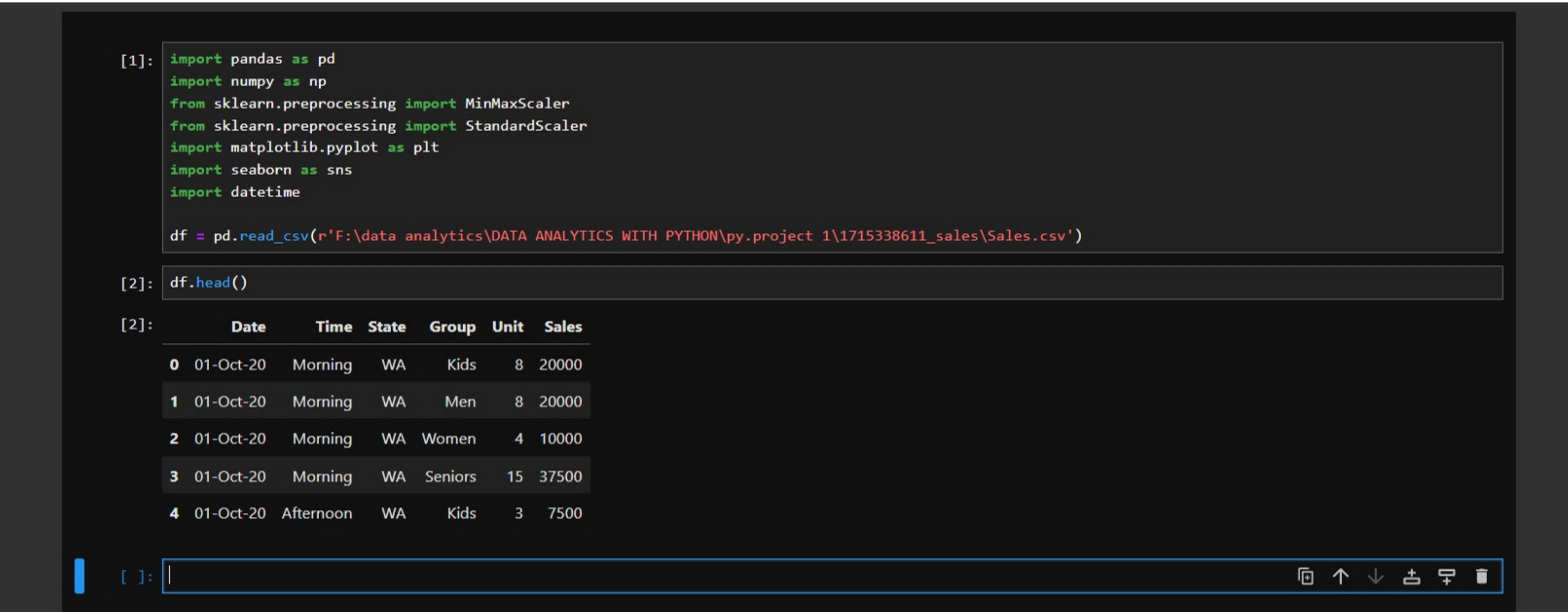
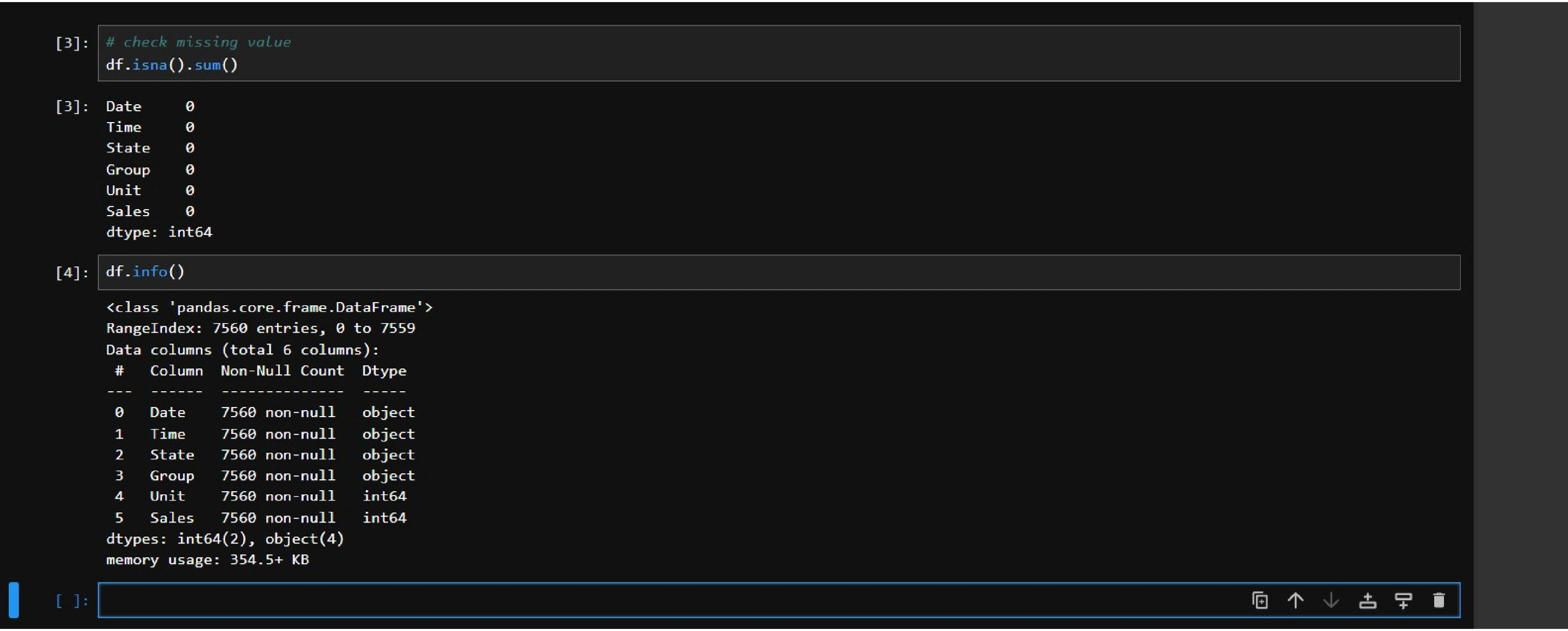


```
[1]: import pandas as pd
     import numpy as np
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.preprocessing import StandardScaler
     import matplotlib.pyplot as plt
     import seaborn as sns
     import datetime

     df = pd.read_csv(r'F:\data analytics\DATA ANALYTICS WITH PYTHON\py.project 1\1715338611_sales\Sales.csv')
```

```
[2]: df.head()
```

[2]:

	Date	Time	State	Group	Unit	Sales
0	01-Oct-20	Morning	WA	Kids	8	20000
1	01-Oct-20	Morning	WA	Men	8	20000
2	01-Oct-20	Morning	WA	Women	4	10000
3	01-Oct-20	Morning	WA	Seniors	15	37500
4	01-Oct-20	Afternoon	WA	Kids	3	7500



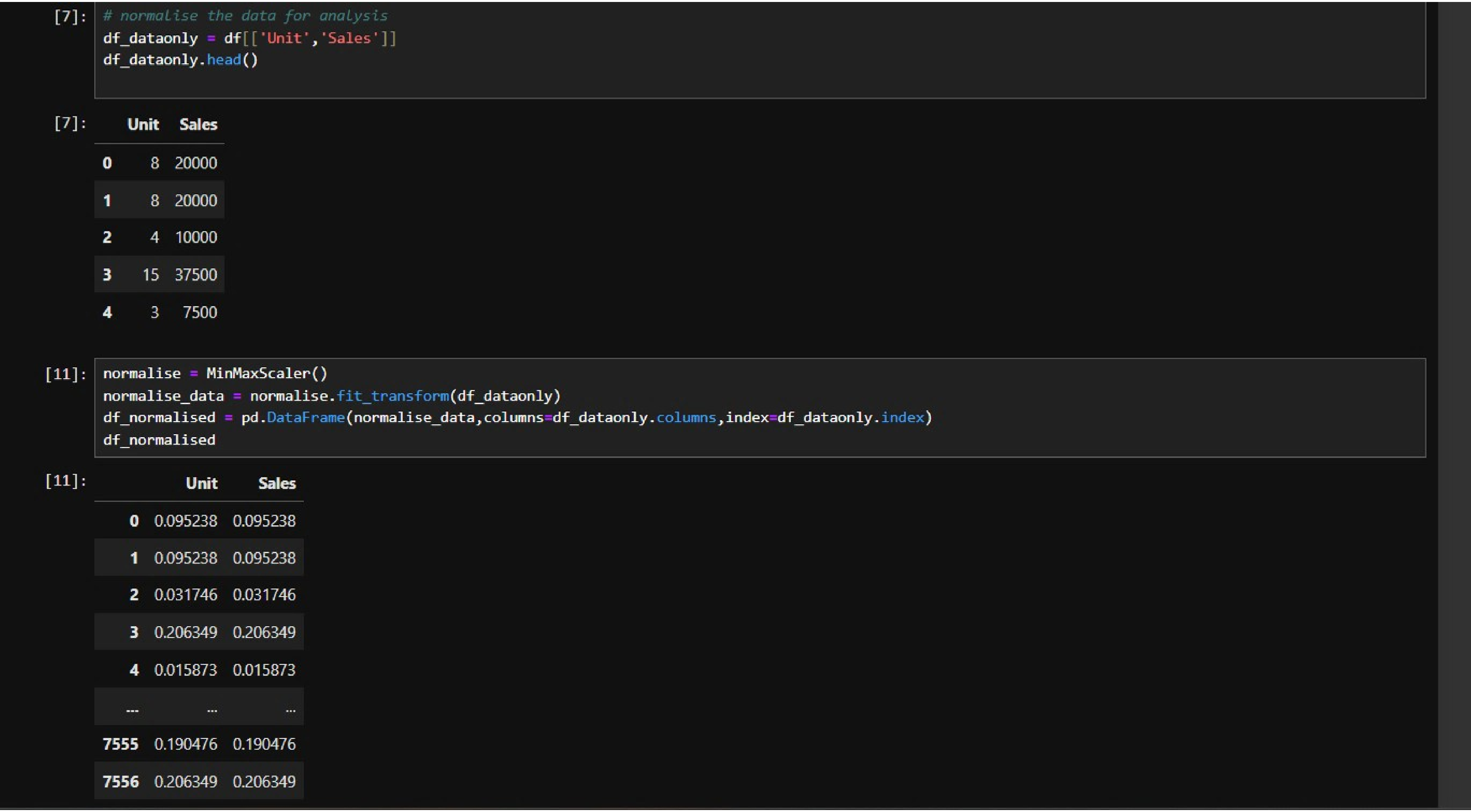
```
[3]: # check missing value
     df.isna().sum()
```

```
[3]: Date     0
     Time     0
     State    0
     Group    0
     Unit     0
     Sales    0
     dtype: int64
```

```
[4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7560 entries, 0 to 7559
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Date    7560 non-null   object
 1   Time    7560 non-null   object
 2   State   7560 non-null   object
 3   Group   7560 non-null   object
 4   Unit    7560 non-null   int64 
 5   Sales   7560 non-null   int64 
dtypes: int64(2), object(4)
memory usage: 354.5+ KB
```

```
[ ]:
```

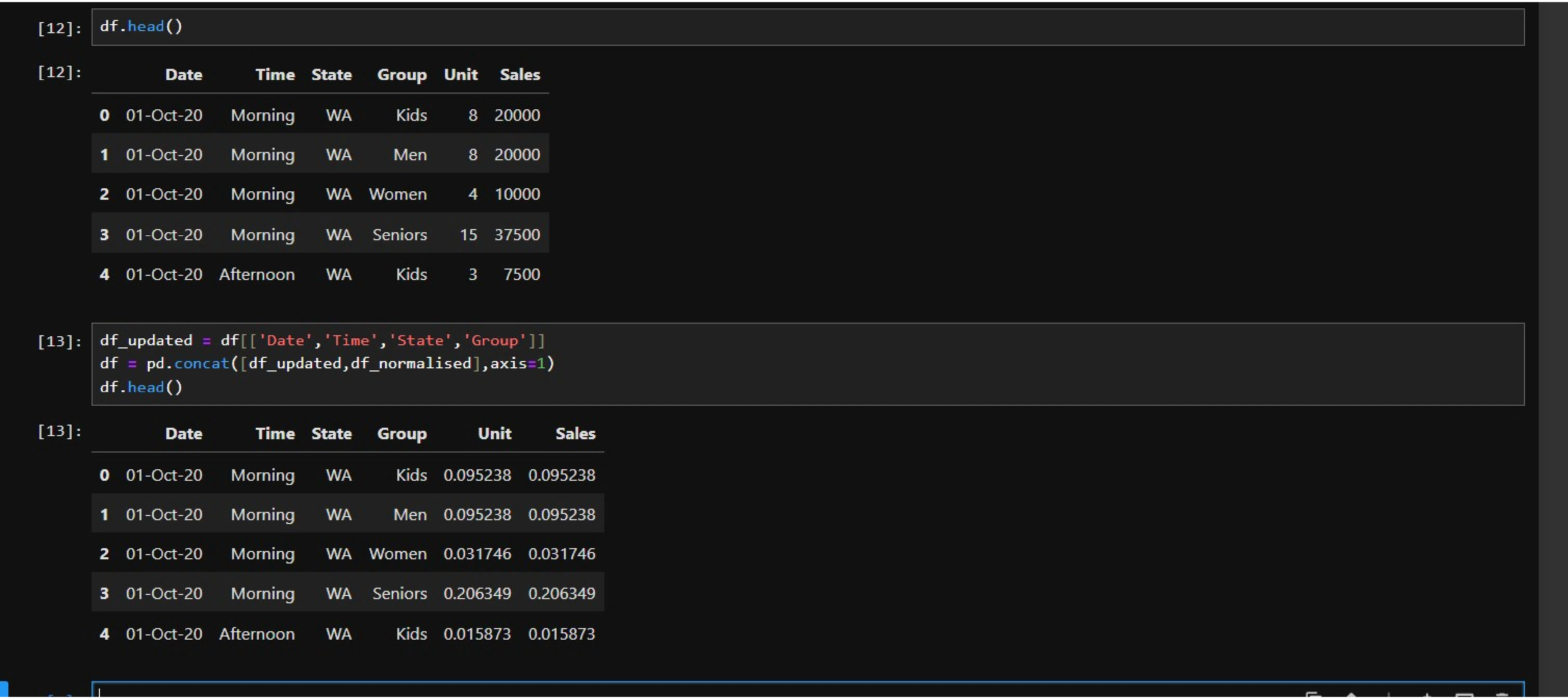


```
[7]: # normalise the data for analysis
     df_dataonly = df[['Unit','Sales']]
     df_dataonly.head()
```

```
[7]:    Unit  Sales
     0     8  20000
     1     8  20000
     2     4  10000
     3    15  37500
     4     3   7500
```

```
[11]: normalise = MinMaxScaler()
      normalise_data = normalise.fit_transform(df_dataonly)
      df_normalised = pd.DataFrame(normalise_data,columns=df_dataonly.columns,index=df_dataonly.index)
      df_normalised
```

```
[11]:          Unit     Sales
         0  0.095238  0.095238
         1  0.095238  0.095238
         2  0.031746  0.031746
         3  0.206349  0.206349
         4  0.015873  0.015873
       ...       ...       ...
      7555  0.190476  0.190476
      7556  0.206349  0.206349
```

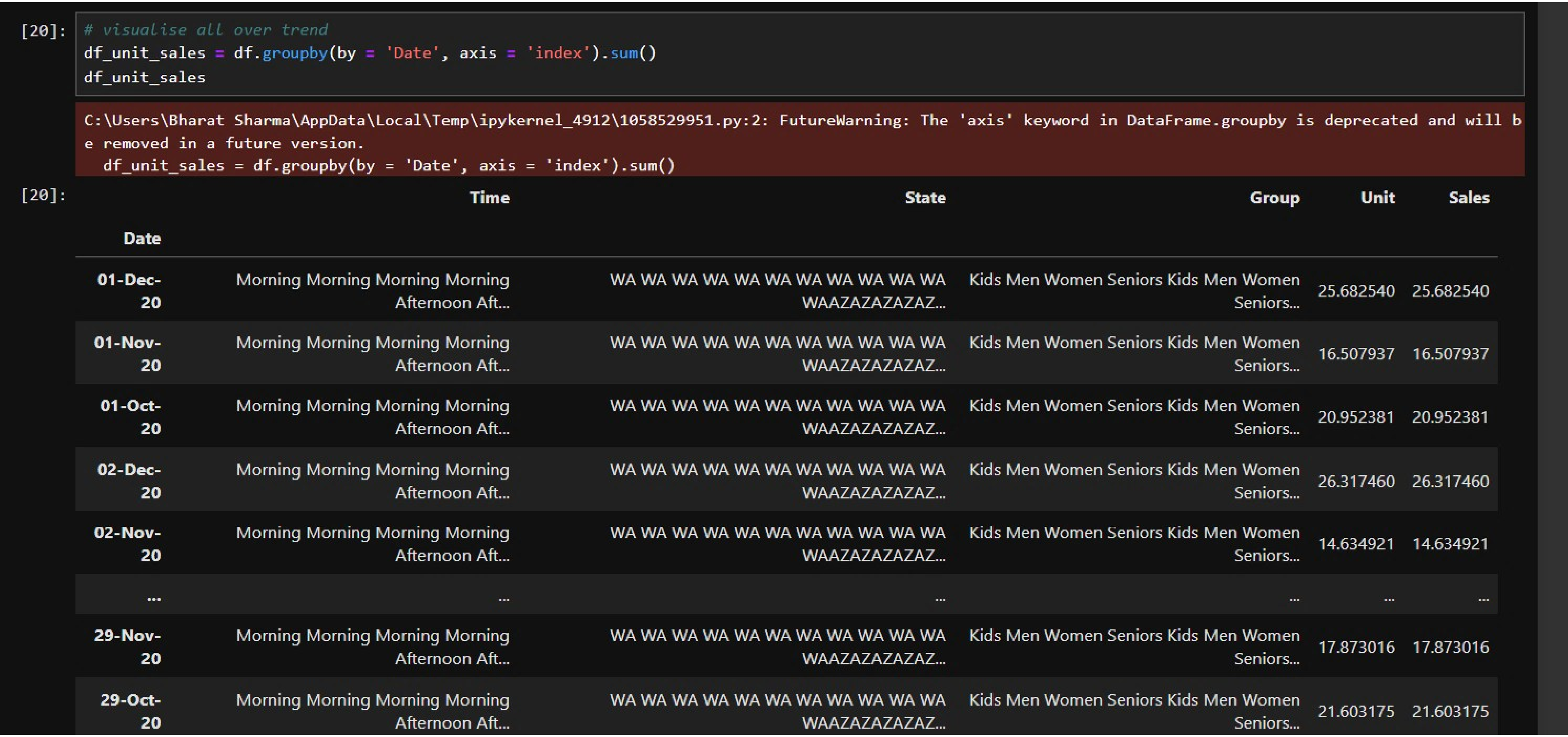


```
[12]: df.head()
```

```
[12]:
        Date       Time  State    Group  Unit  Sales
   0  01-Oct-20    Morning    WA     Kids     8  20000
   1  01-Oct-20    Morning    WA      Men     8  20000
   2  01-Oct-20    Morning    WA    Women     4  10000
   3  01-Oct-20    Morning    WA  Seniors    15  37500
   4  01-Oct-20  Afternoon    WA     Kids     3   7500
```

```
[13]: df_updated = df[['Date','Time','State','Group']]
      df = pd.concat([df_updated,df_normalised],axis=1)
      df.head()
```

```
[13]:
        Date       Time  State    Group      Unit     Sales
   0  01-Oct-20    Morning    WA     Kids  0.095238  0.095238
   1  01-Oct-20    Morning    WA      Men  0.095238  0.095238
   2  01-Oct-20    Morning    WA    Women  0.031746  0.031746
   3  01-Oct-20    Morning    WA  Seniors  0.206349  0.206349
   4  01-Oct-20  Afternoon    WA     Kids  0.015873  0.015873
```

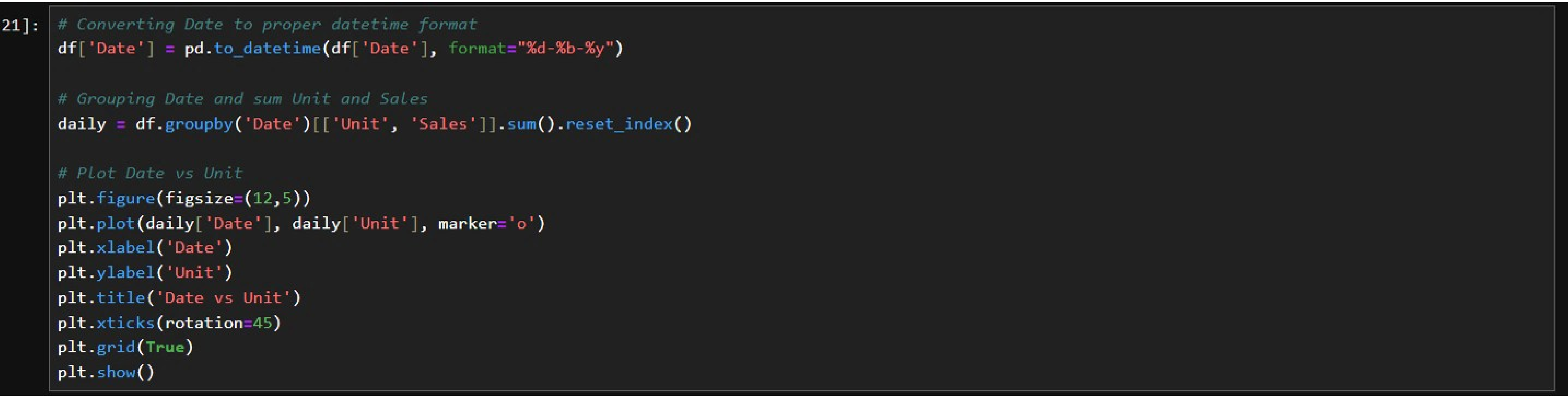


```
[20]: # visualise all over trend
      df_unit_sales = df.groupby(by = 'Date', axis = 'index').sum()
      df_unit_sales
```

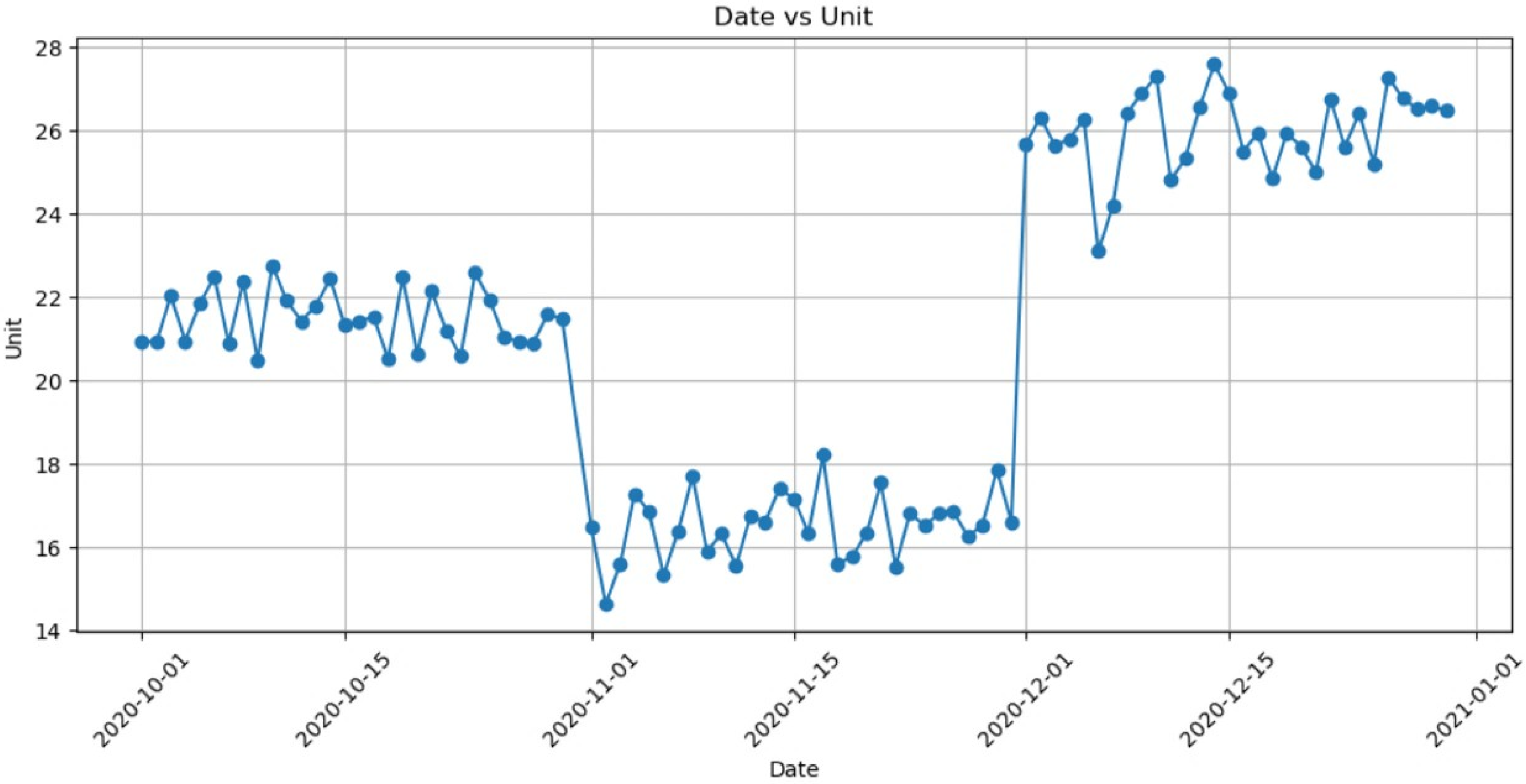
```
C:\Users\Bharat Sharma\AppData\Local\Temp\ipykernel_4912\1058529951.py:2: FutureWarning: The 'axis' keyword in DataFrame.groupby is deprecated and will b
e removed in a future version.
  df_unit_sales = df.groupby(by = 'Date', axis = 'index').sum()
```

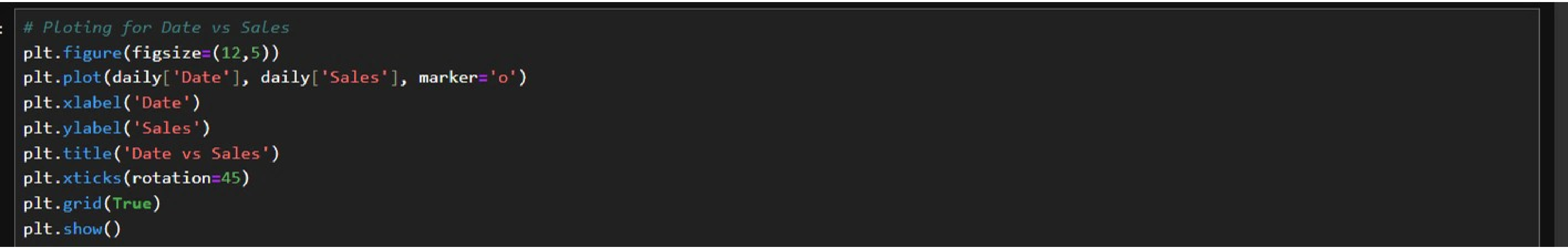
[20]:

	Time	State	Group	Unit	Sales
Date					
01-Dec-20	Morning Morning Morning Morning Afternoon Aft...	WA WA WA WA WA WA WA WA WA WA WA WAAZAZAZAZAZ...	Kids Men Women Seniors Kids Men Women Seniors...	25.682540	25.682540
01-Nov-20	Morning Morning Morning Morning Afternoon Aft...	WA WA WA WA WA WA WA WA WA WA WA WAAZAZAZAZAZ...	Kids Men Women Seniors Kids Men Women Seniors...	16.507937	16.507937
01-Oct-20	Morning Morning Morning Morning Afternoon Aft...	WA WA WA WA WA WA WA WA WA WA WA WAAZAZAZAZAZ...	Kids Men Women Seniors Kids Men Women Seniors...	20.952381	20.952381
02-Dec-20	Morning Morning Morning Morning Afternoon Aft...	WA WA WA WA WA WA WA WA WA WA WA WAAZAZAZAZAZ...	Kids Men Women Seniors Kids Men Women Seniors...	26.317460	26.317460
02-Nov-20	Morning Morning Morning Morning Afternoon Aft...	WA WA WA WA WA WA WA WA WA WA WA WAAZAZAZAZAZ...	Kids Men Women Seniors Kids Men Women Seniors...	14.634921	14.634921
...	...	...	...	...	...
29-Nov-20	Morning Morning Morning Morning Afternoon Aft...	WA WA WA WA WA WA WA WA WA WA WA WAAZAZAZAZAZ...	Kids Men Women Seniors Kids Men Women Seniors...	17.873016	17.873016
29-Oct-20	Morning Morning Morning Morning Afternoon Aft...	WA WA WA WA WA WA WA WA WA WA WA WAAZAZAZAZAZ...	Kids Men Women Seniors Kids Men Women Seniors...	21.603175	21.603175

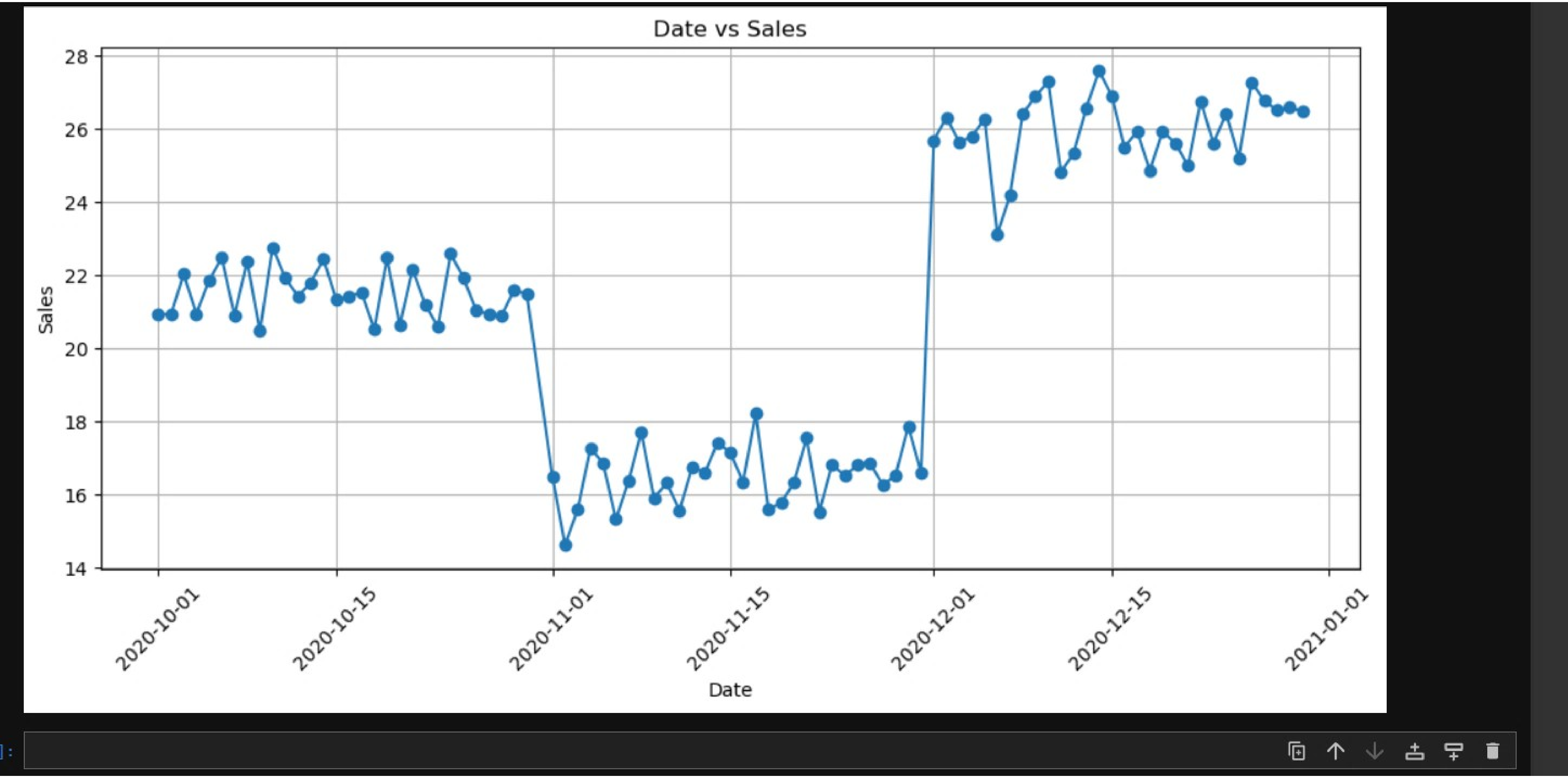


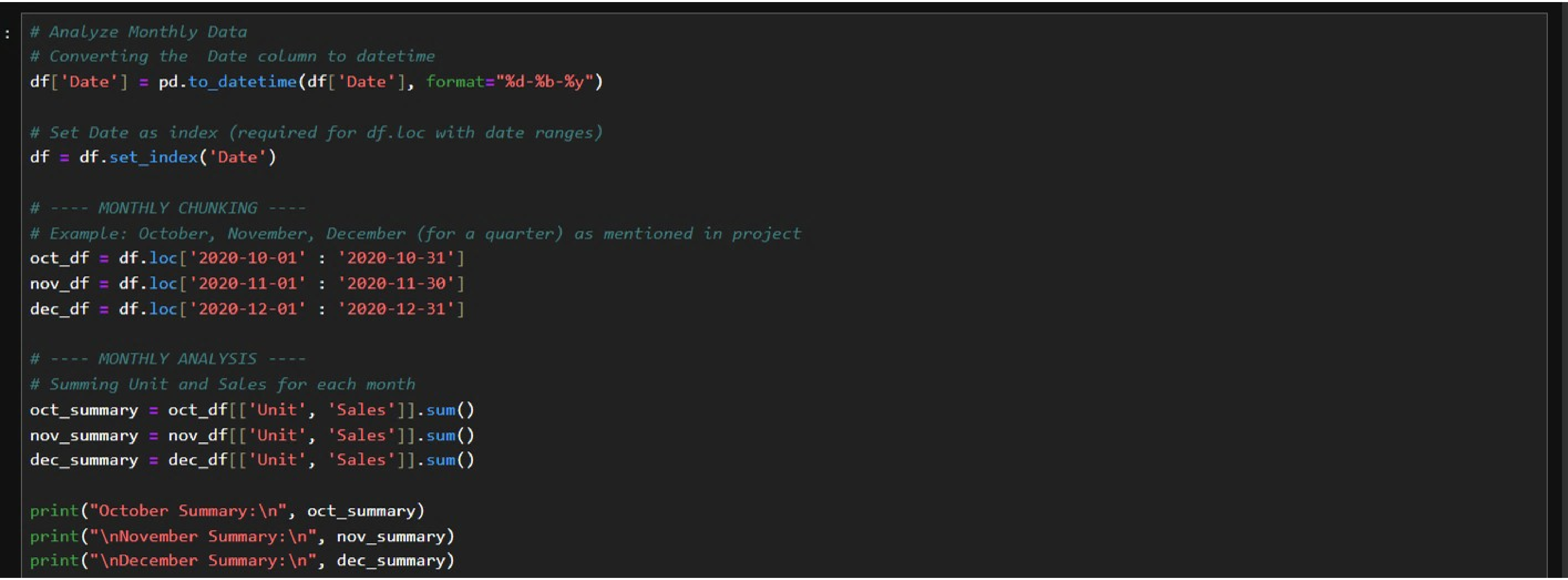
```python
# Converting Date to proper datetime format
df['Date'] = pd.to_datetime(df['Date'], format="%d-%b-%y")

# Grouping Date and sum Unit and Sales
daily = df.groupby('Date')[['Unit', 'Sales']].sum().reset_index()

# Plot Date vs Unit
plt.figure(figsize=(12,5))
plt.plot(daily['Date'], daily['Unit'], marker='o')
plt.xlabel('Date')
plt.ylabel('Unit')
plt.title('Date vs Unit')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```


Date vs Unit
28
26
24
22
20
18
16
14
Unit
2020-10-01
2020-10-15
2020-11-01
2020-11-15
2020-12-01
2020-12-15
2021-01-01
Date

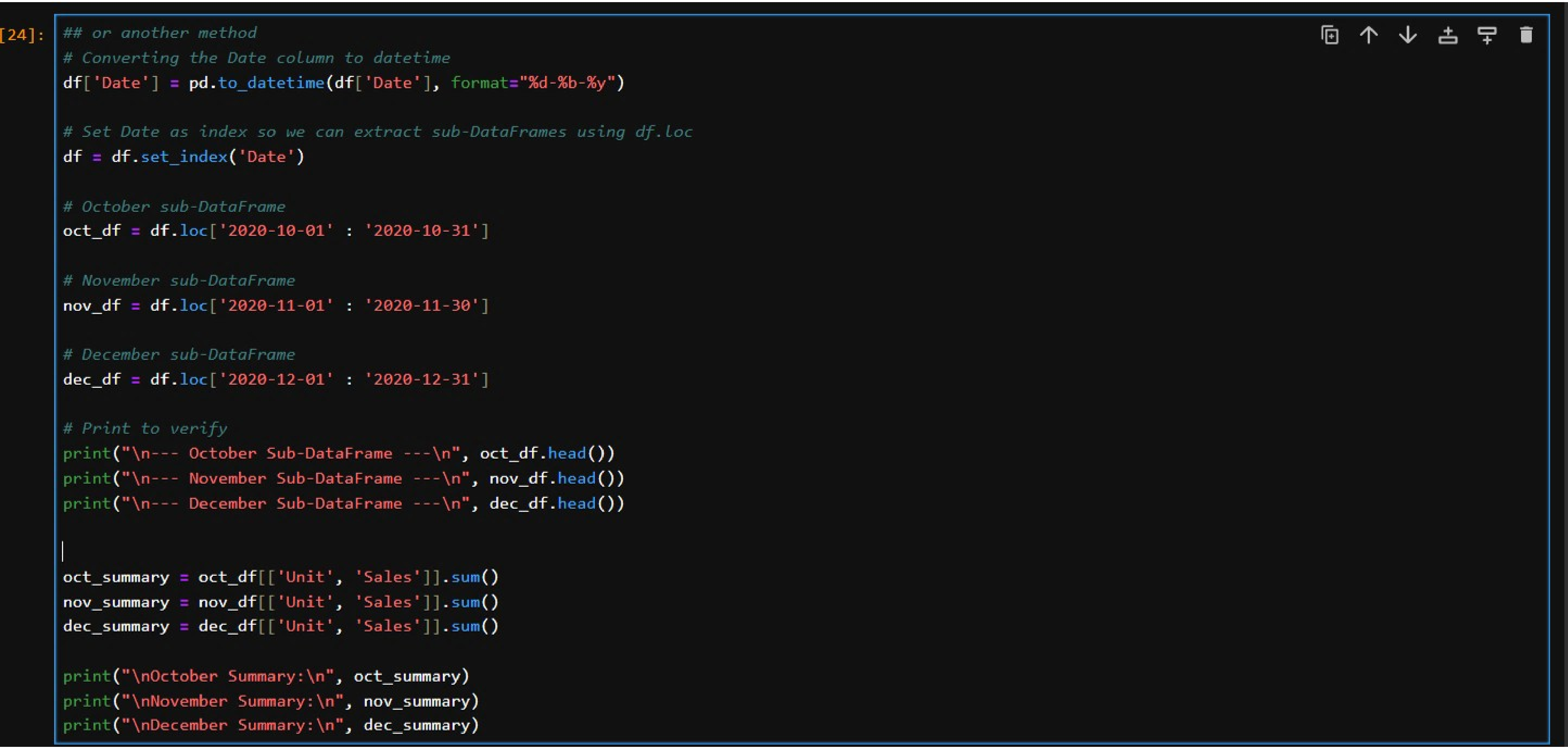


```
# Ploting for Date vs Sales
plt.figure(figsize=(12,5))
plt.plot(daily['Date'], daily['Sales'], marker='o')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Date vs Sales')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

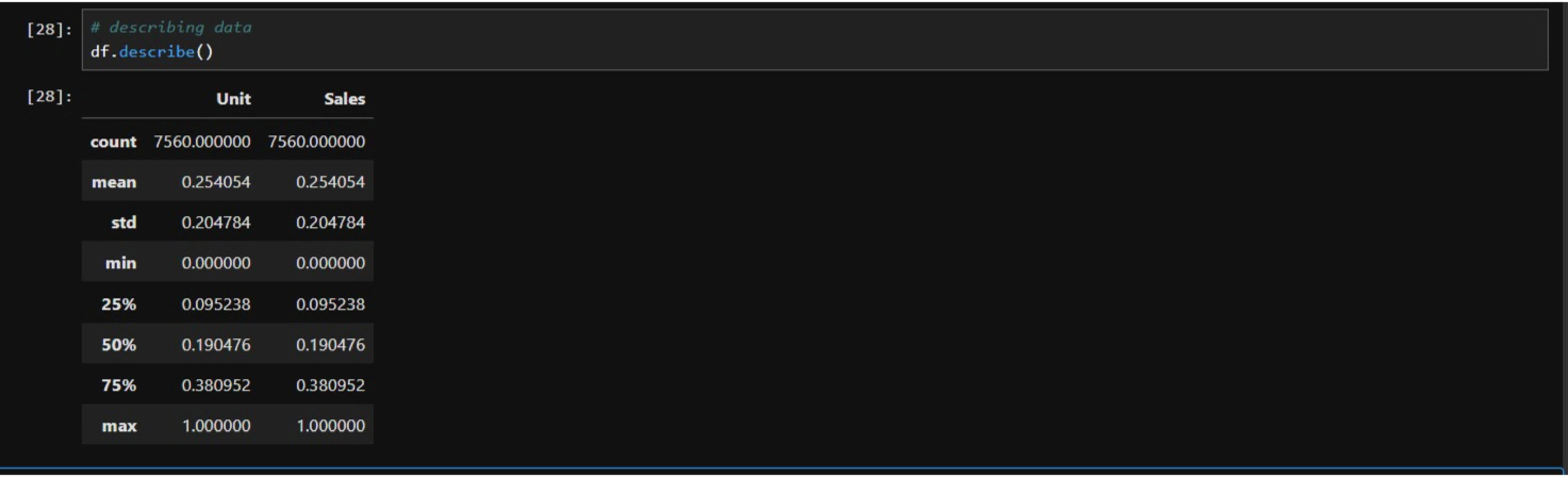

Date vs Sales
Sales
Date
28
26
24
22
20
18
16
14
2020-10-01
2020-10-15
2020-11-01
2020-11-15
2020-12-01
2020-12-15
2021-01-01
]:



```
# Analyze Monthly Data
# Converting the  Date column to datetime
df['Date'] = pd.to_datetime(df['Date'], format="%d-%b-%y")

# Set Date as index (required for df.loc with date ranges)
df = df.set_index('Date')

# ---- MONTHLY CHUNKING ----
# Example: October, November, December (for a quarter) as mentioned in project
oct_df = df.loc['2020-10-01' : '2020-10-31']
nov_df = df.loc['2020-11-01' : '2020-11-30']
dec_df = df.loc['2020-12-01' : '2020-12-31']

# ---- MONTHLY ANALYSIS ----
# Summing Unit and Sales for each month
oct_summary = oct_df[['Unit', 'Sales']].sum()
nov_summary = nov_df[['Unit', 'Sales']].sum()
dec_summary = dec_df[['Unit', 'Sales']].sum()

print("October Summary:\n", oct_summary)
print("\nNovember Summary:\n", nov_summary)
print("\nDecember Summary:\n", dec_summary)
```



```
## or another method
# Converting the Date column to datetime
df['Date'] = pd.to_datetime(df['Date'], format="%d-%b-%y")

# Set Date as index so we can extract sub-DataFrames using df.loc
df = df.set_index('Date')

# October sub-DataFrame
oct_df = df.loc['2020-10-01' : '2020-10-31']

# November sub-DataFrame
nov_df = df.loc['2020-11-01' : '2020-11-30']

# December sub-DataFrame
dec_df = df.loc['2020-12-01' : '2020-12-31']

# Print to verify
print("\n--- October Sub-DataFrame ---\n", oct_df.head())
print("\n--- November Sub-DataFrame ---\n", nov_df.head())
print("\n--- December Sub-DataFrame ---\n", dec_df.head())


oct_summary = oct_df[['Unit', 'Sales']].sum()
nov_summary = nov_df[['Unit', 'Sales']].sum()
dec_summary = dec_df[['Unit', 'Sales']].sum()

print("\nOctober Summary:\n", oct_summary)
print("\nNovember Summary:\n", nov_summary)
print("\nDecember Summary:\n", dec_summary)
```

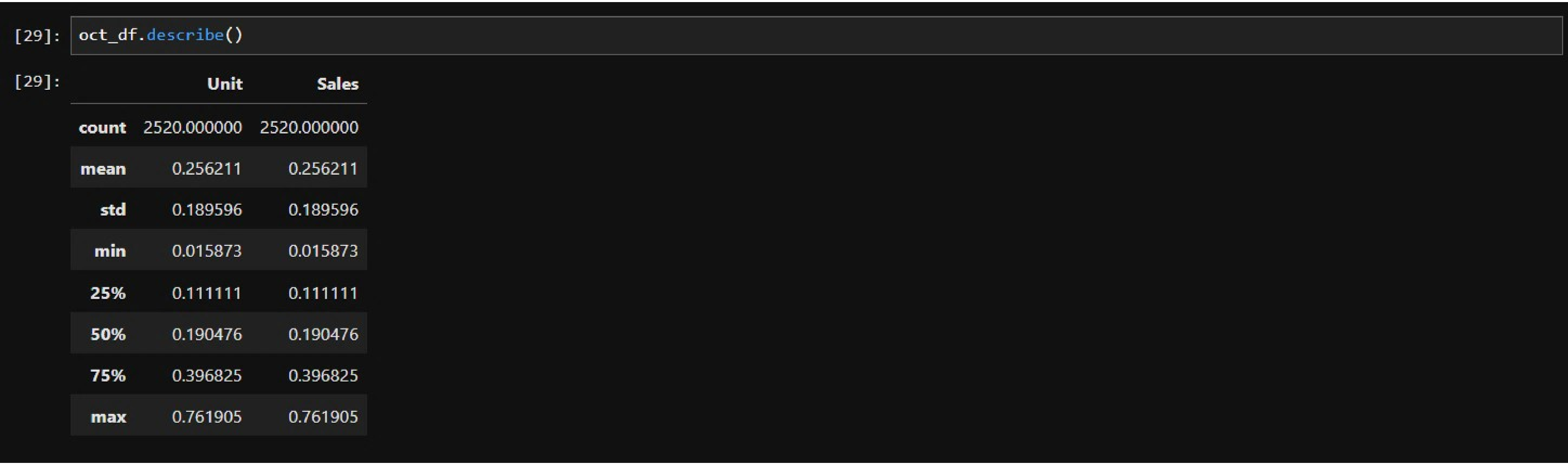


[28]:

```
# describing data
df.describe()
```

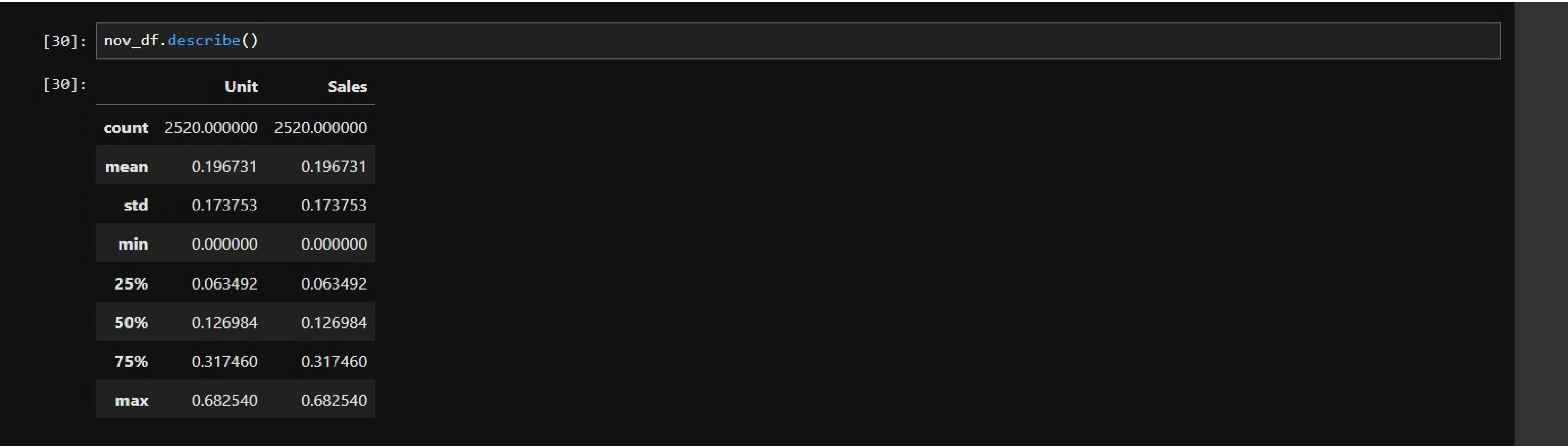
[28]:

	Unit	Sales
count	7560.000000	7560.000000
mean	0.254054	0.254054
std	0.204784	0.204784
min	0.000000	0.000000
25%	0.095238	0.095238
50%	0.190476	0.190476
75%	0.380952	0.380952
max	1.000000	1.000000



```
[29]: oct_df.describe()
```

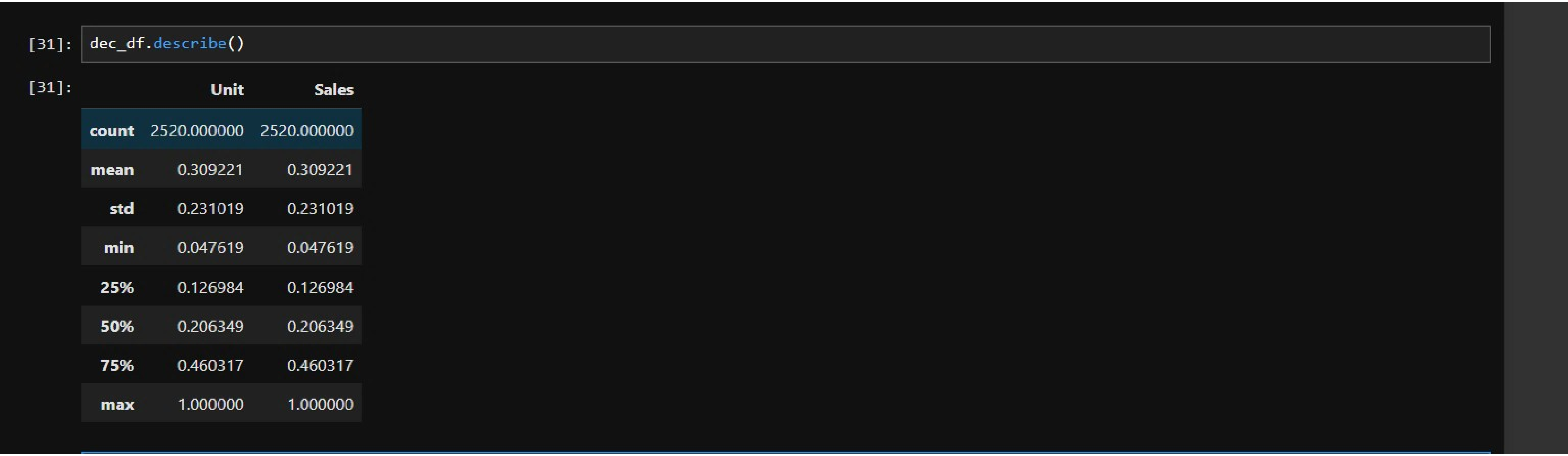
```
[29]:
              Unit        Sales
count  2520.000000  2520.000000
mean      0.256211     0.256211
std       0.189596     0.189596
min       0.015873     0.015873
25%       0.111111     0.111111
50%       0.190476     0.190476
75%       0.396825     0.396825
max       0.761905     0.761905
```



```
[30]: nov_df.describe()
```

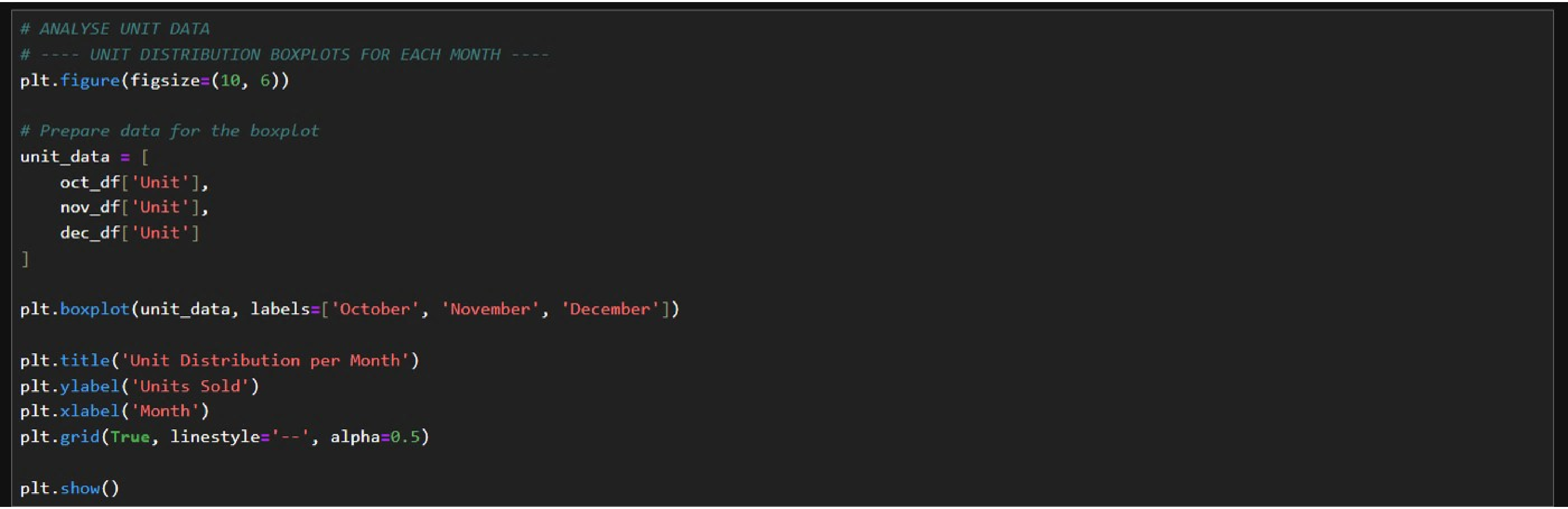
[30]:

	Unit	Sales
count	2520.000000	2520.000000
mean	0.196731	0.196731
std	0.173753	0.173753
min	0.000000	0.000000
25%	0.063492	0.063492
50%	0.126984	0.126984
75%	0.317460	0.317460
max	0.682540	0.682540

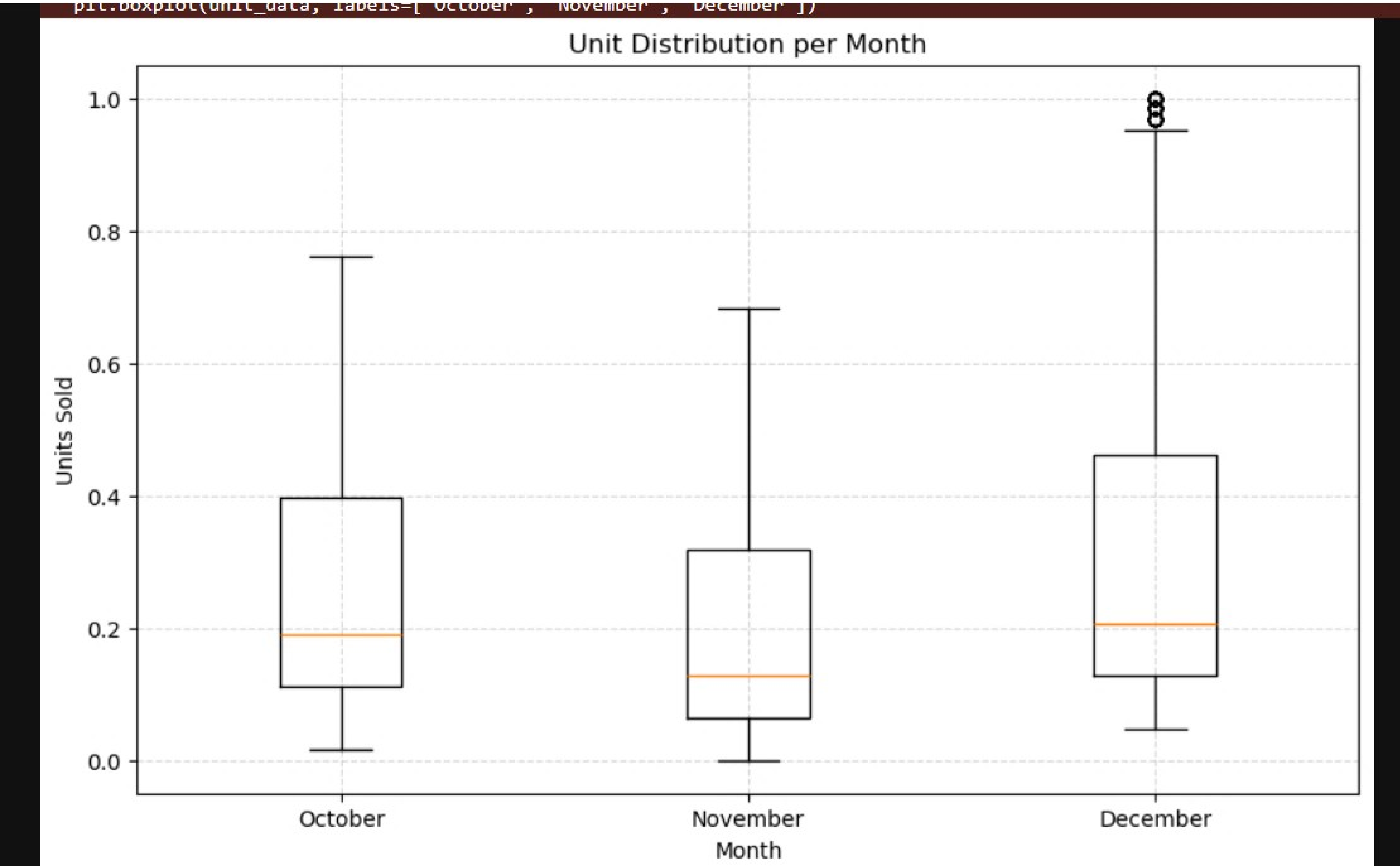


```
[31]: dec_df.describe()
```

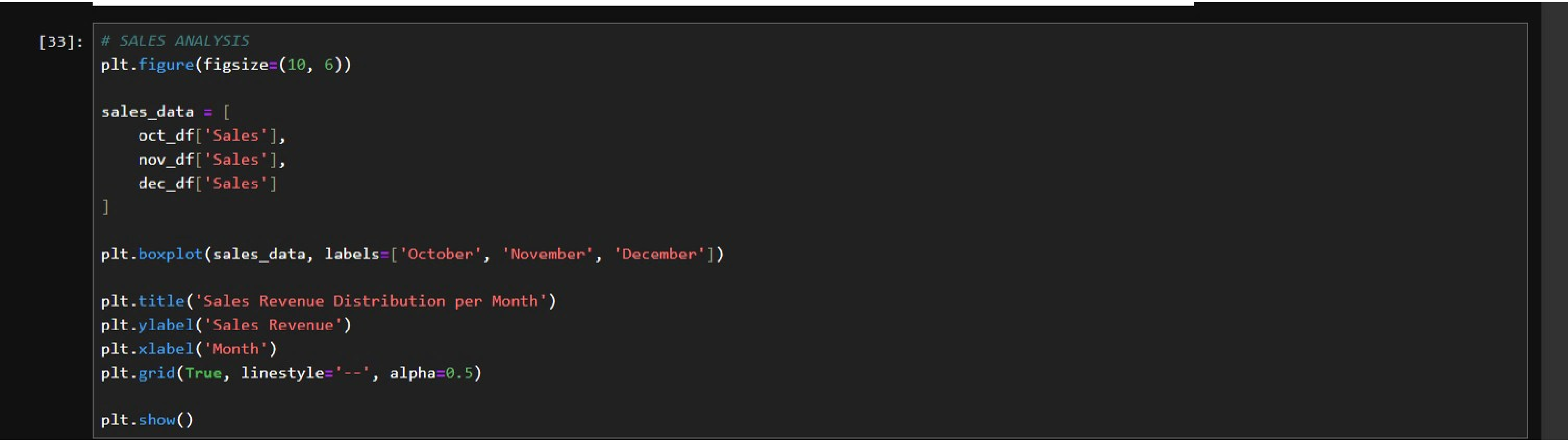
```
[31]:
              Unit        Sales
count  2520.000000  2520.000000
mean      0.309221     0.309221
std       0.231019     0.231019
min       0.047619     0.047619
25%       0.126984     0.126984
50%       0.206349     0.206349
75%       0.460317     0.460317
max       1.000000     1.000000
```



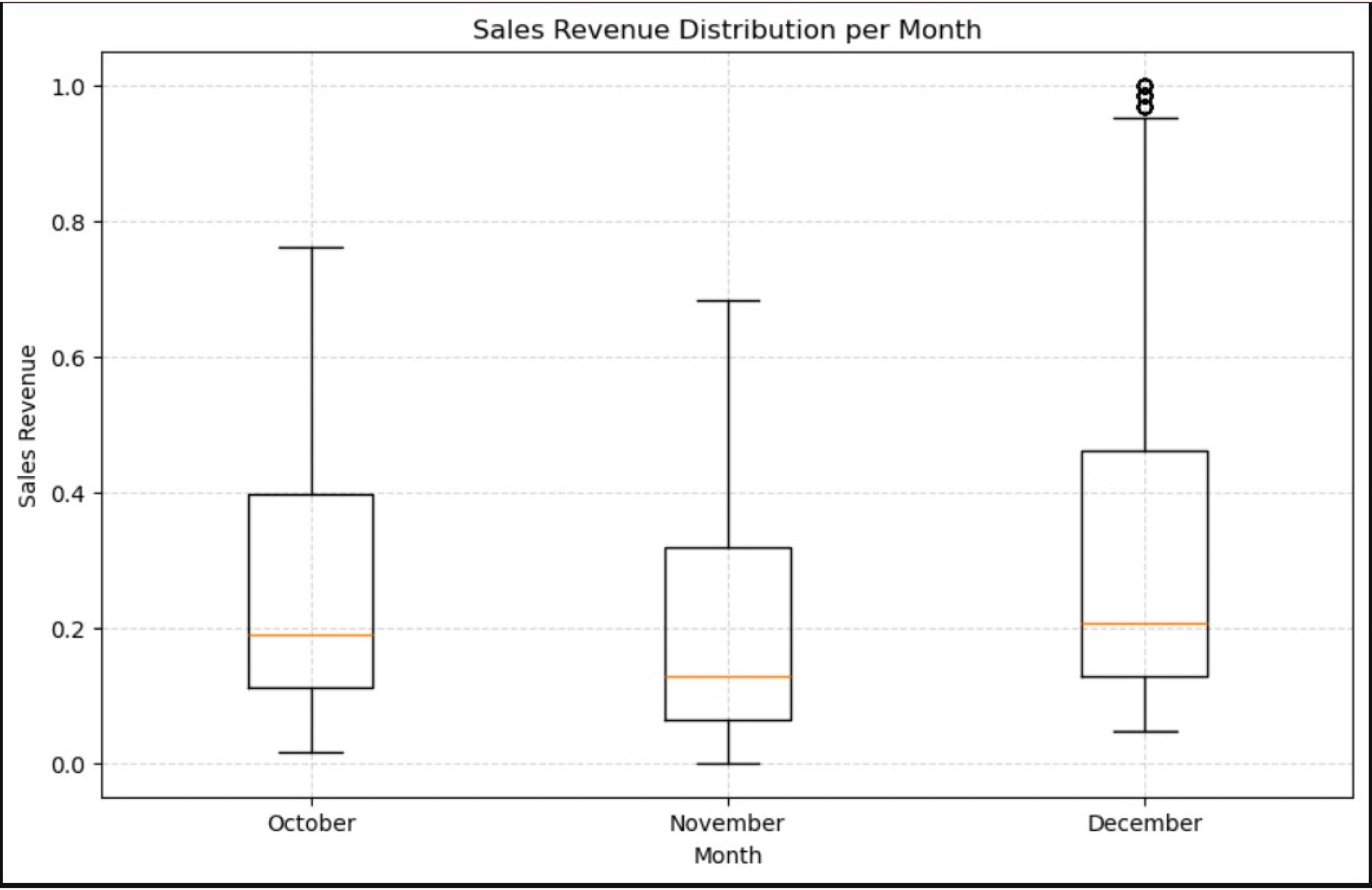
```
# ANALYSE UNIT DATA
# ---- UNIT DISTRIBUTION BOXPLOTS FOR EACH MONTH ----
plt.figure(figsize=(10, 6))

# Prepare data for the boxplot
unit_data = [
    oct_df['Unit'],
    nov_df['Unit'],
    dec_df['Unit']
]

plt.boxplot(unit_data, labels=['October', 'November', 'December'])

plt.title('Unit Distribution per Month')
plt.ylabel('Units Sold')
plt.xlabel('Month')
plt.grid(True, linestyle='--', alpha=0.5)

plt.show()
```

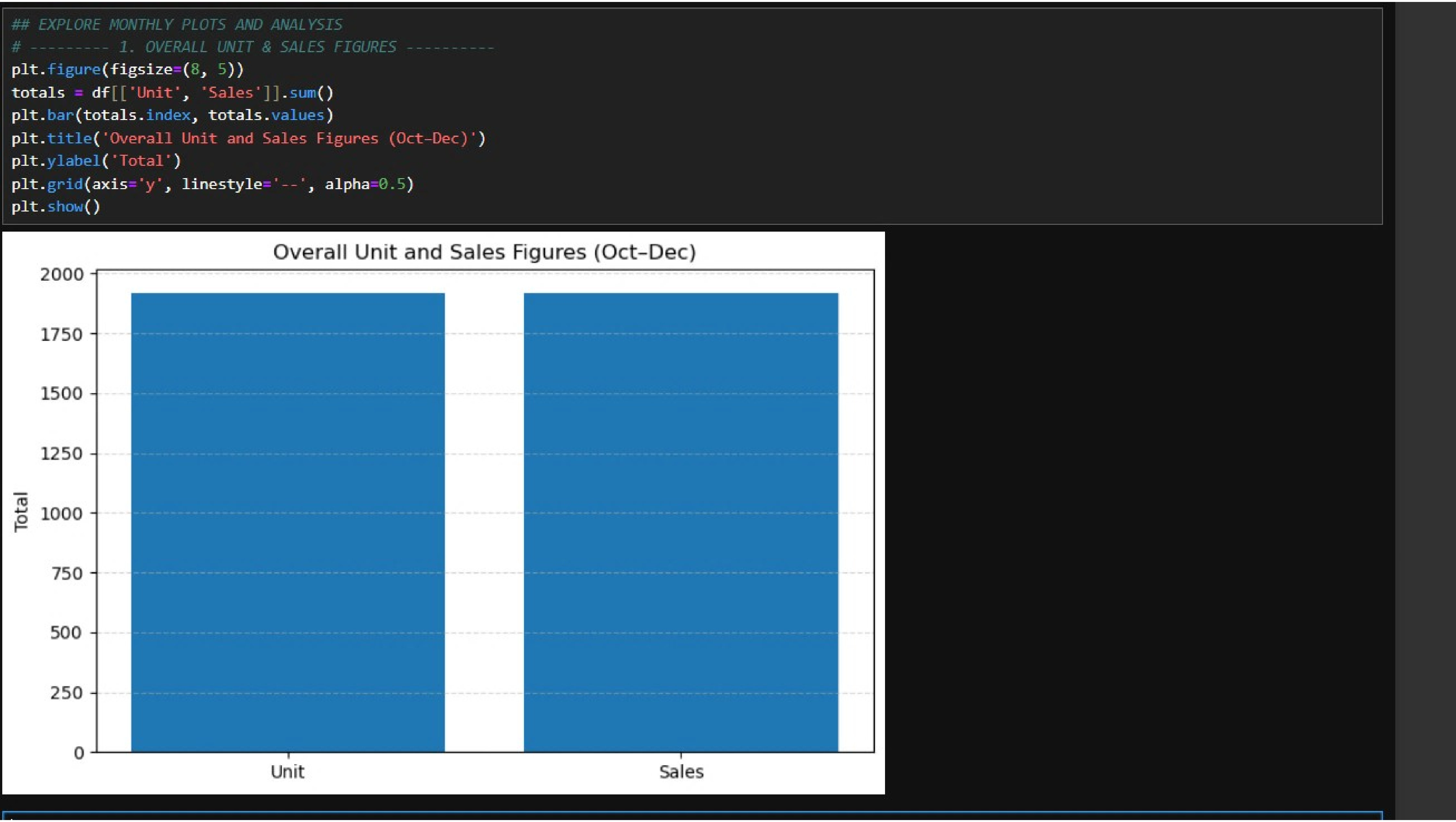


```
plt.boxplot(unit_data, labels=['October', 'November', 'December'])
```

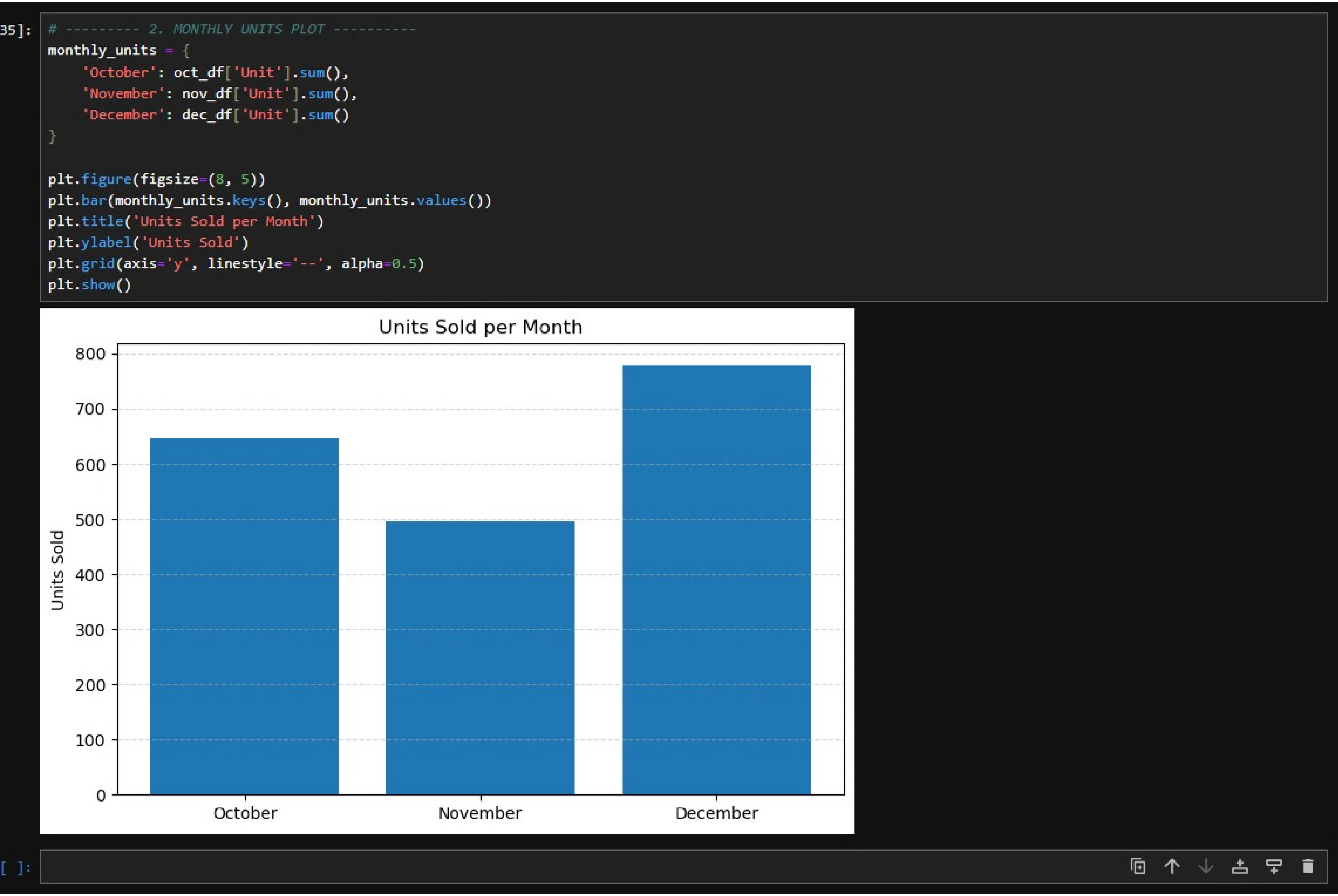


```
# SALES ANALYSIS
plt.figure(figsize=(10, 6))

sales_data = [
    oct_df['Sales'],
    nov_df['Sales'],
    dec_df['Sales']
]

plt.boxplot(sales_data, labels=['October', 'November', 'December'])

plt.title('Sales Revenue Distribution per Month')
plt.ylabel('Sales Revenue')
plt.xlabel('Month')
plt.grid(True, linestyle='--', alpha=0.5)

plt.show()
```

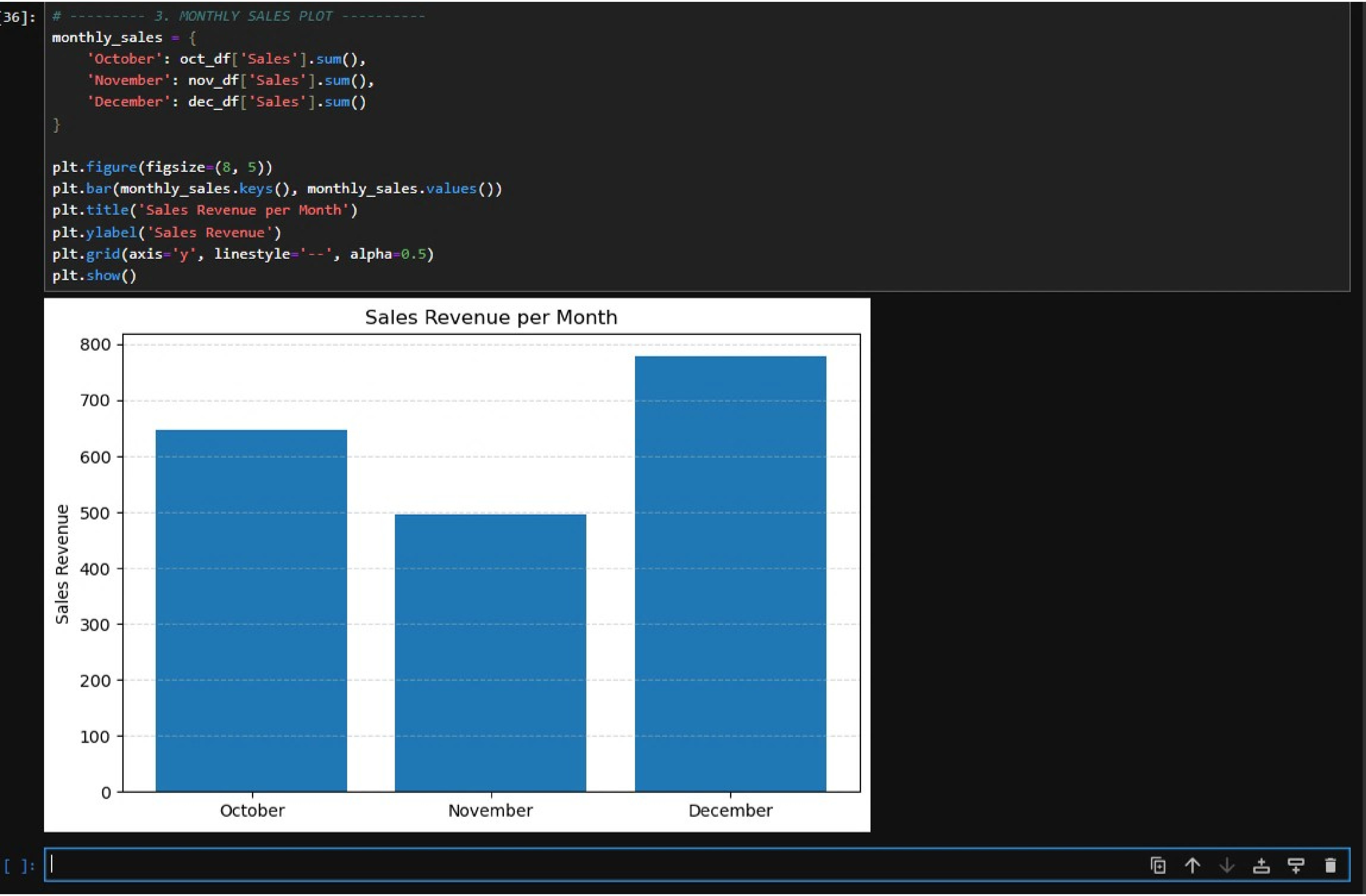

Sales Revenue Distribution per Month
Sales Revenue
1.0
0.8
0.6
0.4
0.2
0.0
October
November
December
Month



```
## EXPLORE MONTHLY PLOTS AND ANALYSIS
# --------- 1. OVERALL UNIT & SALES FIGURES ----------
plt.figure(figsize=(8, 5))
totals = df[['Unit', 'Sales']].sum()
plt.bar(totals.index, totals.values)
plt.title('Overall Unit and Sales Figures (Oct–Dec)')
plt.ylabel('Total')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.show()
```



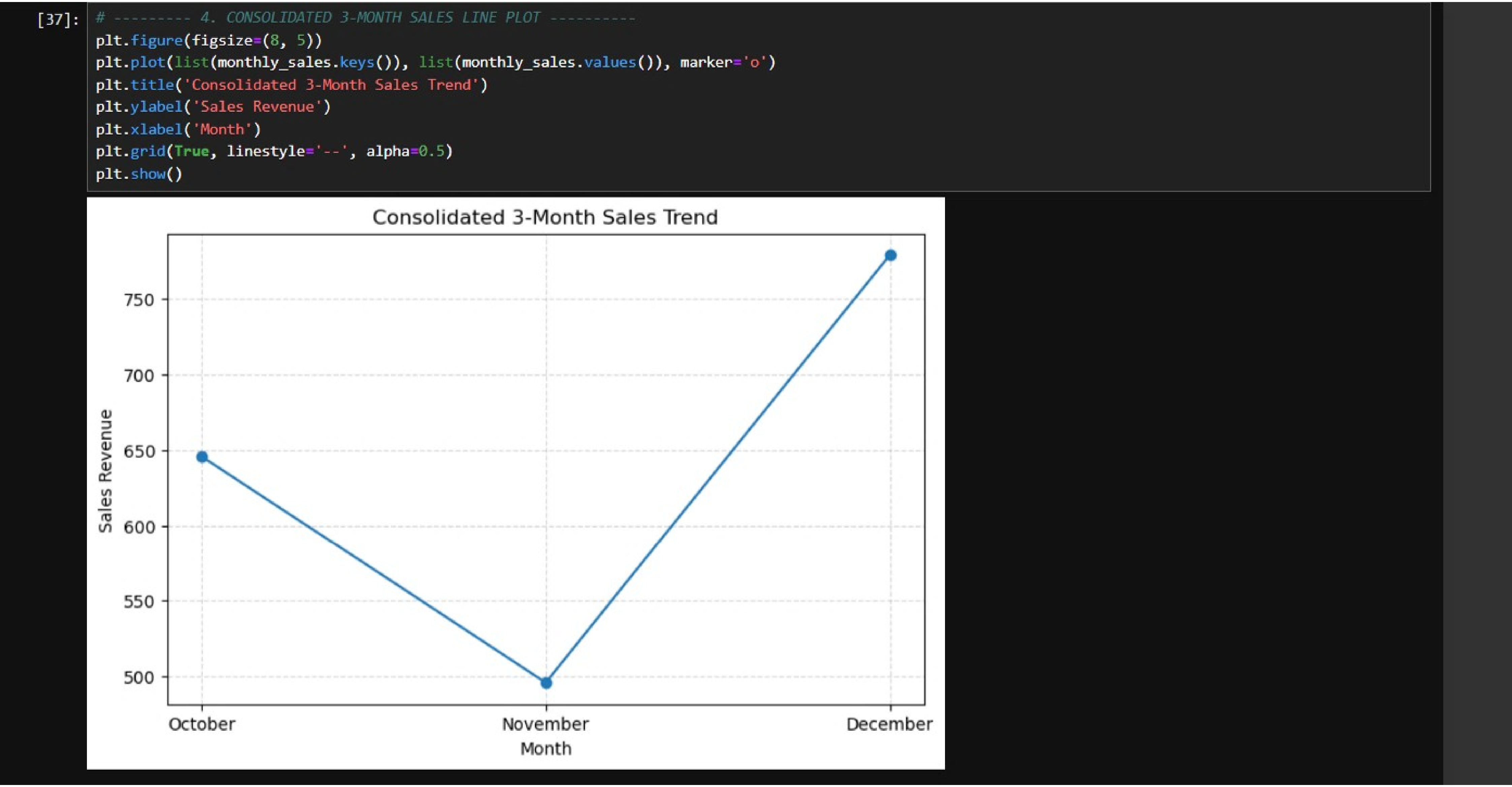
```
# --------- 2. MONTHLY UNITS PLOT ----------
monthly_units = {
    'October': oct_df['Unit'].sum(),
    'November': nov_df['Unit'].sum(),
    'December': dec_df['Unit'].sum()
}

plt.figure(figsize=(8, 5))
plt.bar(monthly_units.keys(), monthly_units.values())
plt.title('Units Sold per Month')
plt.ylabel('Units Sold')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.show()
```



[36]:

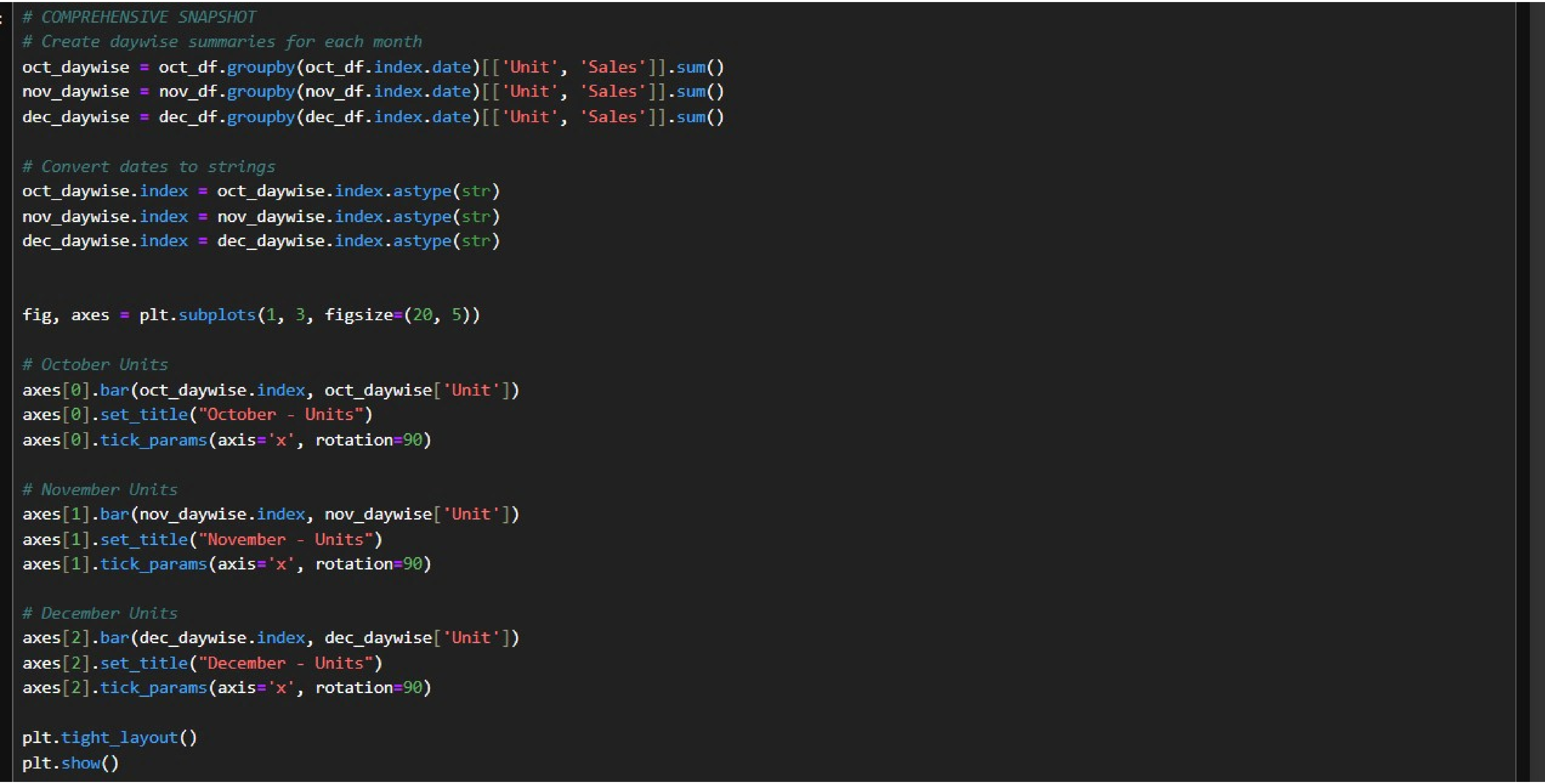
```
# --------- 3. MONTHLY SALES PLOT ---------
monthly_sales = {
    'October': oct_df['Sales'].sum(),
    'November': nov_df['Sales'].sum(),
    'December': dec_df['Sales'].sum()
}

plt.figure(figsize=(8, 5))
plt.bar(monthly_sales.keys(), monthly_sales.values())
plt.title('Sales Revenue per Month')
plt.ylabel('Sales Revenue')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.show()
```

[ ]:

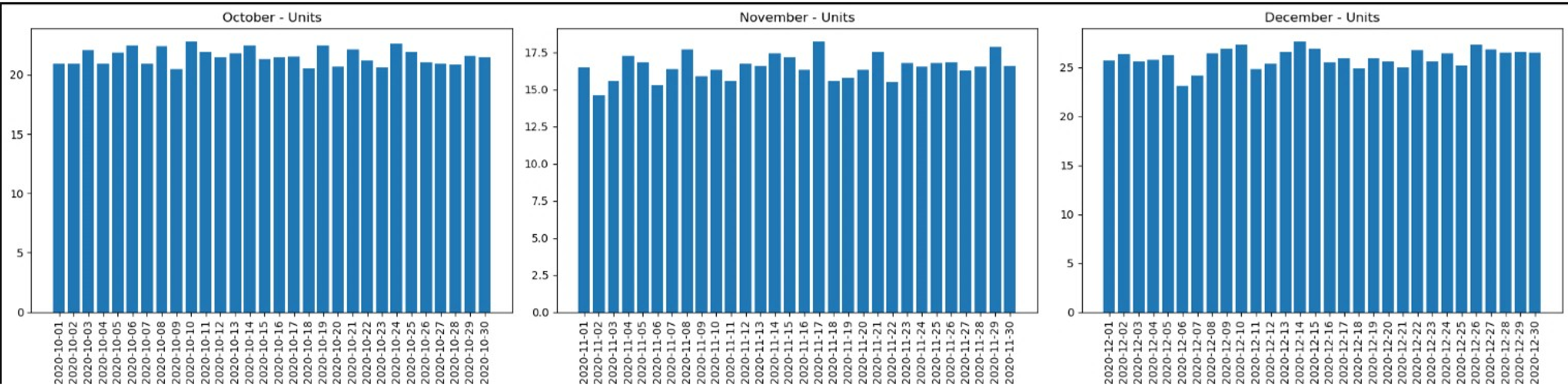


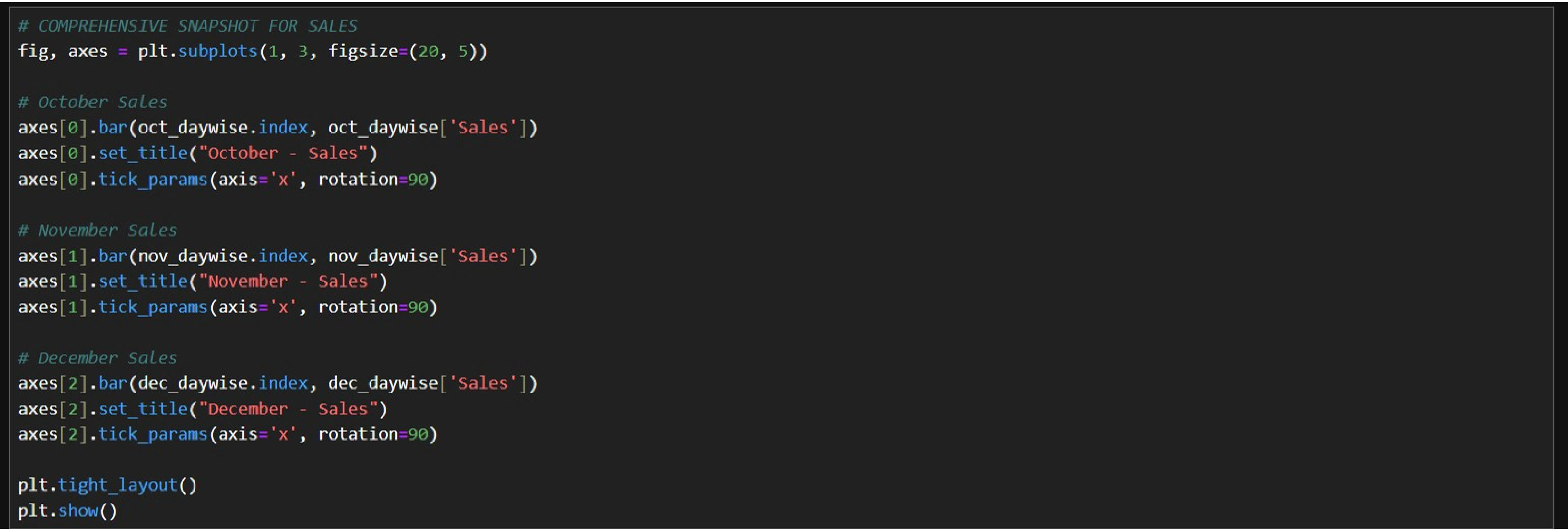
[37]:

```
# --------- 4. CONSOLIDATED 3-MONTH SALES LINE PLOT ----------
plt.figure(figsize=(8, 5))
plt.plot(list(monthly_sales.keys()), list(monthly_sales.values()), marker='o')
plt.title('Consolidated 3-Month Sales Trend')
plt.ylabel('Sales Revenue')
plt.xlabel('Month')
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```

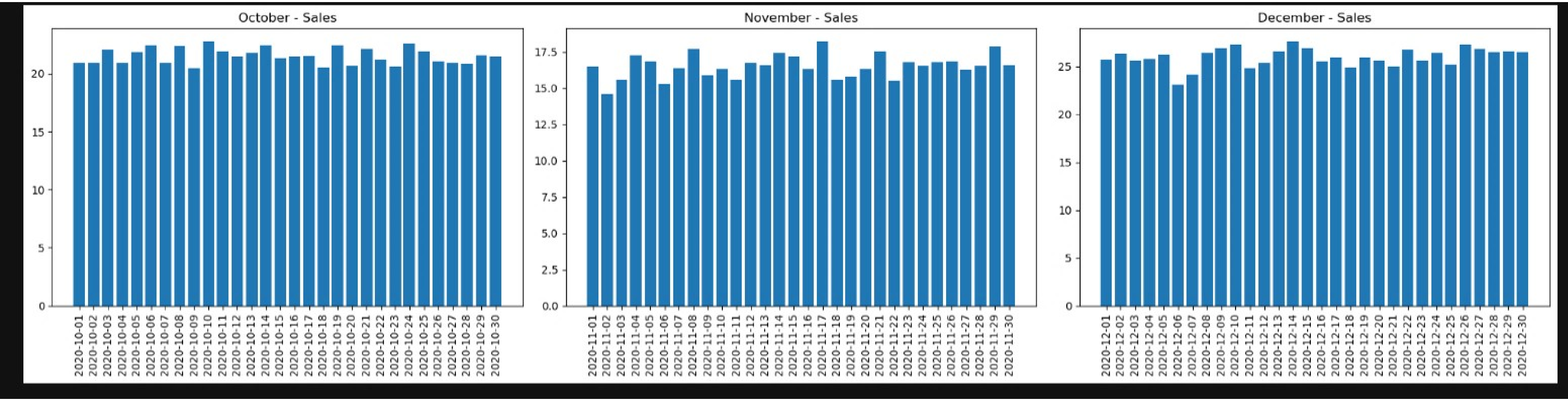


```
# COMPREHENSIVE SNAPSHOT
# Create daywise summaries for each month
oct_daywise = oct_df.groupby(oct_df.index.date)[['Unit', 'Sales']].sum()
nov_daywise = nov_df.groupby(nov_df.index.date)[['Unit', 'Sales']].sum()
dec_daywise = dec_df.groupby(dec_df.index.date)[['Unit', 'Sales']].sum()

# Convert dates to strings
oct_daywise.index = oct_daywise.index.astype(str)
nov_daywise.index = nov_daywise.index.astype(str)
dec_daywise.index = dec_daywise.index.astype(str)


fig, axes = plt.subplots(1, 3, figsize=(20, 5))

# October Units
axes[0].bar(oct_daywise.index, oct_daywise['Unit'])
axes[0].set_title("October - Units")
axes[0].tick_params(axis='x', rotation=90)

# November Units
axes[1].bar(nov_daywise.index, nov_daywise['Unit'])
axes[1].set_title("November - Units")
axes[1].tick_params(axis='x', rotation=90)

# December Units
axes[2].bar(dec_daywise.index, dec_daywise['Unit'])
axes[2].set_title("December - Units")
axes[2].tick_params(axis='x', rotation=90)

plt.tight_layout()
plt.show()
```

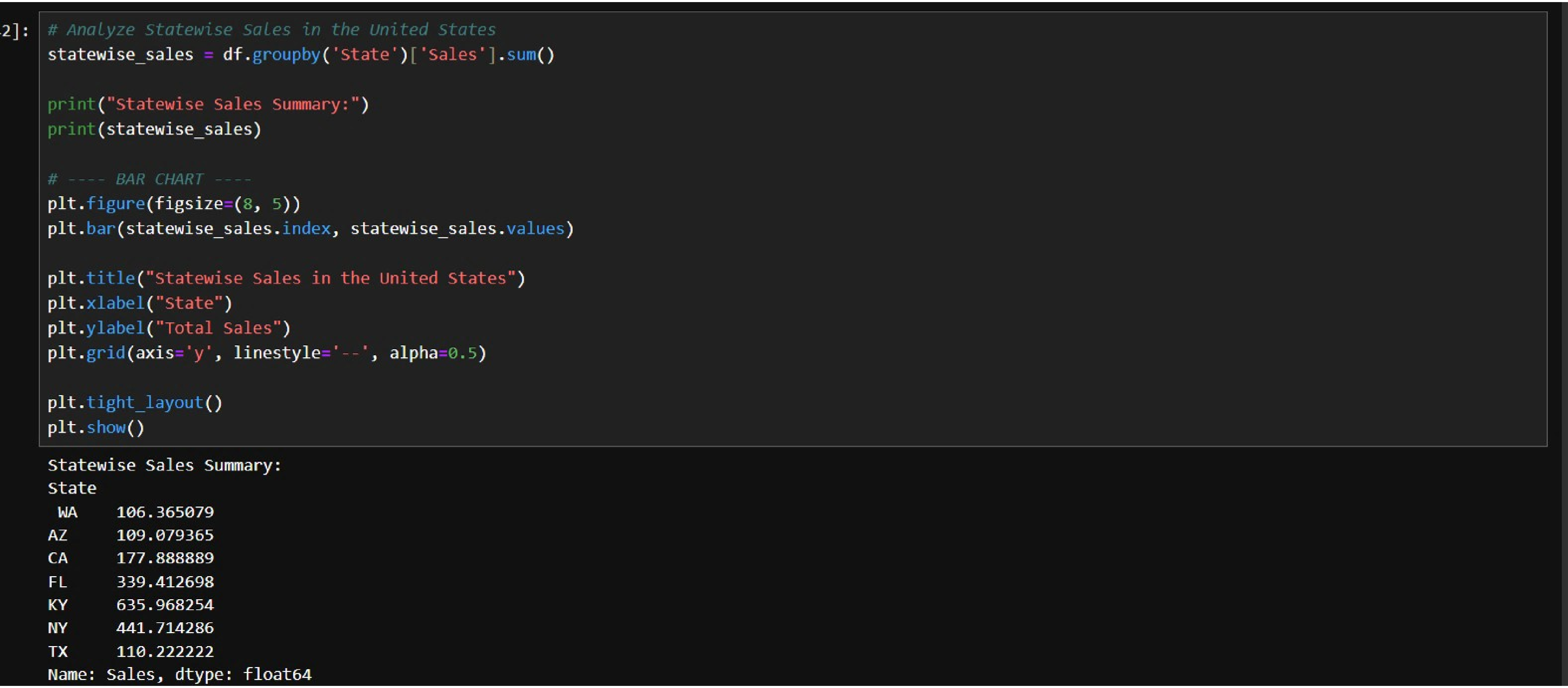

October - Units
0
5
10
15
20
2020-10-01
2020-10-02
2020-10-03
2020-10-04
2020-10-05
2020-10-06
2020-10-07
2020-10-08
2020-10-09
2020-10-10
2020-10-11
2020-10-12
2020-10-13
2020-10-14
2020-10-15
2020-10-16
2020-10-17
2020-10-18
2020-10-19
2020-10-20
2020-10-21
2020-10-22
2020-10-23
2020-10-24
2020-10-25
2020-10-26
2020-10-27
2020-10-28
2020-10-29
2020-10-30
November - Units
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
2020-11-01
2020-11-02
2020-11-03
2020-11-04
2020-11-05
2020-11-06
2020-11-07
2020-11-08
2020-11-09
2020-11-10
2020-11-11
2020-11-12
2020-11-13
2020-11-14
2020-11-15
2020-11-16
2020-11-17
2020-11-18
2020-11-19
2020-11-20
2020-11-21
2020-11-22
2020-11-23
2020-11-24
2020-11-25
2020-11-26
2020-11-27
2020-11-28
2020-11-29
2020-11-30
December - Units
0
5
10
15
20
25
2020-12-01
2020-12-02
2020-12-03
2020-12-04
2020-12-05
2020-12-06
2020-12-07
2020-12-08
2020-12-09
2020-12-10
2020-12-11
2020-12-12
2020-12-13
2020-12-14
2020-12-15
2020-12-16
2020-12-17
2020-12-18
2020-12-19
2020-12-20
2020-12-21
2020-12-22
2020-12-23
2020-12-24
2020-12-25
2020-12-26
2020-12-27
2020-12-28
2020-12-29
2020-12-30



```
# COMPREHENSIVE SNAPSHOT FOR SALES
fig, axes = plt.subplots(1, 3, figsize=(20, 5))

# October Sales
axes[0].bar(oct_daywise.index, oct_daywise['Sales'])
axes[0].set_title("October - Sales")
axes[0].tick_params(axis='x', rotation=90)

# November Sales
axes[1].bar(nov_daywise.index, nov_daywise['Sales'])
axes[1].set_title("November - Sales")
axes[1].tick_params(axis='x', rotation=90)

# December Sales
axes[2].bar(dec_daywise.index, dec_daywise['Sales'])
axes[2].set_title("December - Sales")
axes[2].tick_params(axis='x', rotation=90)

plt.tight_layout()
plt.show()
```

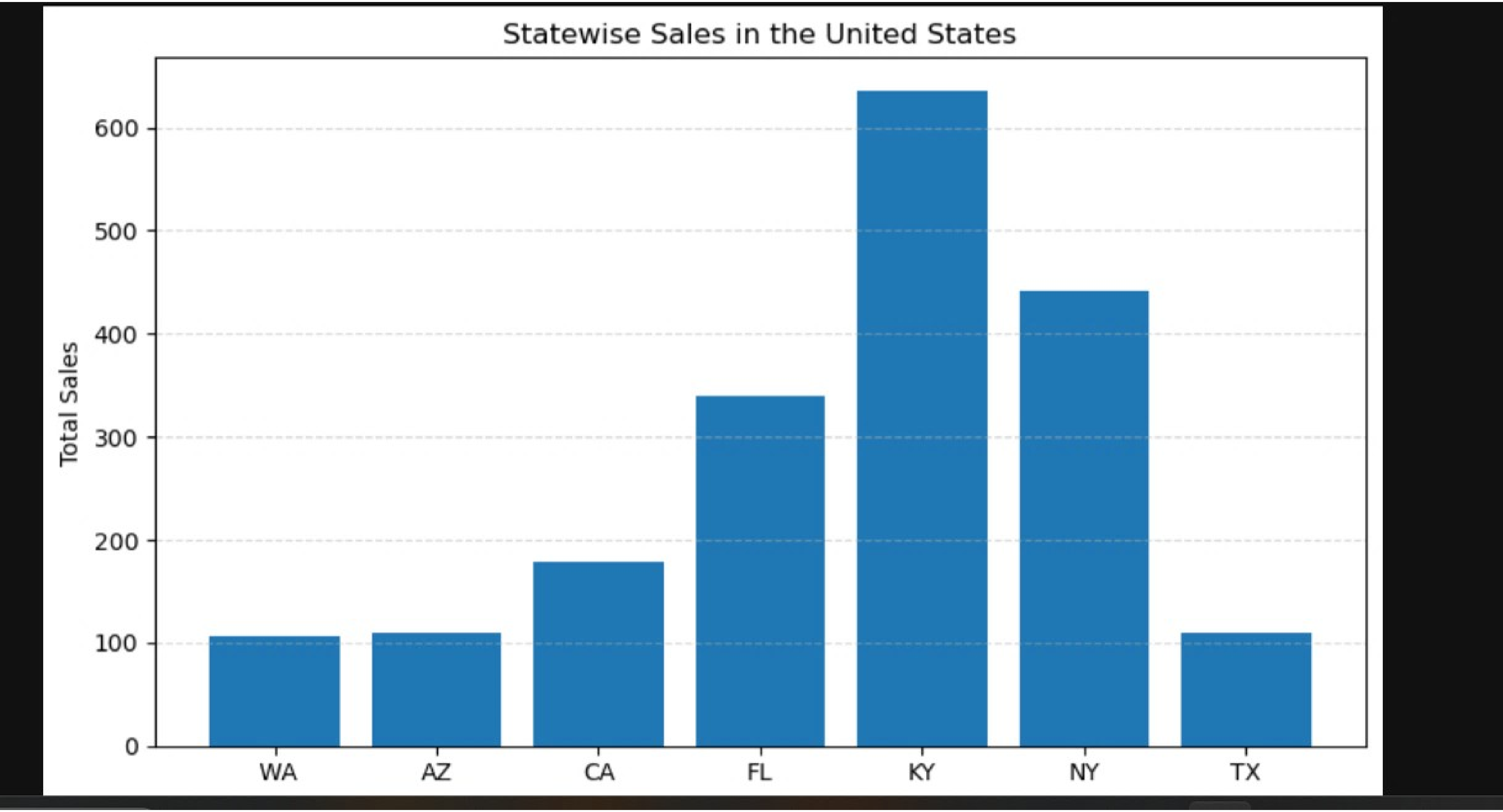


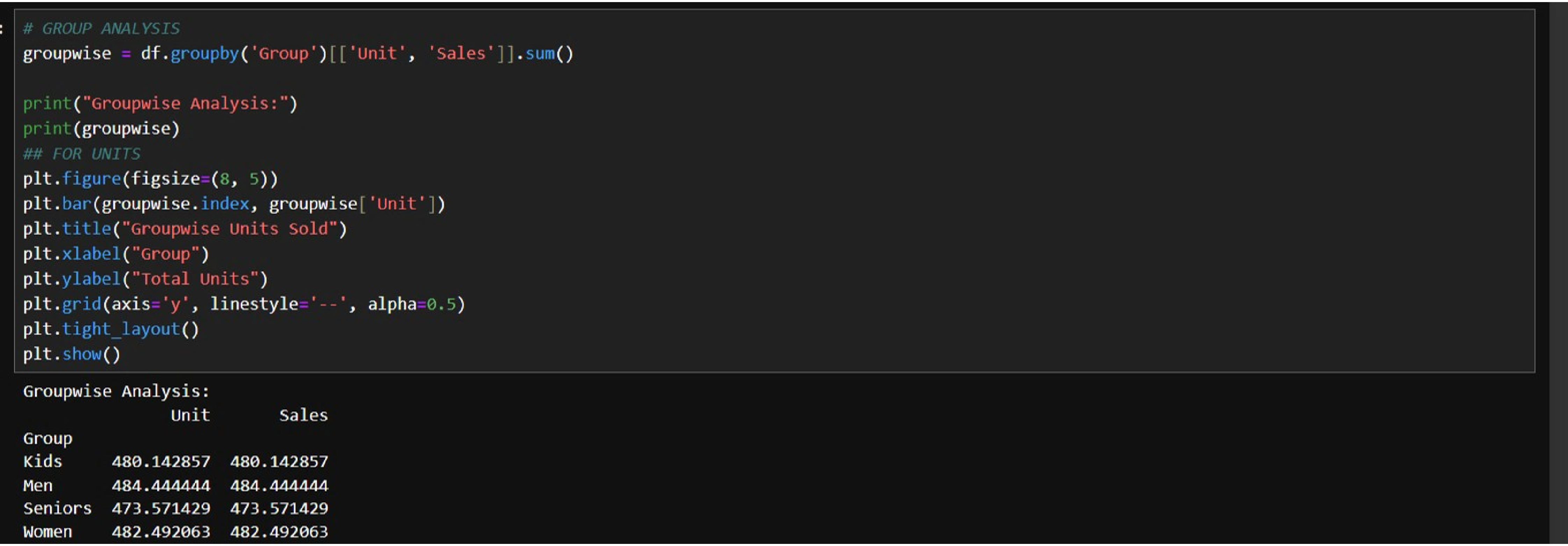
October - Sales
0
5
10
15
20
2020-10-01
2020-10-02
2020-10-03
2020-10-04
2020-10-05
2020-10-06
2020-10-07
2020-10-08
2020-10-09
2020-10-10
2020-10-11
2020-10-12
2020-10-13
2020-10-14
2020-10-15
2020-10-16
2020-10-17
2020-10-18
2020-10-19
2020-10-20
2020-10-21
2020-10-22
2020-10-23
2020-10-24
2020-10-25
2020-10-26
2020-10-27
2020-10-28
2020-10-29
2020-10-30
November - Sales
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
2020-11-01
2020-11-02
2020-11-03
2020-11-04
2020-11-05
2020-11-06
2020-11-07
2020-11-08
2020-11-09
2020-11-10
2020-11-11
2020-11-12
2020-11-13
2020-11-14
2020-11-15
2020-11-16
2020-11-17
2020-11-18
2020-11-19
2020-11-20
2020-11-21
2020-11-22
2020-11-23
2020-11-24
2020-11-25
2020-11-26
2020-11-27
2020-11-28
2020-11-29
2020-11-30
December - Sales
0
5
10
15
20
25
2020-12-01
2020-12-02
2020-12-03
2020-12-04
2020-12-05
2020-12-06
2020-12-07
2020-12-08
2020-12-09
2020-12-10
2020-12-11
2020-12-12
2020-12-13
2020-12-14
2020-12-15
2020-12-16
2020-12-17
2020-12-18
2020-12-19
2020-12-20
2020-12-21
2020-12-22
2020-12-23
2020-12-24
2020-12-25
2020-12-26
2020-12-27
2020-12-28
2020-12-29
2020-12-30



```python
# Analyze Statewise Sales in the United States
statewise_sales = df.groupby('State')['Sales'].sum()

print("Statewise Sales Summary:")
print(statewise_sales)

# ---- BAR CHART ----
plt.figure(figsize=(8, 5))
plt.bar(statewise_sales.index, statewise_sales.values)

plt.title("Statewise Sales in the United States")
plt.xlabel("State")
plt.ylabel("Total Sales")
plt.grid(axis='y', linestyle='--', alpha=0.5)

plt.tight_layout()
plt.show()
```

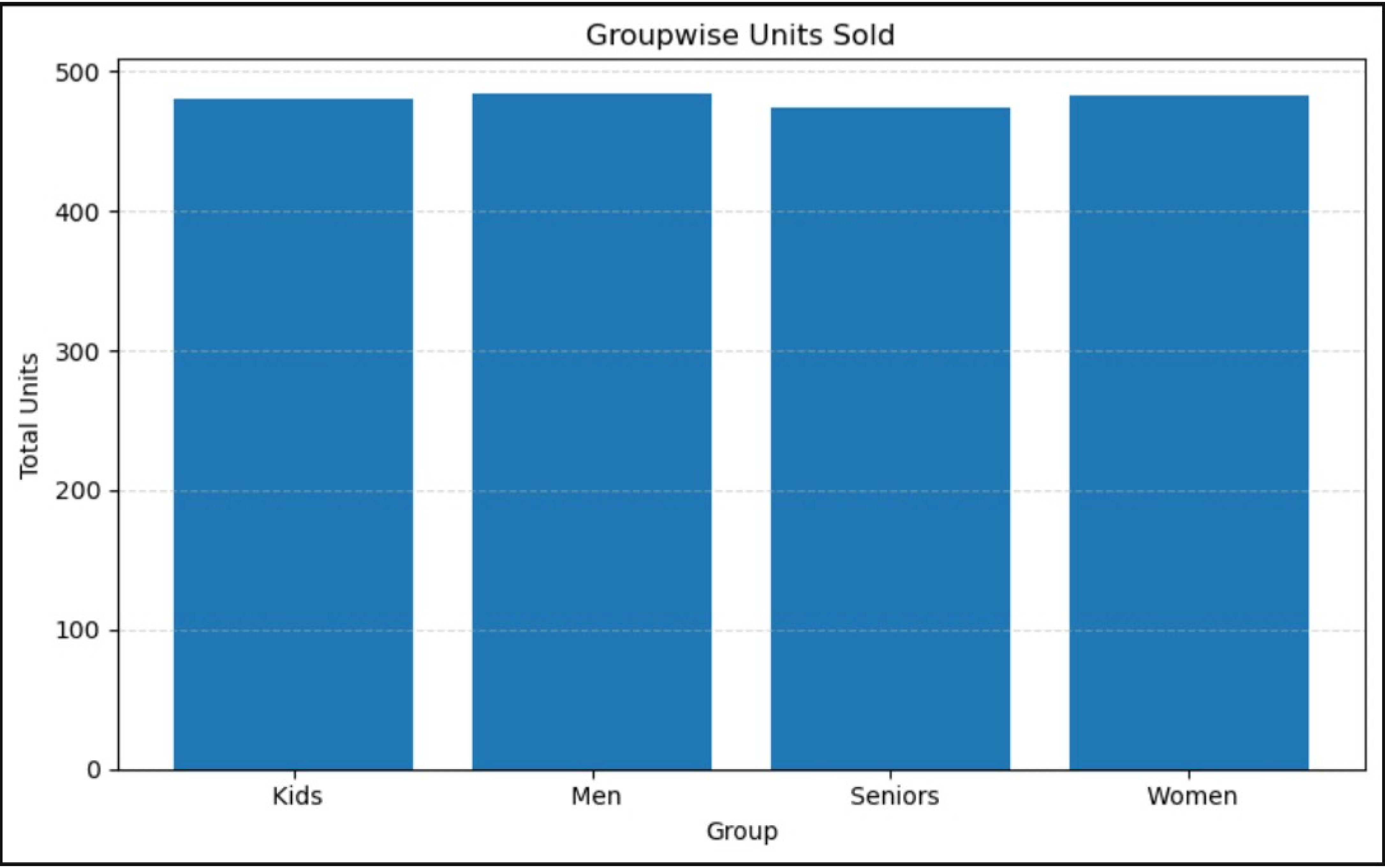
```
Statewise Sales Summary:
State
 WA    106.365079
AZ     109.079365
CA     177.888889
FL     339.412698
KY     635.968254
NY     441.714286
TX     110.222222
Name: Sales, dtype: float64
```

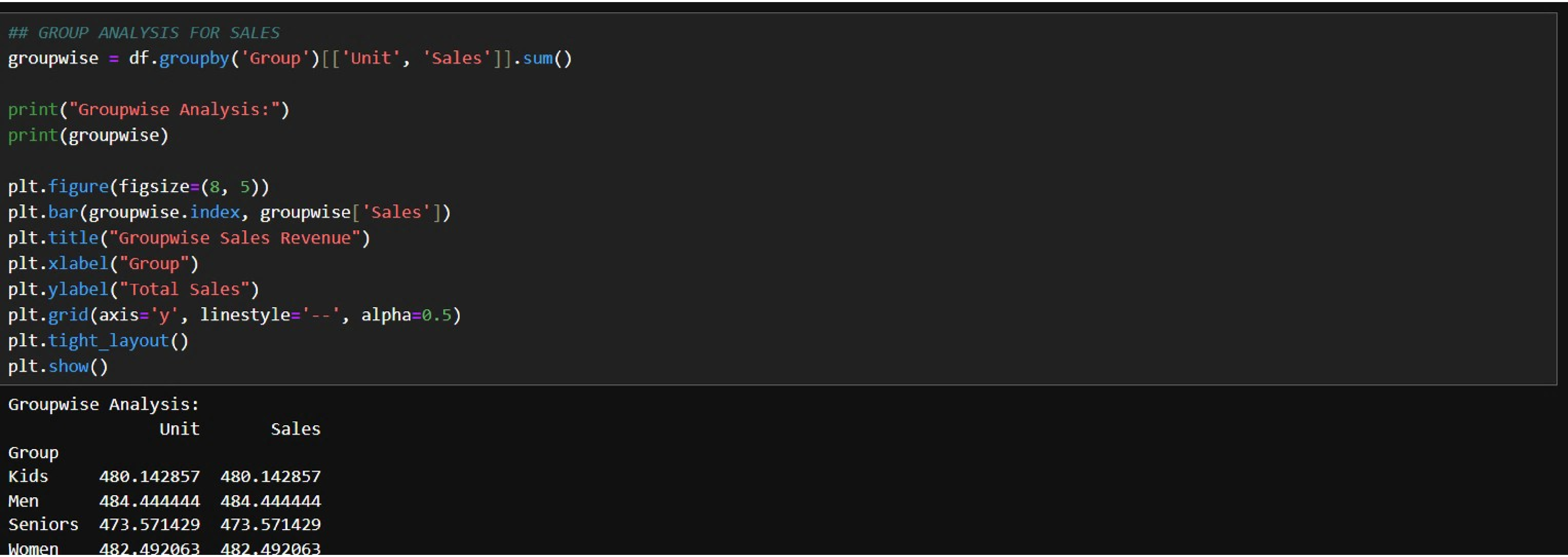

Statewise Sales in the United States
Total Sales
0
100
200
300
400
500
600
WA
AZ
CA
FL
KY
NY
TX



```python
# GROUP ANALYSIS
groupwise = df.groupby('Group')[['Unit', 'Sales']].sum()

print("Groupwise Analysis:")
print(groupwise)
## FOR UNITS
plt.figure(figsize=(8, 5))
plt.bar(groupwise.index, groupwise['Unit'])
plt.title("Groupwise Units Sold")
plt.xlabel("Group")
plt.ylabel("Total Units")
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

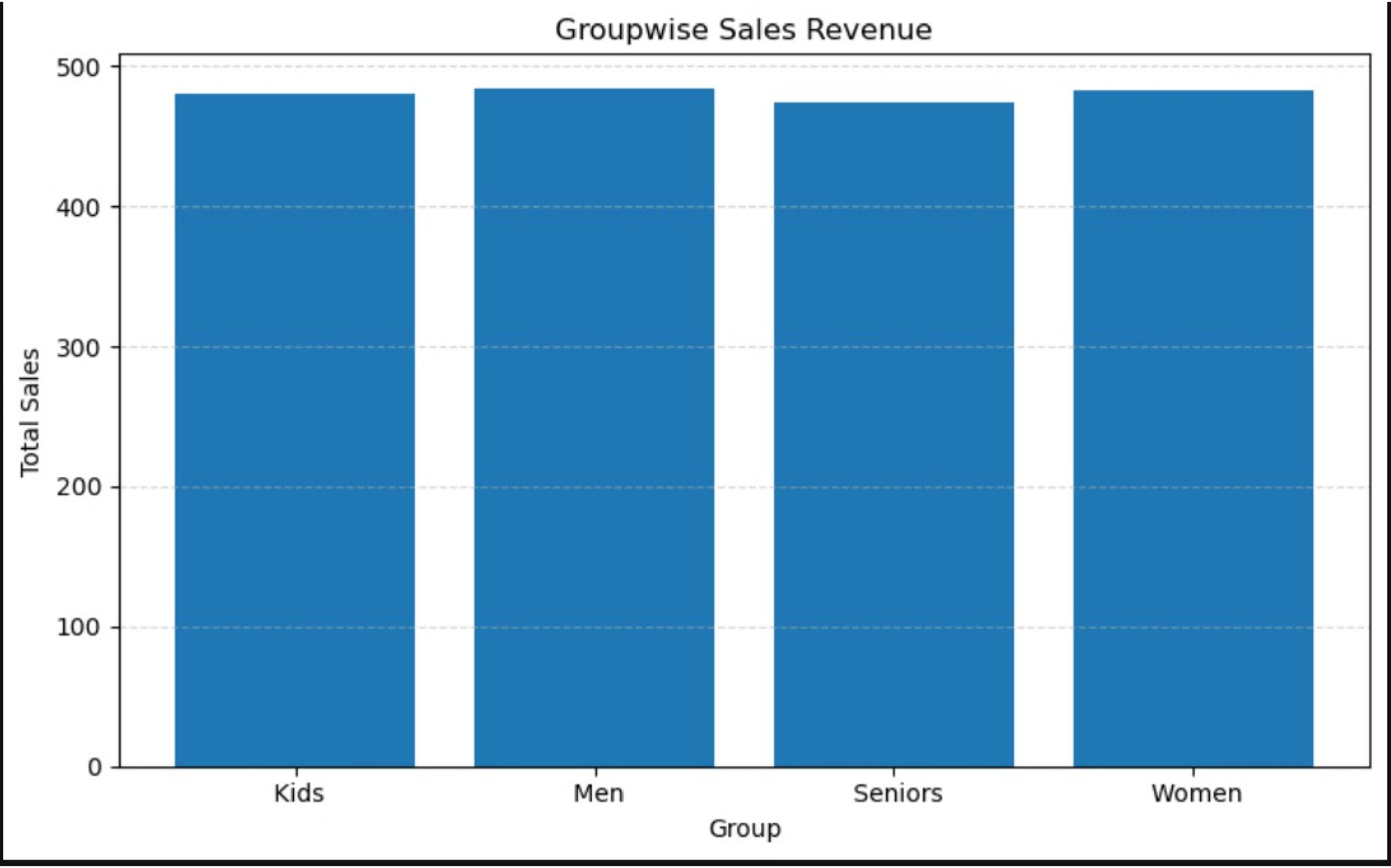
```
Groupwise Analysis:
               Unit       Sales
Group
Kids     480.142857  480.142857
Men      484.444444  484.444444
Seniors  473.571429  473.571429
Women    482.492063  482.492063
```

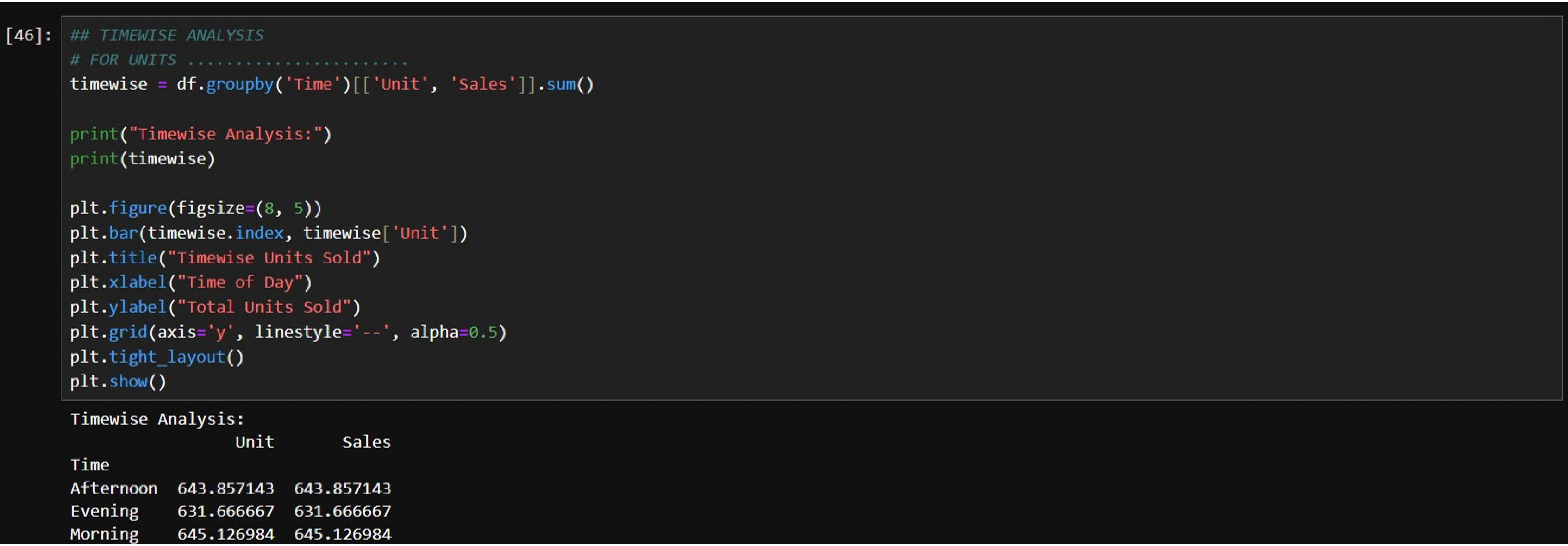

Groupwise Units Sold
Total Units
500
400
300
200
100
0
Kids
Men
Seniors
Women
Group



```python
## GROUP ANALYSIS FOR SALES
groupwise = df.groupby('Group')[['Unit', 'Sales']].sum()

print("Groupwise Analysis:")
print(groupwise)

plt.figure(figsize=(8, 5))
plt.bar(groupwise.index, groupwise['Sales'])
plt.title("Groupwise Sales Revenue")
plt.xlabel("Group")
plt.ylabel("Total Sales")
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

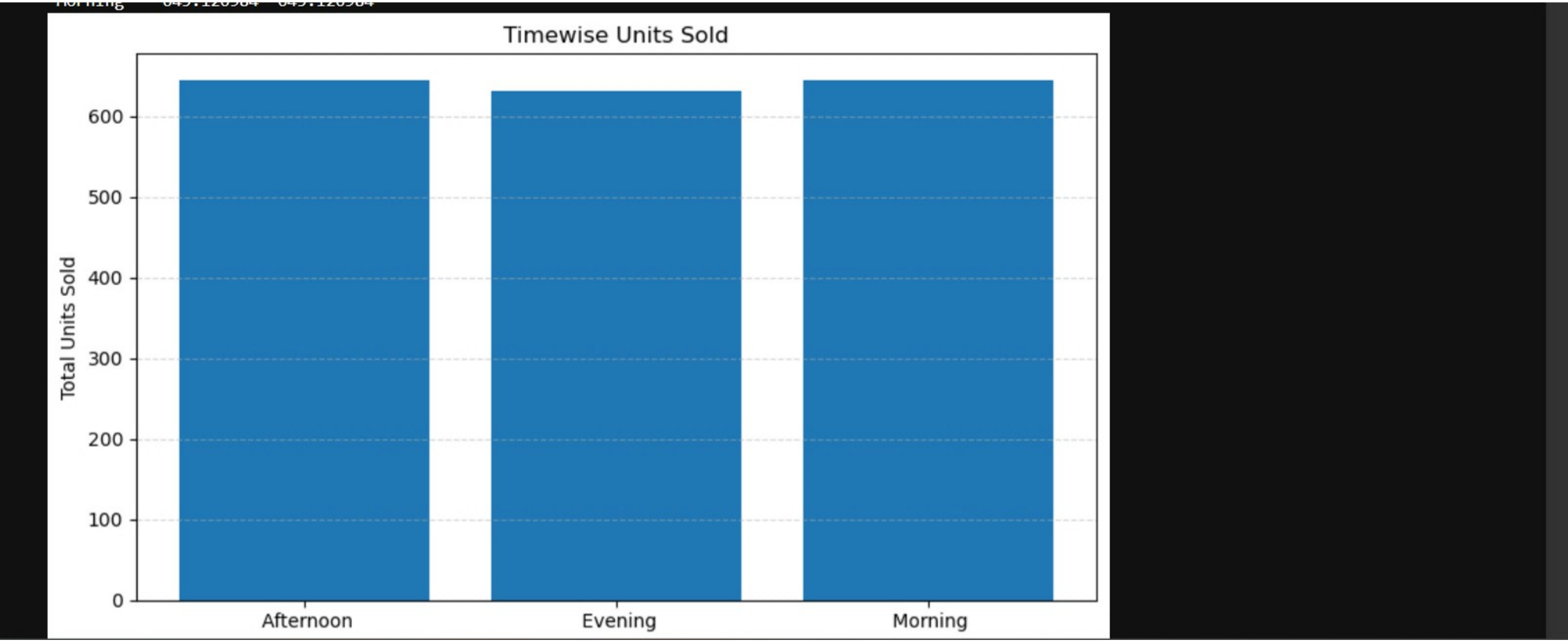
```
Groupwise Analysis:
               Unit       Sales
Group
Kids     480.142857  480.142857
Men      484.444444  484.444444
Seniors  473.571429  473.571429
Women    482.492063  482.492063
```


Groupwise Sales Revenue
500
400
300
200
100
0
Total Sales
Kids
Men
Seniors
Women
Group

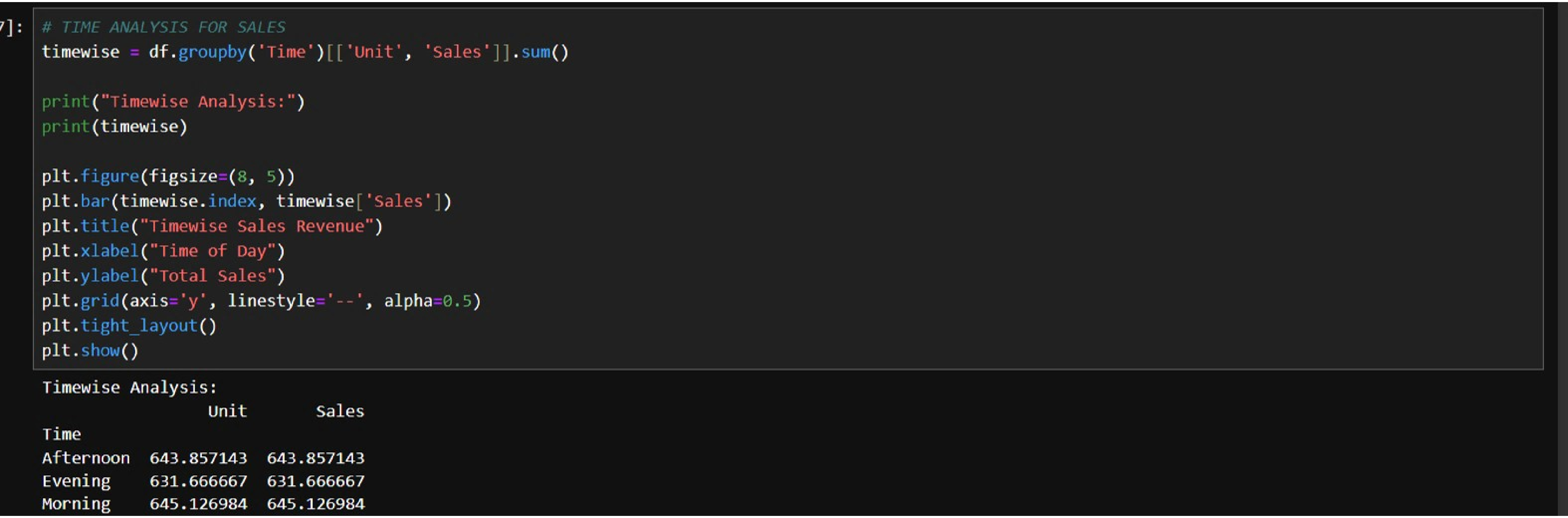


```
[46]: ## TIMEWISE ANALYSIS
      # FOR UNITS .......................
      timewise = df.groupby('Time')[['Unit', 'Sales']].sum()

      print("Timewise Analysis:")
      print(timewise)

      plt.figure(figsize=(8, 5))
      plt.bar(timewise.index, timewise['Unit'])
      plt.title("Timewise Units Sold")
      plt.xlabel("Time of Day")
      plt.ylabel("Total Units Sold")
      plt.grid(axis='y', linestyle='--', alpha=0.5)
      plt.tight_layout()
      plt.show()
```

```
Timewise Analysis:
                 Unit       Sales
Time
Afternoon  643.857143  643.857143
Evening    631.666667  631.666667
Morning    645.126984  645.126984
```



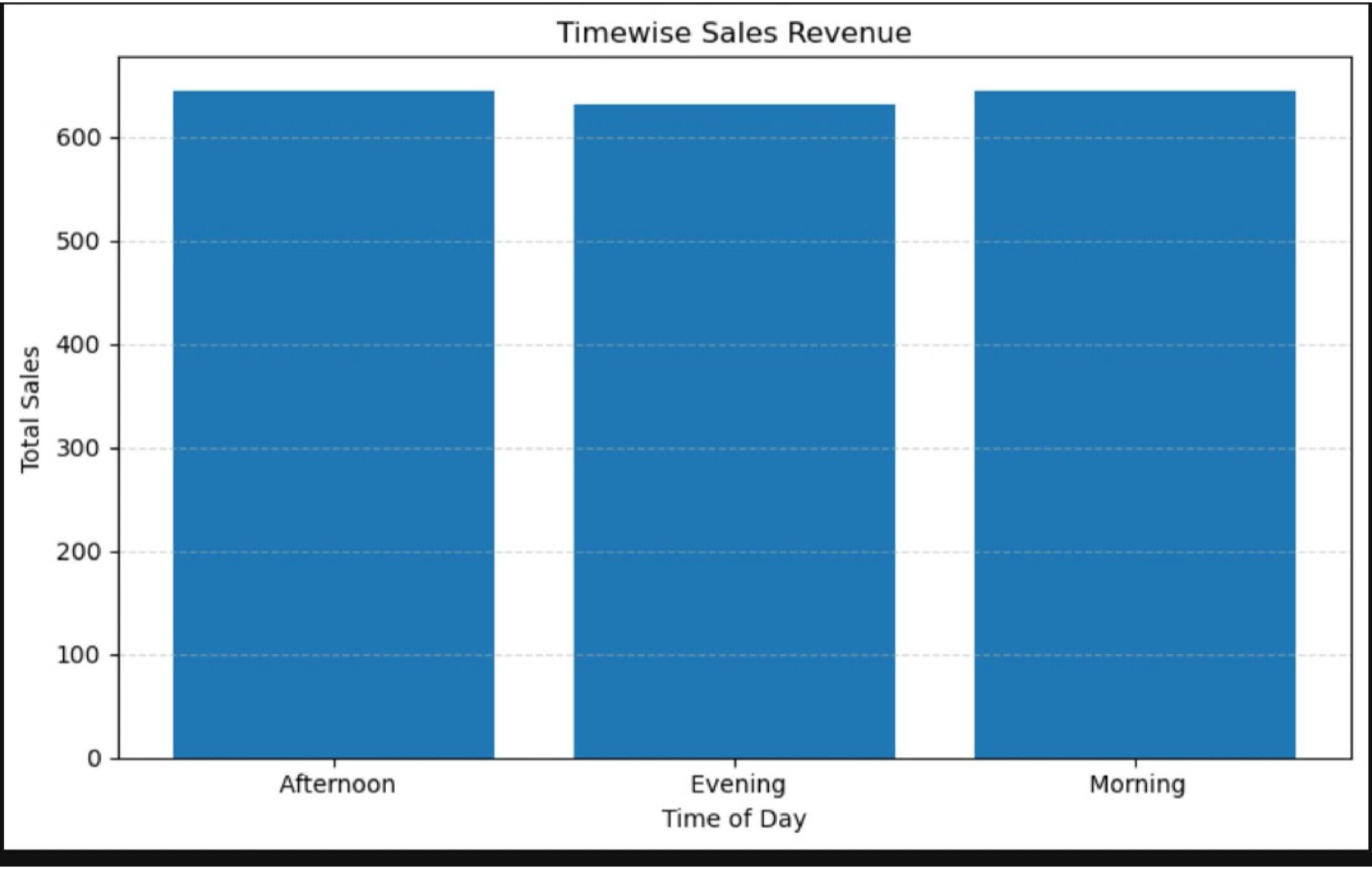
```
Morning      649.120984   649.120984
```



7]:

```
# TIME ANALYSIS FOR SALES
timewise = df.groupby('Time')[['Unit', 'Sales']].sum()

print("Timewise Analysis:")
print(timewise)

plt.figure(figsize=(8, 5))
plt.bar(timewise.index, timewise['Sales'])
plt.title("Timewise Sales Revenue")
plt.xlabel("Time of Day")
plt.ylabel("Total Sales")
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

```
Timewise Analysis:
                 Unit       Sales
Time
Afternoon  643.857143  643.857143
Evening    631.666667  631.666667
Morning    645.126984  645.126984
```


Timewise Sales Revenue
Total Sales
0
100
200
300
400
500
600
Afternoon
Evening
Morning
Time of Day