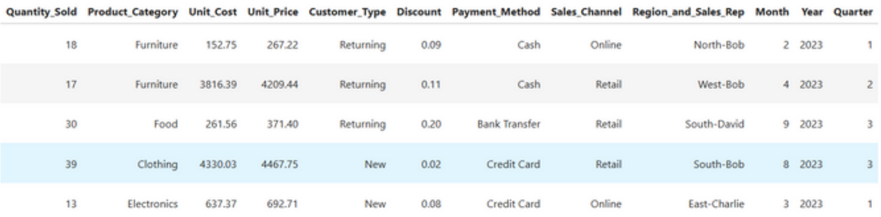
Sales_Analytics_In_Advance

1.Data Preparation

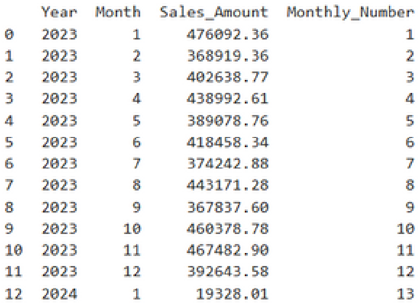
```
import pandas as pd
import datetime as dt
data=pd.read_csv("sales_data.csv")
df=pd.DataFrame(data)
df['Sale_Date']=pd.to_datetime(df['Sale_Date'])
df['Month']=df['Sale_Date'].dt.month
df['Year']=df['Sale_Date'].dt.year
df['Quarter']=df['Sale_Date'].dt.quarter
df.head()
```



Quantity_Sold	Product_Category	Unit_Cost	Unit_Price	Customer_Type	Discount	Payment_Method	Sales_Channel	Region_and_Sales_Rep	Month	Year	Quarter
18	Furniture	152.75	267.22	Returning	0.09	Cash	Online	North-Bob	2	2023	1
17	Furniture	3816.39	4209.44	Returning	0.11	Cash	Retail	West-Bob	4	2023	2
30	Food	261.56	371.40	Returning	0.20	Bank Transfer	Retail	South-David	9	2023	3
39	Clothing	4330.03	4467.75	New	0.02	Credit Card	Retail	South-Bob	8	2023	3
13	Electronics	637.37	692.71	New	0.08	Credit Card	Online	East-Charlie	3	2023	1

2.Monthly_Sales

```
monthly_sales=df.groupby(['Year','Month'])['Sales_Amount'].sum().reset_index()
monthly_sales['Monthly_Number']=range(1,len(monthly_sales)+1)
print(monthly_sales)
```

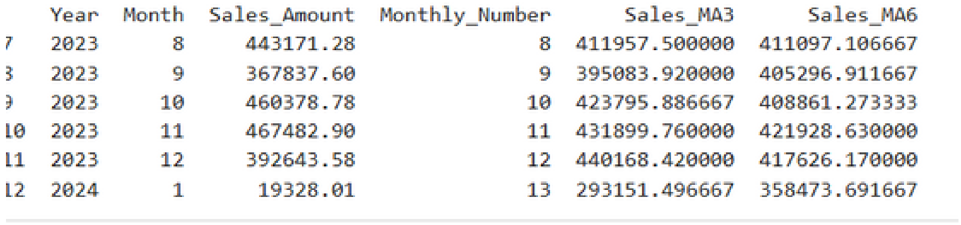


```
    Year  Month  Sales_Amount  Monthly_Number
0   2023      1     476092.36               1
1   2023      2     368919.36               2
2   2023      3     402638.77               3
3   2023      4     438992.61               4
4   2023      5     389078.76               5
5   2023      6     418458.34               6
6   2023      7     374242.88               7
7   2023      8     443171.28               8
8   2023      9     367837.60               9
9   2023     10     460378.78              10
10  2023     11     467482.90              11
11  2023     12     392643.58              12
12  2024      1      19328.01              13
```

3.Now rolling averages

```
monthly_sales['Sales_MA3'] = monthly_sales['Sales_Amount'].rolling(window=3,
min_periods=1).mean()
monthly_sales['Sales_MA6'] = monthly_sales['Sales_Amount'].rolling(window=6,
min_periods=1).mean()

print(monthly_sales.tail(6))
```



```
    Year  Month  Sales_Amount  Monthly_Number      Sales_MA3      Sales_MA6
7   2023      8     443171.28               8  411957.500000  411097.106667
3   2023      9     367837.60               9  395083.920000  405296.911667
)   2023     10     460378.78              10  423795.886667  408861.273333
l0  2023     11     467482.90              11  431899.760000  421928.630000
l1  2023     12     392643.58              12  440168.420000  417626.170000
l2  2024      1      19328.01              13  293151.496667  358473.691667
```

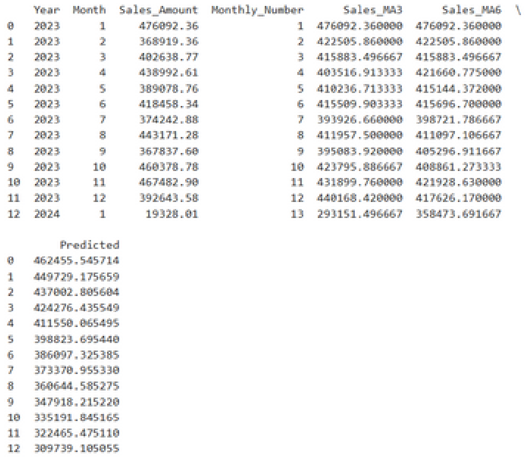
4.Predictive Modeling (Linear Regression)

```
from sklearn.linear_model import LinearRegression
import numpy as np

X = monthly_sales[['Monthly_Number']]
y = monthly_sales['Sales_Amount']

model = LinearRegression()
model.fit(X, y)
# Predict for next 3 months
future = pd.DataFrame({
   "Monthly_Number": [len(monthly_sales)+1,
                 len(monthly_sales)+2,
                 len(monthly_sales)+3]
})

predictions = model.predict(future)
monthly_sales['Predicted'] = model.predict(X)
print(monthly_sales)
```

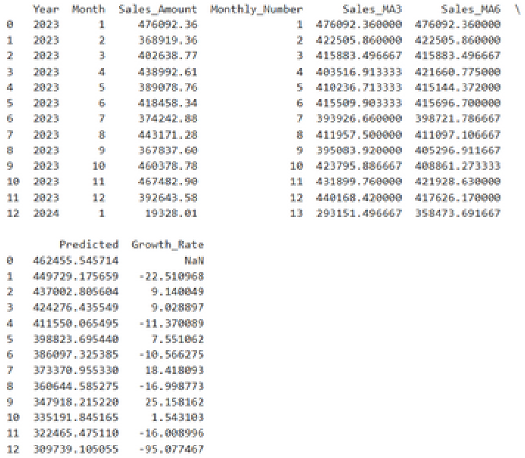


```
    Year  Month  Sales_Amount  Monthly_Number      Sales_MA3      Sales_MA6  \
0   2023      1     476092.36               1  476092.360000  476092.360000
1   2023      2     368919.36               2  422505.860000  422505.860000
2   2023      3     402638.77               3  415883.496667  415883.496667
3   2023      4     438992.61               4  403516.913333  421660.775000
4   2023      5     389078.76               5  410236.713333  415144.372000
5   2023      6     418458.34               6  415509.903333  415696.700000
6   2023      7     374242.88               7  393926.660000  398721.786667
7   2023      8     443171.28               8  411957.500000  411097.106667
8   2023      9     367837.60               9  395083.920000  405296.911667
9   2023     10     460378.78              10  423795.886667  408861.273333
10  2023     11     467482.90              11  431899.760000  421928.630000
11  2023     12     392643.58              12  440168.420000  417626.170000
12  2024      1      19328.01              13  293151.496667  358473.691667

        Predicted
0   462455.545714
1   449729.175659
2   437002.805604
3   424276.435549
4   411550.065495
5   398823.695440
6   386097.325385
7   373370.955330
8   360644.585275
9   347918.215220
10  335191.845165
11  322465.475110
12  309739.105055
```

5.Growth_rate

```
monthly_sales['Growth_Rate'] = monthly_sales['Sales_Amount'].pct_change() * 100
print(monthly_sales)
```



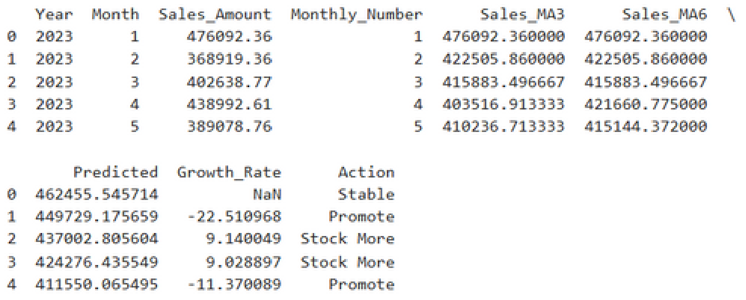
```
    Year  Month  Sales_Amount  Monthly_Number      Sales_MA3      Sales_MA6  \
0   2023      1     476092.36               1  476092.360000  476092.360000
1   2023      2     368919.36               2  422505.860000  422505.860000
2   2023      3     402638.77               3  415883.496667  415883.496667
3   2023      4     438992.61               4  403516.913333  421660.775000
4   2023      5     389078.76               5  410236.713333  415144.372000
5   2023      6     418458.34               6  415509.903333  415696.700000
6   2023      7     374242.88               7  393926.660000  398721.786667
7   2023      8     443171.28               8  411957.500000  411097.106667
8   2023      9     367837.60               9  395083.920000  405296.911667
9   2023     10     460378.78              10  423795.886667  408861.273333
10  2023     11     467482.90              11  431899.760000  421928.630000
11  2023     12     392643.58              12  440168.420000  417626.170000
12  2024      1      19328.01              13  293151.496667  358473.691667

        Predicted  Growth_Rate
0   462455.545714          NaN
1   449729.175659   -22.510968
2   437002.805604     9.140049
3   424276.435549     9.028897
4   411550.065495   -11.370089
5   398823.695440     7.551062
6   386097.325385   -10.566275
7   373370.955330    18.418093
8   360644.585275   -16.998773
9   347918.215220    25.158162
10  335191.845165     1.543103
11  322465.475110   -16.008996
12  309739.105055   -95.077467
```

6.Recommendation for growth_rate

```
def recommendation(growth):
 if growth > 5:
 return "Stock More"
 elif growth < -5:
 return "Promote"
 else:
 return "Stable"

monthly_sales['Action'] = monthly_sales['Growth_Rate'].apply(recommendation)
print(monthly_sales.head())
```



```
   Year  Month  Sales_Amount  Monthly_Number       Sales_MA3       Sales_MA6  \
0  2023      1     476092.36               1  476092.360000  476092.360000
1  2023      2     368919.36               2  422505.860000  422505.860000
2  2023      3     402638.77               3  415883.496667  415883.496667
3  2023      4     438992.61               4  403516.913333  421660.775000
4  2023      5     389078.76               5  410236.713333  415144.372000

       Predicted  Growth_Rate      Action
0  462455.545714          NaN      Stable
1  449729.175659   -22.510968     Promote
2  437002.805604     9.140049  Stock More
3  424276.435549     9.028897  Stock More
4  411550.065495   -11.370089     Promote
```

7.Visuale Chart

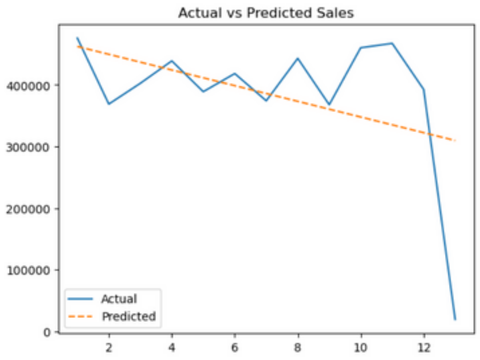
```
import matplotlib.pyplot as plt
import seaborn as sns
```

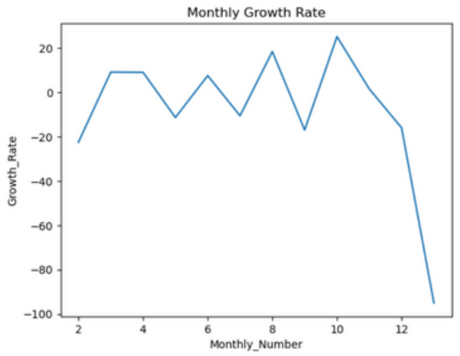
Actual vs Predicted

```
plt.plot(monthly_sales['Monthly_Number'], monthly_sales['Sales_Amount'], label="Actual")
plt.plot(monthly_sales['Monthly_Number'], monthly_sales['Predicted'], label="Predicted", linestyle="--")
plt.legend()
plt.title("Actual vs Predicted Sales")
plt.show()
```

Growth Rate Trend

```
sns.lineplot(x='Monthly_Number', y='Growth_Rate', data=monthly_sales)
plt.title("Monthly Growth Rate")
plt.show()
```





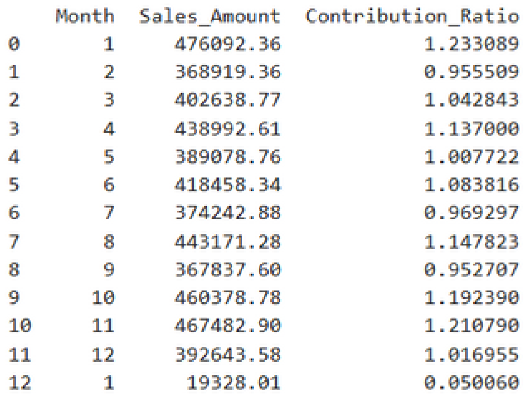
8.Average monthly sales in 2023 and Contribution Ratio

```
avg_sales = monthly_sales['Sales_Amount'].mean()
```

Contribution ratio = actual / average

```
monthly_sales['Contribution_Ratio'] = monthly_sales['Sales_Amount'] / avg_sales

print(monthly_sales[['Month','Sales_Amount','Contribution_Ratio']])
```



```
    Month  Sales_Amount  Contribution_Ratio
0       1     476092.36            1.233089
1       2     368919.36            0.955509
2       3     402638.77            1.042843
3       4     438992.61            1.137000
4       5     389078.76            1.007722
5       6     418458.34            1.083816
6       7     374242.88            0.969297
7       8     443171.28            1.147823
8       9     367837.60            0.952707
9      10     460378.78            1.192390
10     11     467482.90            1.210790
11     12     392643.58            1.016955
12      1      19328.01            0.050060
```