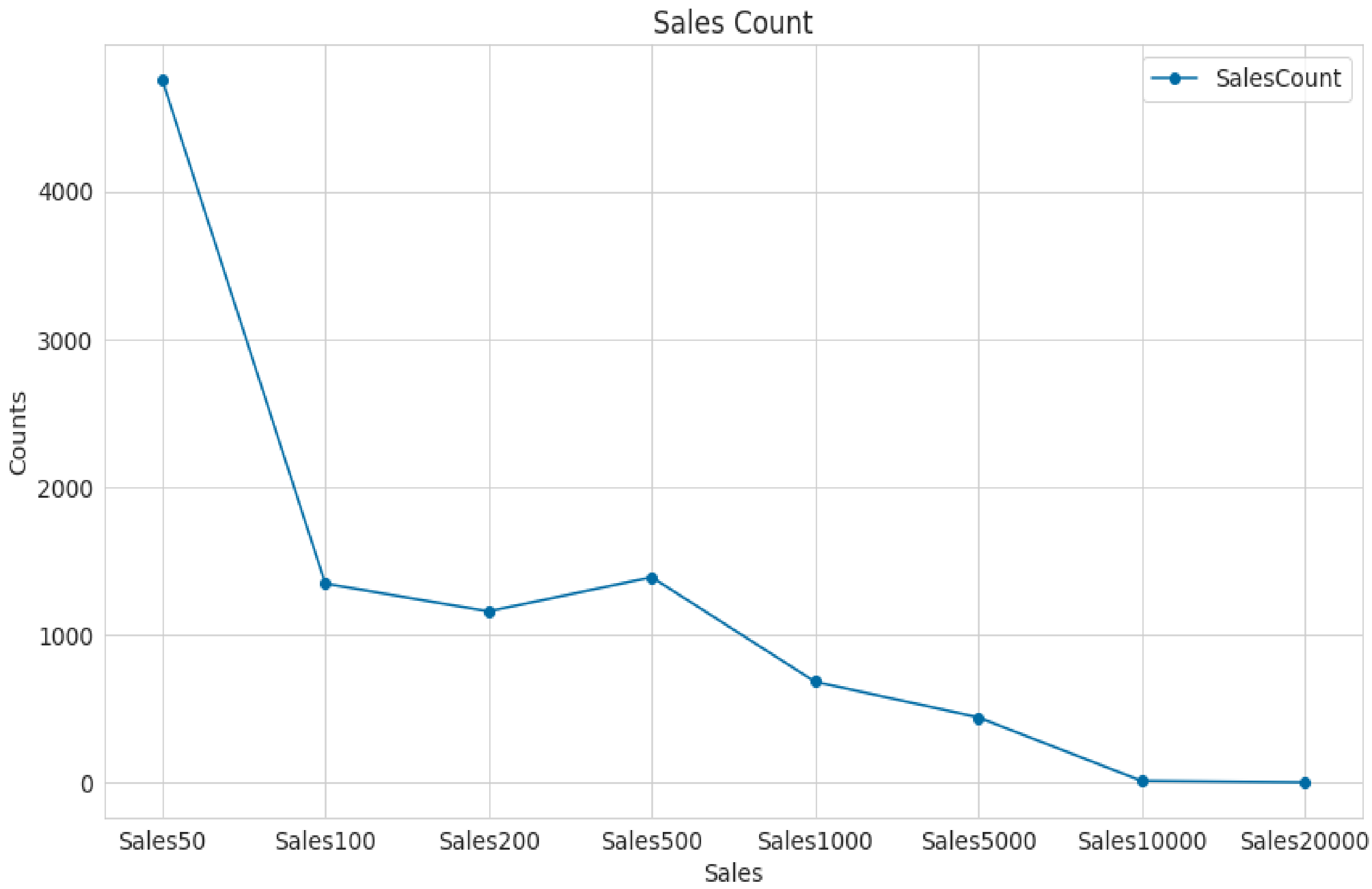
```
# Sales Distribution
plt.figure(figsize=(14, 8))
sns.distplot(data.Sales)

plt.title('Sales Distribution Plot')
plt.show()
```



- Distribution is not Gaussian Distribution. The shape has long right tail, which means that data is Right Skewed. The most of the sales values are less than 50.

```
bins = [0, 50, 100, 200, 500, 1000, 5000, 10000, 20000]
labels=['Sales50', 'Sales100', 'Sales200', 'Sales500', 'Sales1000', 'Sa
les5000', 'Sales10000', 'Sales20000']
a = pd.DataFrame(pd.cut(df['Sales'], bins=bins, labels=labels))
a['SalesCount'] = df['Order ID']

# visualization
```

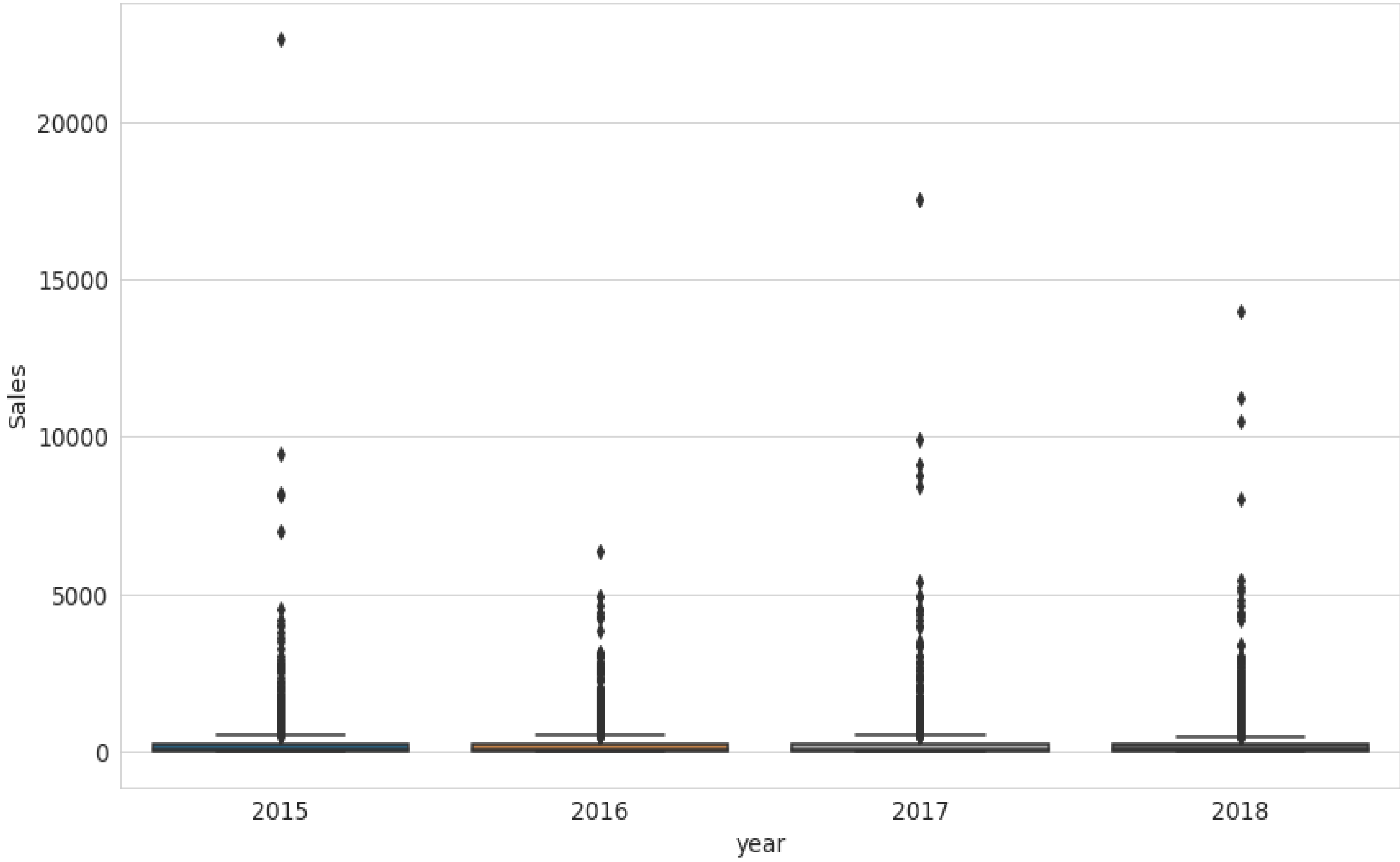
```
a.groupby('Sales').count().plot(kind='line', marker='o', figsize=(14, 8
))

plt.ylabel("Counts")
plt.title("Sales Count")

plt.legend()
plt.show()
```



Box and Whisker plots for Distribution.

➢ Yearly observation of Distribution of Data. This will gives us an idea of spread of observation for each year.

```python
# check for outliers in Sales.
plt.figure(figsize=(14, 8))
sns.boxplot(data=df, x='year', y='Sales', saturation=0.5)
plt.show()
```



➢ We can see that there are outliers in Sales values for each year.

2. Stationarity of Time Series

- To use the time series forecasting models, we need to ensure that the our data is stationay. The time series is stationay when data has constant mean, constant variance, and constant covariance with respect to time.
- There are two ways to check Stationarity of Time Series.

---

1. **Rolling Mean:**

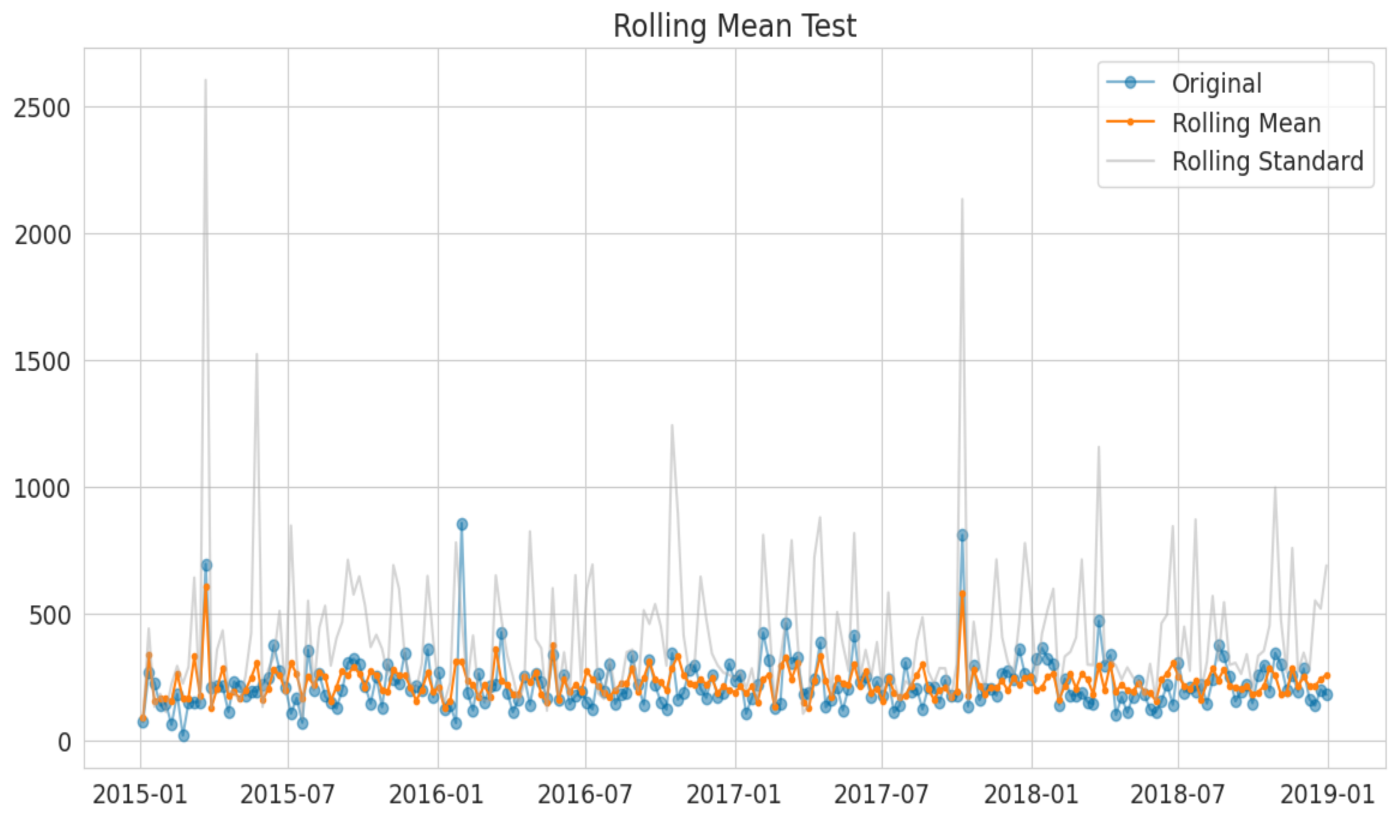
    - A rolling analysis of a time series model is often used to assess the model's stability over time.
    - The window is rolled (slid across the data) on a weekly basis, in which the average is taken on a weekly basis.
    - Rolling Statistics is a visualization test, where we can compare the original data with the rolled data and check if the data is stationary or not.

2. **Augmented Dickey-Fuller test:**

    - The Dickey Fuller test is one of the most popular statistical tests.
    - It can be used to determine the presence of unit root in the series, and hence help us to understand if the series is stationary or not.
    - The null hypothesis of the Augmented Dickey-Fuller is that there is a unit root(i.e data is non-stationary), with the alternative that there is no unit root(i.e data is stationary).
    - if the p-value is less than critical value (i.e 0.05) we reject the null hypothesis which means that data is Stationary.

1.Rolling Mean

```
# prepare data
sales_data = df[['order_date', 'Sales']]
sales_data = sales_data.set_index('order_date')

# calculating rolling statistics.
roll_mean = sales_data.rolling(window=7).mean()
roll_std = sales_data.rolling(window=7).std()

# plotting rolling statistics with orignal data mean.
plt.figure(figsize=(14, 7), dpi=100)
data_mean = plt.plot(sales_data.resample('W').mean(), label='Original',
 marker="o", alpha=0.5)
mean = plt.plot(roll_mean.resample('W').mean(), label="Rolling Mean", m
arker=".")
std = plt.plot(roll_std.resample('W').std(), label="Rolling Standard",
alpha=0.5)

plt.title("Rolling Mean Test")
plt.legend()
plt.show()
```



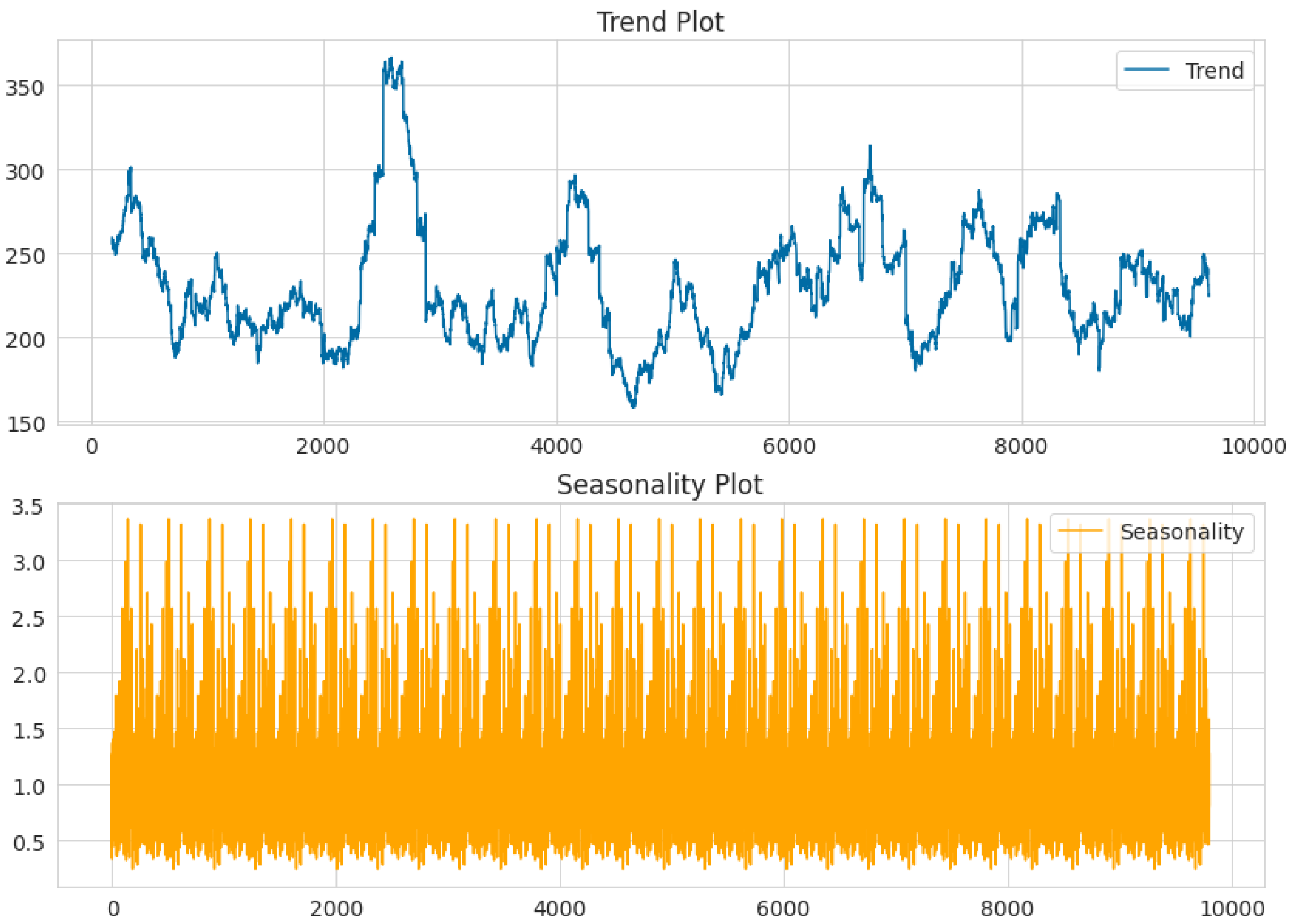
2.Augmented Dickey-Fuller test

```
from statsmodels.tsa.stattools import adfuller

print("Augmented Dickey-fuller test result: ")
result = adfuller(sales_data, autolag="AIC")

print("ADF test statistic: ", result[0])
print("p-value:", result[1])

print("Critical Values:")
for key, val in result[4].items():
    print("\t%s : %f" %(key, val))
```

```
Augmented Dickey-fuller test result:
ADF test statistic:  -98.33059943935697
p-value: 0.0
Critical Values:
        1% : -3.431018
        5% : -2.861835
        10% : -2.566927
```

- Above plot show that, The Mean and Standard deviation does not change over time much which means that the Mean and Deviation is constant.
- The result output of ADF (Augmented Dickey-Fuller) statistical test has value -98.33059943935697 which is smaller than critical value at 1% of -3.431018.
- This suggest that we can reject the null hypothesis with the significance level less than 1%. Rejecting null hypothesis means that, The time series is stationary and does not have time-dependent structure.

```
from statsmodels.tsa.seasonal import seasonal_decompose

decomposition = seasonal_decompose(df.Sales, model = 'multiplicative',
freq=365)

estimated_seasonal = decomposition.seasonal
estimated_trend = decomposition.trend
estimated_residuals = decomposition.resid

fig, axs = plt.subplots(nrows=2, ncols=1, figsize=(14, 10))
axs[0].plot(estimated_trend, label='Trend')
axs[0].set_title("Trend Plot")
axs[0].legend()

axs[1].plot(estimated_seasonal, label='Seasonality', color='orange')
axs[1].set_title("Seasonality Plot")
axs[1].legend()

plt.show()
```



- The above line plot does not show any trends in data. So, There no differencing is required.

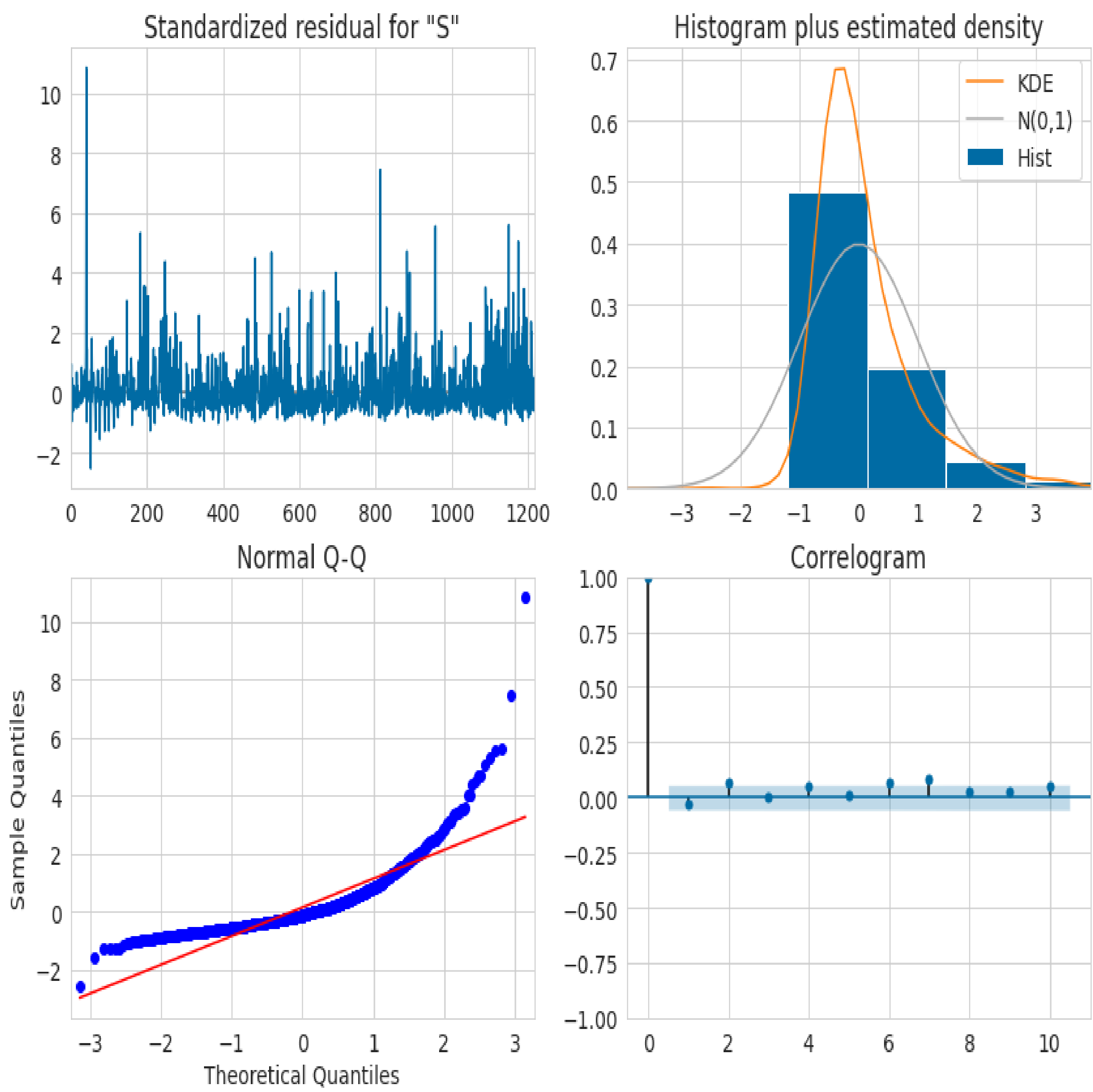
Building a Model

1. AutoRegressive Integrated Moving Average (ARIMA) Model

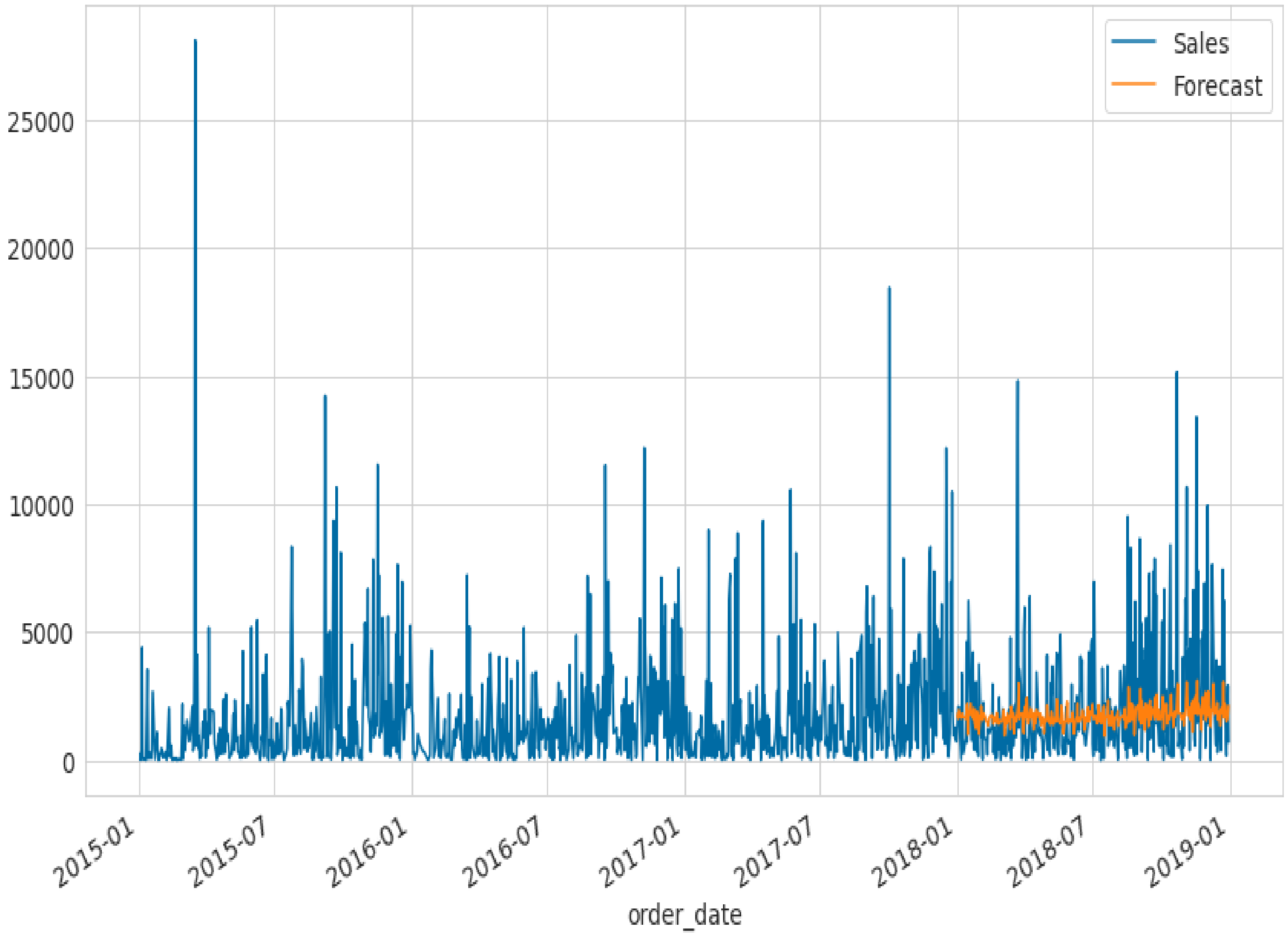
```
import statsmodels.api as sm

import warnings
warnings.filterwarnings('ignore')

sales = pd.DataFrame(df.groupby(by=['order_date']).sum()['Sales'])

# Fitting ARIMA model
model = sm.tsa.statespace.SARIMAX(sales,order=(1, 0, 0), seasonal_order
=(1, 1, 1, 12))
result = model.fit()
print("SARIMAX Summary")
print(result.summary().tables[1])
```

```
SARIMAX Summary
======================================================================
=======
                 coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
-------
ar.L1          0.1067      0.031      3.466      0.001       0.046
0.167
ar.S.L12       0.0394      0.023      1.689      0.091      -0.006
0.085
ma.S.L12      -0.9993      0.014    -69.130      0.000      -1.028
-0.971
sigma2      5.199e+06   2.82e-09   1.84e+15      0.000     5.2e+06
5.2e+06
      ================================================================
```

```
# Visualization of the performance of our model
result.plot_diagnostics(figsize=(14, 10))
plt.show()
```



```
sales['Forecast']= pd.DataFrame(result.predict(start='2018-01-
01', end='2018-12-30', dynamic=False))

# visualization for the same
sales.plot(figsize=(14, 8))
plt.show()
```



```
actual = sales.loc['2018-01-01':'2018-12-30']['Sales']
preds = sales.loc['2018-01-01':'2018-12-30']['Forecast']
rmse_sarima = sqrt(mean_squared_error(preds, actual))
print("Root Mean Squared Error for SARIMAX:", rmse_sarima)
```

```
Root Mean Squared Error for SARIMAX: 2404.877289903688
```

2. XGBoost

```
from xgboost import XGBRegressor

xgb_sales = pd.DataFrame(df.groupby(by=['order_date']).sum())

x = xgb_sales.drop('Sales', axis=1)
y = xgb_sales['Sales']

x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.
3, random_state=42)

model = XGBRegressor(learning_rate=0.03, max_depth=1,)
model.fit(x_train, y_train)

preds = model.predict(x_test)
rmse_xgb = sqrt(mean_squared_error(y_test, preds))

print("Root Mean Squared Error for XGBoost:", rmse_xgb)
```

```
[08:15:47] WARNING: /workspace/src/objective/regression_obj.cu:152:
reg:linear is now deprecated in favor of reg:squarederror.
      Root Mean Squared Error for XGBoost: 1714.6441625235727
```

Model Evaluation

```
result = pd.DataFrame([[rmse_sarima], [rmse_xgb]], columns=['RMSE'], in
dex=['SARIMAX','XGBRegressor'])
result
```

	RMSE
SARIMAX	2404.877290
XGBRegressor	1714.644163

- The Root mean squared error of XGBRegressor model is less than SARIMAX. We can use XGBRegressor for forecasting Sales.

Conclusion

Working with 4 years of sales data in the United States. Data dictionary of the dataset is appended below. We can see that Data columns is of Object types, Let's change it into Data datatype. There is increasing trends or growth in Sales over time. There may be seasonality to the sales for each year. From above bar plot, we can see that, overall growth in sales observed in month of September, December, November. Let's examine, if the same sales pattern observed in each year. We can see that, There is rise in months of December, November, and September. The same pattern observed in each year, however it appears at the different levels.
We can see that, there is maximum sales on Wednesday and Thursday. Distribution is not Gaussian Distribution. The shape has long right tail, which means that data is Right Skewed. The most of the sales values are less then 50. Yearly observation of Distribution of Data. This will gives us an idea of spread od observation for each year. We can see that there are outliers in Sales values for each year. The Mean and Standard deviation does not change over time much which means that the Mean and Deviation is constant. The above line plot does not show any trends in data. So, There no differencing is required**. Root Mean Square Error for SARIMAX** : **2404.877** and **Root Mean Square Error for XGBoost** : **1714.644**. The Root mean squared error of XGBRegressor model is less than SARIMAX. We can XFBRegressor for forecasting Sales.