0kbfelgxo

April 3, 2024

[1]:
```
import numpy as np
import pandas as pd
import os
```

[6]:
```
# os.getcwd()
# os.listdir()
```

[2]:
```
df = pd.read_excel('C:\\POWER BI\\Walmart.xlsx')
df
```

[2]:

	Order ID	Order Date	Ship Date	Customer Name	Country	City	State	Category	Product Name	Sales	Quantity
0	CA-2013-138688	2013-06-13	2013-06-17	Darrin Van Huff	United States	Los Angeles	California	Labels			
1	CA-2011-115812	2011-06-09	2011-06-14	Brosina Hoffman	United States	Los Angeles	California	Furnishings			
2	CA-2011-115812	2011-06-09	2011-06-14	Brosina Hoffman	United States	Los Angeles	California	Art			
3	CA-2011-115812	2011-06-09	2011-06-14	Brosina Hoffman	United States	Los Angeles	California	Phones			
4	CA-2011-115812	2011-06-09	2011-06-14	Brosina Hoffman	United States	Los Angeles	California	Binders			
…	…	…	…	…	…	…	…	…			
3198	CA-2013-125794	2013-09-30	2013-10-04	Maris LaWare	United States	Los Angeles	California	Accessories			
3199	CA-2014-121258	2014-02-27	2014-03-04	Dave Brooks	United States	Costa Mesa	California	Furnishings			
3200	CA-2014-121258	2014-02-27	2014-03-04	Dave Brooks	United States	Costa Mesa	California	Phones			
3201	CA-2014-121258	2014-02-27	2014-03-04	Dave Brooks	United States	Costa Mesa	California	Paper			
3202	CA-2014-119914	2014-05-05	2014-05-10	Chris Cortes	United States	Westminster	California	Appliances			

```
0     Self-Adhesive Address Labels for Typewriters b…   14.620         2
1     Eldon Expressions Wood and Plastic Desk Access…   48.860         7
2                                          Newell 322    7.280         4
3                       Mitel 5320 IP Phone VoIP phone  907.152         4
4     DXL Angle-View Binders with Locking Rings by S…   18.504         3
…                                                    …      …       …
3198  Memorex Mini Travel Drive 64 GB USB 2.0 Flash …   36.240         1
3199  Tenex B1-RE Series Chair Mats for Low Pile Car…   91.960         2
3200                              Aastra 57i VoIP phone  258.576         2
3201  It's Hot Message Books with Stickers, 2 3/4" x 5"   29.600         4
3202  Acco 7-Outlet Masterpiece Power Center, Wihtou…  243.160         2

       Profit
0      6.8714
1     14.1694
2      1.9656
3     90.7152
4      5.7825
…       …
3198  15.2208
3199  15.6332
3200  19.3932
3201  13.3200
3202  72.9480

[3203 rows x 12 columns]
```

[3]:
```
df.head()
```

[3]:
```
          Order ID Order Date  Ship Date    Customer Name        Country  \
0  CA-2013-138688 2013-06-13 2013-06-17  Darrin Van Huff  United States
1  CA-2011-115812 2011-06-09 2011-06-14  Brosina Hoffman  United States
2  CA-2011-115812 2011-06-09 2011-06-14  Brosina Hoffman  United States
3  CA-2011-115812 2011-06-09 2011-06-14  Brosina Hoffman  United States
4  CA-2011-115812 2011-06-09 2011-06-14  Brosina Hoffman  United States

          City       State     Category  \
0  Los Angeles  California       Labels
1  Los Angeles  California  Furnishings
2  Los Angeles  California          Art
3  Los Angeles  California       Phones
4  Los Angeles  California      Binders

                                        Product Name    Sales  Quantity  \
0  Self-Adhesive Address Labels for Typewriters b…   14.620         2
1  Eldon Expressions Wood and Plastic Desk Access…   48.860         7
2                                     Newell 322    7.280         4
```

```
3                    Mitel 5320 IP Phone VoIP phone  907.152         4
4  DXL Angle-View Binders with Locking Rings by S…   18.504         3

    Profit
0   6.8714
1  14.1694
2   1.9656
3  90.7152
4   5.7825
```

[4]:
```
df.shape
```

[4]:
```
(3203, 12)
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3203 entries, 0 to 3202
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Order ID       3203 non-null   object
 1   Order Date     3203 non-null   datetime64[ns]
 2   Ship Date      3203 non-null   datetime64[ns]
 3   Customer Name  3203 non-null   object
 4   Country        3203 non-null   object
 5   City           3203 non-null   object
 6   State          3203 non-null   object
 7   Category       3203 non-null   object
 8   Product Name   3203 non-null   object
 9   Sales          3203 non-null   float64
 10  Quantity       3203 non-null   int64
 11  Profit         3203 non-null   float64
dtypes: datetime64[ns](2), float64(2), int64(1), object(7)
memory usage: 300.4+ KB
```

[5]:
```
df.describe()
```

[5]:
```
                          Order Date                     Ship Date  \
count                           3203                          3203
mean   2013-05-10 03:06:07.530440192  2013-05-14 01:25:25.195129600
min              2011-01-07 00:00:00            2011-01-09 00:00:00
25%              2012-05-22 00:00:00            2012-05-26 00:00:00
50%              2013-07-22 00:00:00            2013-07-25 00:00:00
75%              2014-05-23 00:00:00            2014-05-27 00:00:00
max              2014-12-31 00:00:00            2015-01-06 00:00:00
std                              NaN                           NaN
```

```
              Sales     Quantity       Profit
count   3203.000000  3203.000000  3203.000000
mean     226.493233     3.828910    33.849032
min        0.990000     1.000000 -3399.980000
25%       19.440000     2.000000     3.852000
50%       60.840000     3.000000    11.166400
75%      215.809000     5.000000    33.000400
max    13999.960000    14.000000  6719.980800
std      524.876877     2.260947   174.109081
```

[6]:
```
df.dtypes
```

[6]:
```
Order ID                 object
Order Date       datetime64[ns]
Ship Date        datetime64[ns]
Customer Name            object
Country                  object
City                     object
State                    object
Category                 object
Product Name             object
Sales                   float64
Quantity                  int64
Profit                  float64
dtype: object
```

[7]:
```
df.columns
# contains all the column names.
```

[7]:
```
Index(['Order ID', 'Order Date', 'Ship Date', 'Customer Name', 'Country',
       'City', 'State', 'Category', 'Product Name', 'Sales', 'Quantity',
       'Profit'],
      dtype='object')
```

[8]:
```
x = df.columns.tolist()
x = tuple(x)
print(x)
```

```
('Order ID', 'Order Date', 'Ship Date', 'Customer Name', 'Country', 'City',
'State', 'Category', 'Product Name', 'Sales', 'Quantity', 'Profit')
```

[10]:
```
# info about object dtype columns.
text_col = df.dtypes[df.dtypes=='object'].index
# print(text_col)
df[text_col].describe()
```

[10]:
```
              Order ID  Customer Name        Country         City       State  \
count             3203           3203           3203         3203        3203
unique            1611            686              1          169          11
top     CA-2013-165330  William Brown  United States  Los Angeles  California
freq                11             24           3203          747        2001

       Category Product Name
count      3203         3203
unique       17         1494
top     Binders       Staples
freq        471           60
```

[5]:
```
df.describe()
```

[5]:
```
                          Order Date                     Ship Date  \
count                           3203                          3203
mean   2013-05-10 03:06:07.530440192  2013-05-14 01:25:25.195129600
min              2011-01-07 00:00:00            2011-01-09 00:00:00
25%              2012-05-22 00:00:00            2012-05-26 00:00:00
50%              2013-07-22 00:00:00            2013-07-25 00:00:00
75%              2014-05-23 00:00:00            2014-05-27 00:00:00
max              2014-12-31 00:00:00            2015-01-06 00:00:00
std                              NaN                            NaN

               Sales     Quantity       Profit
count   3203.000000  3203.000000  3203.000000
mean     226.493233     3.828910    33.849032
min        0.990000     1.000000 -3399.980000
25%       19.440000     2.000000     3.852000
50%       60.840000     3.000000    11.166400
75%      215.809000     5.000000    33.000400
max    13999.960000    14.000000  6719.980800
std      524.876877     2.260947   174.109081
```

1 Basic Information:

1. What is the overall sales figure for the given dataset?

[11]:
```
print("Overall sales figure of the dataset is ",df['Sales'].sum())
# df['Sales'].sum()
```

```
Overall sales figure of the dataset is  725457.8245
```

2.How many unique customers are there in the dataset?

[12]:
```
unique_cust = df['Customer Name'].unique()
print('Numbers of Unique customer is ',len(unique_cust))
```

```
Numbers of Unique customer is  686
```

[13]:
```
# way 2
unique_cust1=  df['Customer Name'].describe()
# unique_cust1['unique']
print('Number of unique customer is ',unique_cust1['unique'])
```

```
Number of unique customer is  686
```

2 Time-based Analysis:

2.1 1.What is the average time taken to ship orders?

[5]:
```
time_taken = df['Ship Date'] - df['Order Date']
print(time_taken)
```

```
0      4 days
1      5 days
2      5 days
3      5 days
4      5 days
        …
3198   4 days
3199   5 days
3200   5 days
3201   5 days
3202   5 days
Length: 3203, dtype: timedelta64[ns]
```

[55]:
```
avg = np.mean(time_taken)
print(avg)
```

```
3 days 22:19:17.664689353
```

[14]:
```
# this is the actually method.
# way 2
df['Ship Date'] = pd.to_datetime(df['Ship Date']) #to convert it to a valid␣
 ↪date and time.
df['Order Date'] = pd.to_datetime(df['Order Date'])

time_taken = (df['Ship Date'] - df['Order Date']).dt.days  #to only extract the␣
 ↪integer part
print(time_taken)


# df['Order Date']
```

```
0       4
```

```
1       5
2       5
3       5
4       5
       ..
3198    4
3199    5
3200    5
3201    5
3202    5
Length: 3203, dtype: int64
```

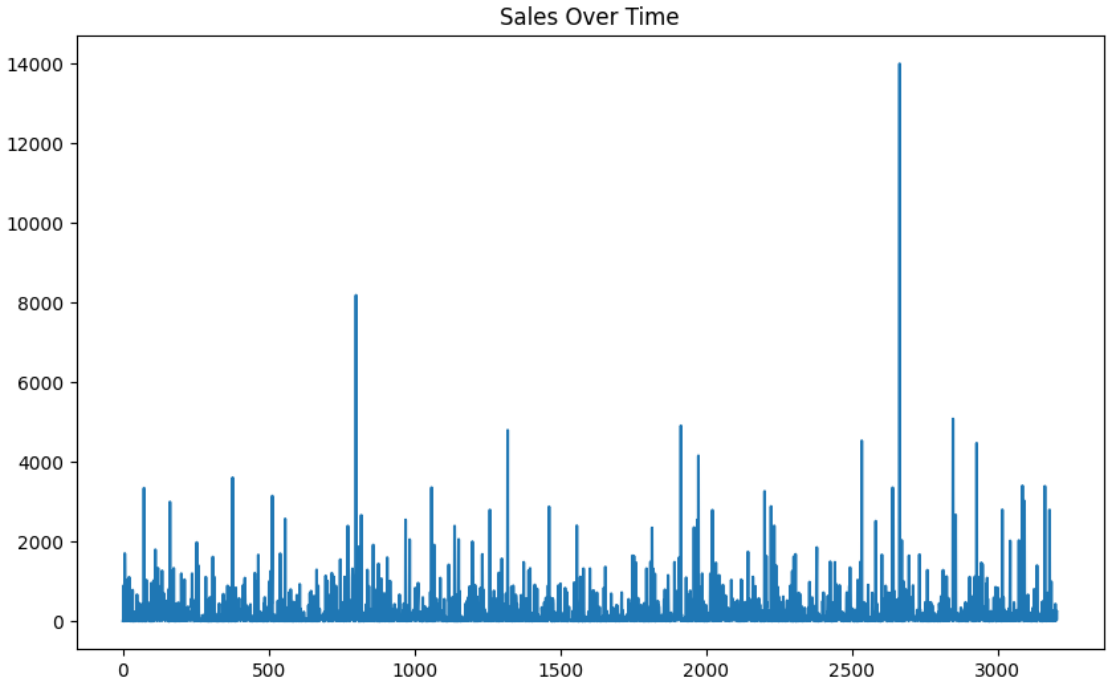
[15]:
```
avg = np.mean(time_taken)
# print(avg)
avg_round = np.round(avg)  #to round over.

print('Average time taken to ship order is ',avg_round,' days')
```

```
Average time taken to ship order is  4.0  days
```

2.2 2.Are there any trends or patterns in sales over time?

[14]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'])
df.set_index('Order Date',inplace=True)
```

[3]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
plt.plot(df['Sales'])
plt.title('Sales Over Time')
plt.show()
```



2.3 3.Which month has the highest sales?

[18]:
```
df['Month'] = df['Order Date'].dt.month
df['Month']
```

[18]:
```
0       6
1       6
2       6
3       6
4       6
       ..
3198    9
3199    2
3200    2
3201    2
3202    5
Name: Month, Length: 3203, dtype: int32
```

[19]:
```
monthly_sales = df.groupby('Month')['Sales'].sum()
print(monthly_sales)
# type(monthly_sales)

print(sorted(monthly_sales,reverse = True))
# sorted(monthly_sales,reverse = True)
```

```
Month
1      24918.6190
2      16268.6150
3      73023.7390
4      41966.5850
5      45320.8745
6      48519.5455
7      64706.8940
8      62674.3280
9      81618.4605
10     57436.7980
11     93068.8225
12    115934.5435
Name: Sales, dtype: float64
[115934.5435, 93068.8225, 81618.4605, 73023.739, 64706.894, 62674.328,
57436.798, 48519.5455, 45320.8745, 41966.585, 24918.619, 16268.615]
```

[20]:
```
highest_sales_month = monthly_sales.idxmax()
# print(highest_sales_month)
print(monthly_sales.idxmax())
```

```
12
```

[21]:
```
# it reverse a pandas series
monthly_sales.iloc[::-1]
```

[21]:
```
Month
12    115934.5435
11     93068.8225
10     57436.7980
9      81618.4605
8      62674.3280
7      64706.8940
6      48519.5455
5      45320.8745
4      41966.5850
3      73023.7390
2      16268.6150
1      24918.6190
Name: Sales, dtype: float64
```

[22]:
```
# suppose i want to show the highest sale month as well as the amount.
max_sale =  monthly_sales.max()
max_sale = np.round(max_sale,2)
print("Highest sale month is ",highest_sales_month," and the sale amount␣
 ↪is",max_sale)
```

```
Highest sale month is  12  and the sale amount is 115934.54
```

3 Customer Analysis:

3.1 1.Who are the top 5 customers based on total sales?

[23]:
```
top5cust =  df.groupby(df['Customer Name'])['Sales'].sum()
print(top5cust)
```

```
Customer Name
Aaron Bergman          309.592
Aaron Hawkins         1328.124
Aaron Smayling         737.028
Adam Bellavance       2693.918
Adam Hart              463.770
                         ...
Xylona Preis           660.190
Yana Sorensen         5754.172
Yoseph Carroll        1215.676
Zuschuss Carroll      2641.089
Zuschuss Donatelli     306.200
Name: Sales, Length: 686, dtype: float64
```

[24]:
```
# here we have learn how to solve.
# var.sort_values() will sort by values and
# var.sort_index() will sort by indexes

sort_top5cust = top5cust.sort_values(ascending=False)
# sort_top5cust = top5cust.sort_index()
sort_top5cust[:5]
```

[24]:
```
Customer Name
Raymond Buch      14345.276
Ken Lonsdale       8472.394
Edward Hooks       7447.770
Jane Waco          7391.530
Karen Ferguson     7182.766
Name: Sales, dtype: float64
```

[25]:
```
# or we can achieve this with the help of loop
count =0
for Customer_Name,Sales in sort_top5cust.items():
    print(f'{Customer_Name} : {Sales}')  #this is f-string, new to python.
#  print(Customer_Name,Sales)
#  print(Sales)
    count = count+1
    if count ==5:
     break
```

```
Raymond Buch : 14345.276
```

```
Ken Lonsdale : 8472.394
Edward Hooks : 7447.77
Jane Waco : 7391.530000000001
Karen Ferguson : 7182.766
```

3.2 2.What is the average quantity of products purchased per customer?

[26]:
```
avgQuant = df.groupby(df['Customer Name'])['Quantity'].mean()
# print(avgQuant)
round_avgQuant = np.round(avgQuant)
round_avgQuant
```

[26]:
```
Customer Name
Aaron Bergman         2.0
Aaron Hawkins         5.0
Aaron Smayling        4.0
Adam Bellavance       3.0
Adam Hart             3.0
                     …
Xylona Preis          4.0
Yana Sorensen         5.0
Yoseph Carroll        4.0
Zuschuss Carroll      3.0
Zuschuss Donatelli    3.0
Name: Quantity, Length: 686, dtype: float64
```

4 Geographical Analysis

4.1 1.What are the top 3 cities with the highest sales?

[27]:
```
high_sale_city = df.groupby('City')['Sales'].sum()
sort_high_sale_city = high_sale_city.sort_values(ascending=False)
high_city =  np.round(sort_high_sale_city,2)
high_city
```

[27]:
```
City
Los Angeles        175851.34
Seattle            119540.74
San Francisco      112669.09
San Diego           47521.03
Denver              12198.79
                       …
Billings                8.29
Layton                  4.96
Auburn                  4.18
Everett                 3.86
San Luis Obispo         3.62
```

```
Name: Sales, Length: 169, dtype: float64
```

[30]:
```
# finding the top 3
count = 0
for City,Sales in high_city.items():
    print(f' {City} : {Sales}')
    count = count + 1
    if count == 3:
        break
```

```
Los Angeles : 175851.34
Seattle : 119540.74
San Francisco : 112669.09
```

4.2 2.How does the sales distribution vary across different city?

[31]:
```
# top 10 cities
top10_high_city = high_city[:10]
top10_high_city
```

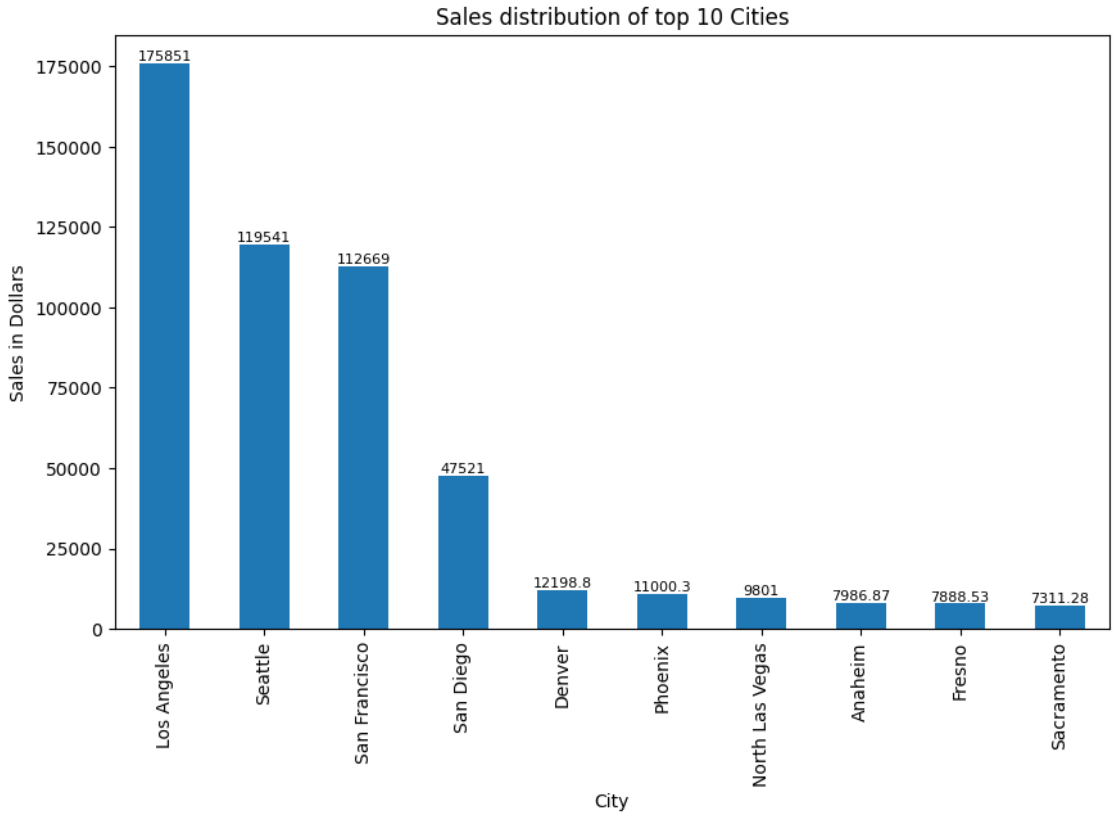
[31]:
```
City
Los Angeles        175851.34
Seattle            119540.74
San Francisco      112669.09
San Diego           47521.03
Denver              12198.79
Phoenix             11000.26
North Las Vegas      9801.00
Anaheim              7986.87
Fresno               7888.53
Sacramento           7311.28
Name: Sales, dtype: float64
```

[32]:
```
# least 10 cities
least10_high_city = high_city[::-1][:10]
least10_high_city
```

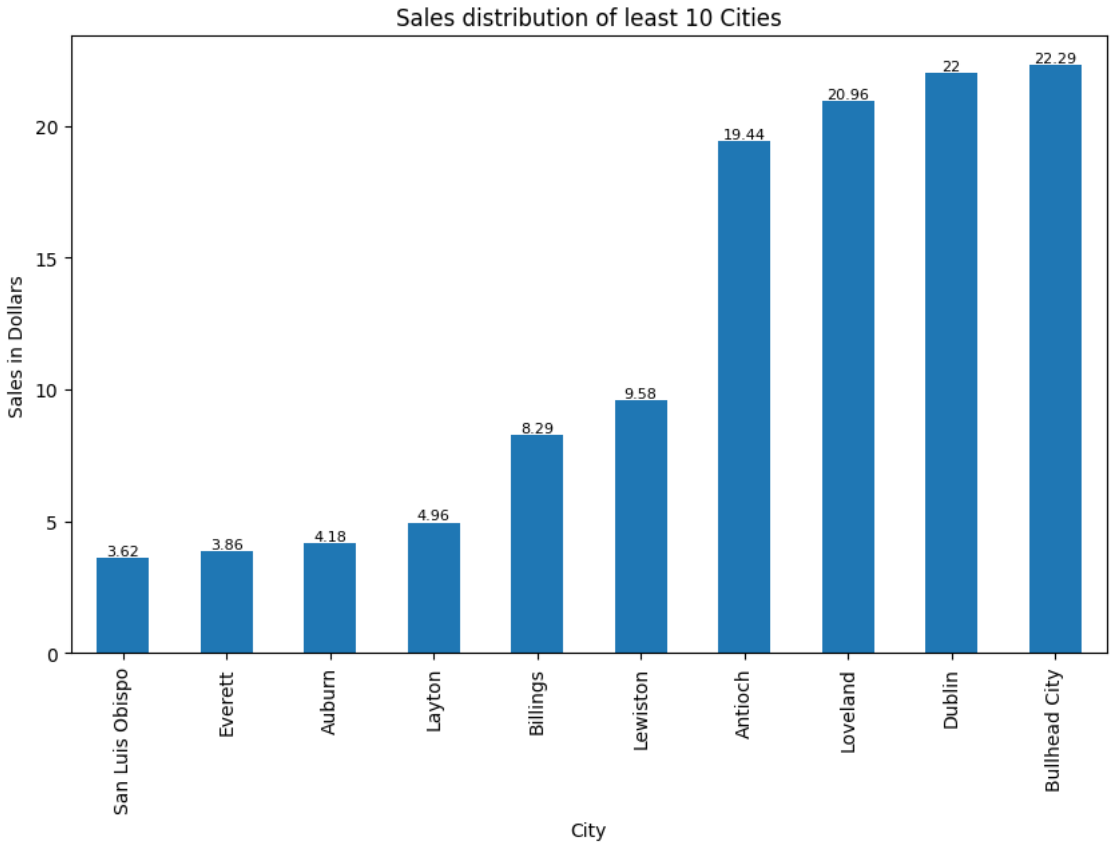
[32]:
```
City
San Luis Obispo     3.62
Everett             3.86
Auburn              4.18
Layton              4.96
Billings            8.29
Lewiston            9.58
Antioch            19.44
Loveland           20.96
Dublin             22.00
```

```
Bullhead City      22.29
Name: Sales, dtype: float64
```

[34]:
```
import matplotlib.pyplot as plt

top10_high_city.plot(kind = 'bar', figsize=(10,6))
plt.title("Sales distribution of top 10 Cities")
plt.xlabel("City")
plt.ylabel("Sales in Dollars")
plt.bar_label(plt.gca().containers[0],fontsize=8)
plt.show()
```



[36]:
```
least10_high_city.plot(kind = 'bar', figsize=(10,6))
plt.title("Sales distribution of least 10 Cities")
plt.xlabel("City")
plt.ylabel("Sales in Dollars")
plt.bar_label(plt.gca().containers[0],fontsize=8)
plt.show()
```

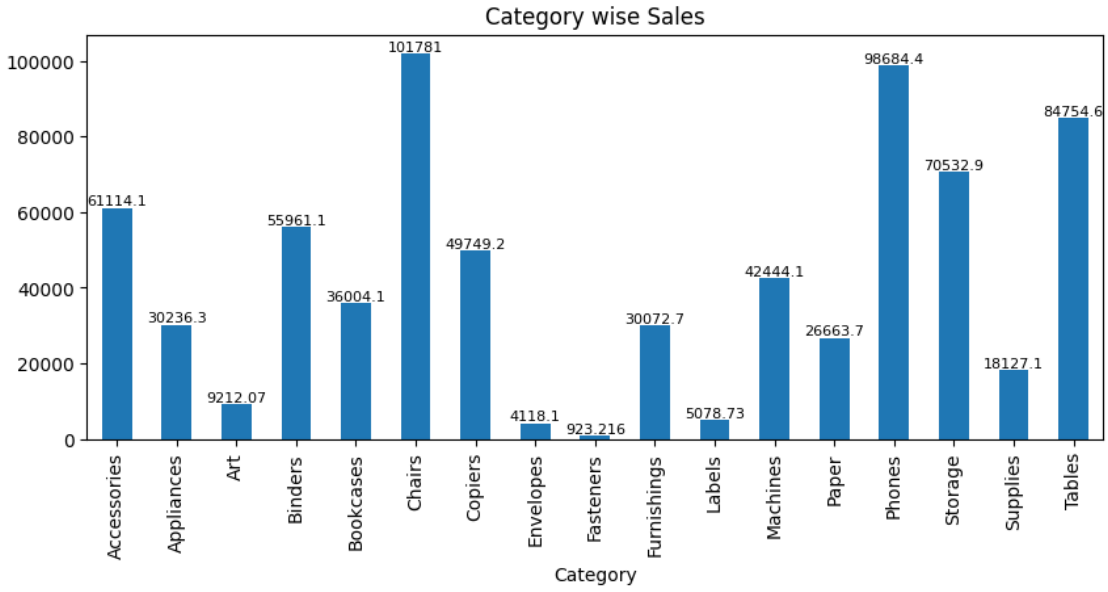


4.3 3.Is there any correlation between the category and the products sold?

[37]:
```
uni = df['Category'].unique()
uni.size
```

[37]: 17

[60]:
```
# but this is not what the question ask for. JUST FOR PRACTICE.
correlation = df.groupby('Category')['Sales'].sum()
correlation.plot(kind = 'bar', figsize=(10,4))
plt.bar_label(plt.gca().containers[0],fontsize=8)
plt.title("Category wise Sales")
plt.show()
```



To convert a categorical variable like 'Product Category' into a numerical format suitable for correlation analysis, you can use one-hot encoding or label encoding. Both methods have their use cases, but for correlation analysis, label encoding is more straightforward and commonly used.

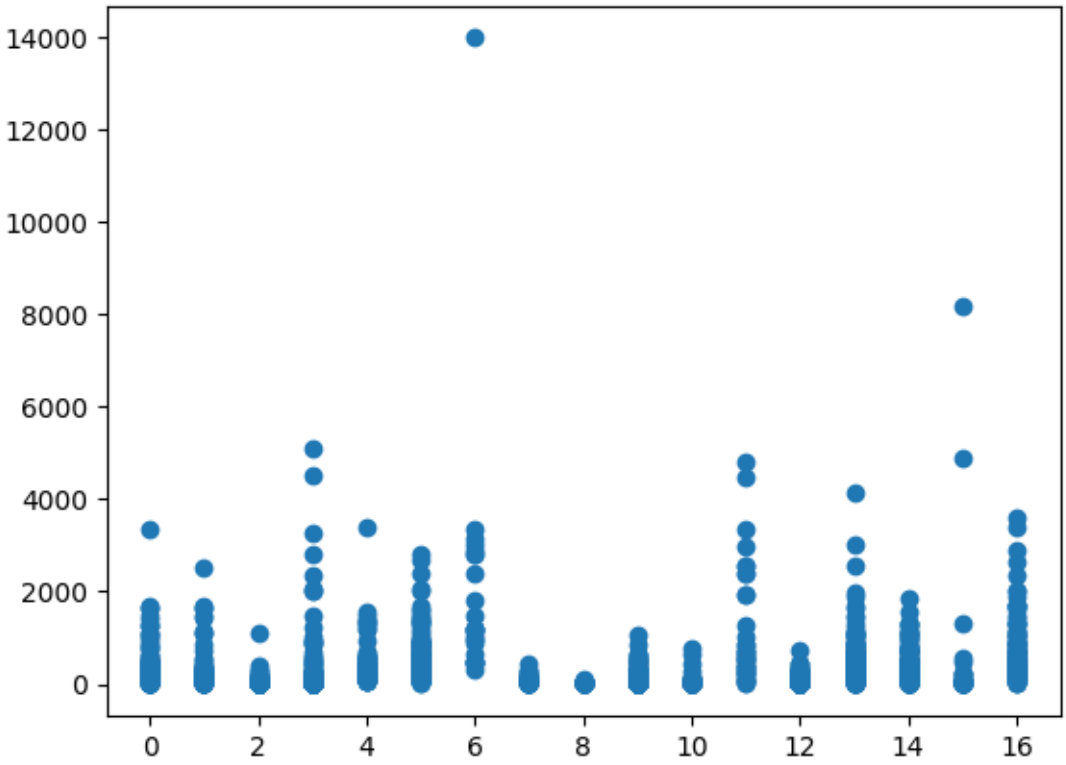
Label Encoding - Label encoding assigns each unique category in the 'Product Category' column a unique integer. This is useful for ordinal variables where the order matters. However, it's important to note that label encoding can introduce a false sense of order if the categories are not ordinal.

[40]:
```
#import LabelEncoder
from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder
le = LabelEncoder()

# Apply label encoding to the 'Product Category' column
df['Product Category Encoded'] = le.fit_transform(df['Category'])

correlation = df['Product Category Encoded'].corr(df['Sales'])

correlation
```

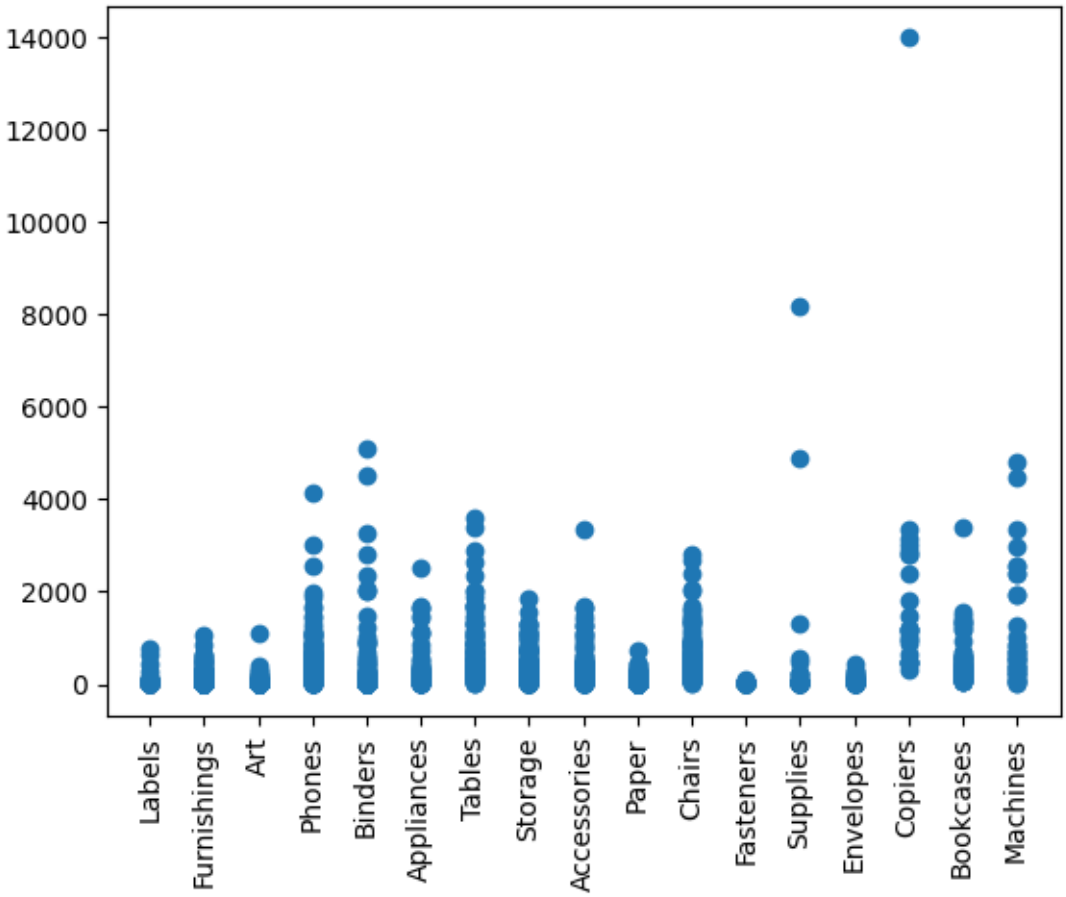
[40]:
```
0.07649638020563898
```

[61]:
```
plt.scatter(df['Product Category Encoded'],df['Sales'])
```

[61]:
```
<matplotlib.collections.PathCollection at 0x2a5d6124110>
```



[44]:
```
plt.scatter(df['Category'],df['Sales'])
plt.xticks(rotation=90)
plt.show()
```



5 Product Analysis:

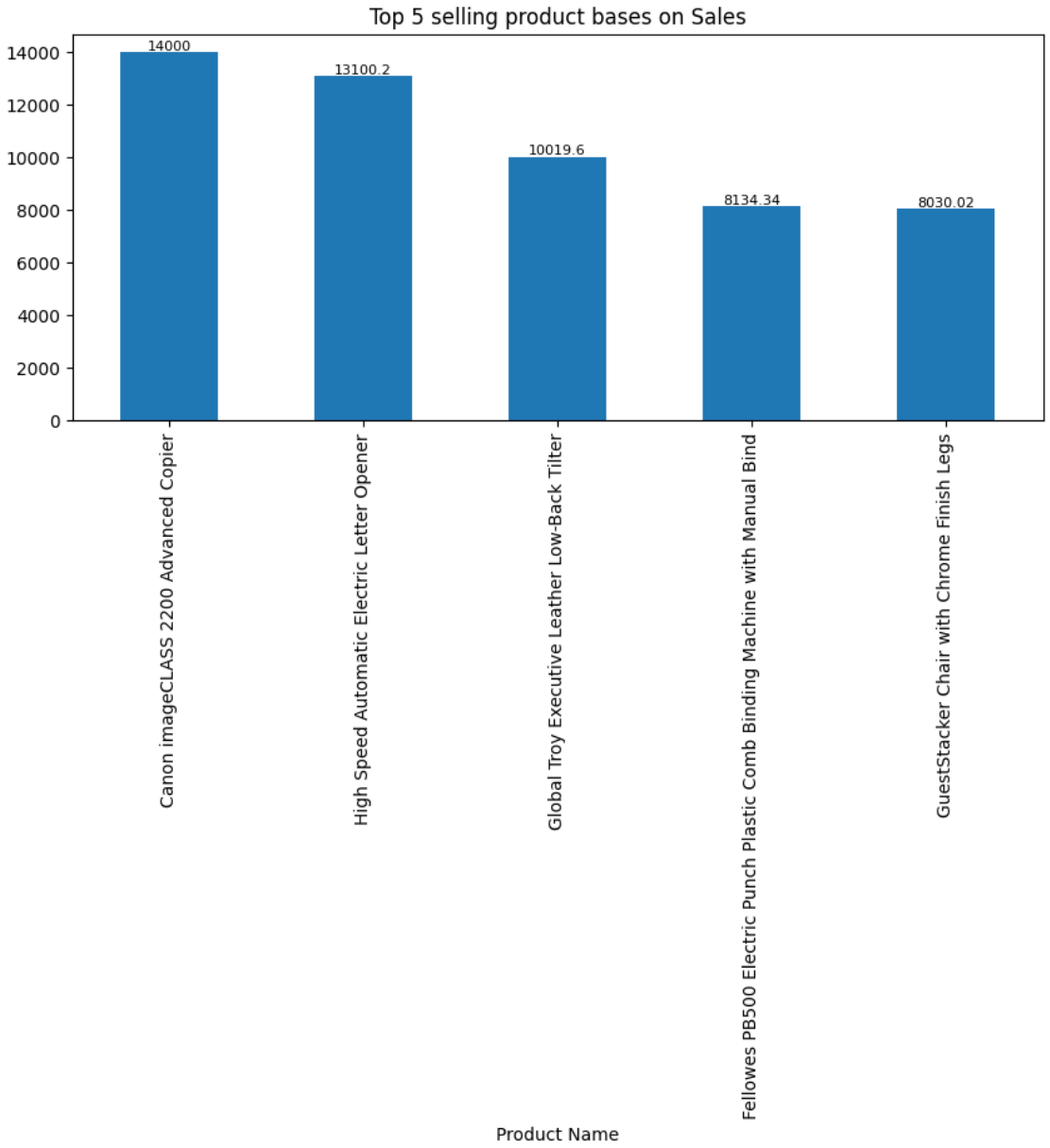
5.1 1.What are the top 5 selling products based on quantity and sales?

[62]:
```
# Top 5 selling product bases on Sales
top5_prod_sale = df.groupby('Product Name')['Sales'].sum()
# top5_prod_sale[::-1][:5]
top5_prod_sale = top5_prod_sale.sort_values(ascending=False)
top5_prod_sale = top5_prod_sale[:5]
top5_prod_sale
# type(top5_prod_sale)
```

[62]:
```
Product Name
Canon imageCLASS 2200 Advanced Copier
13999.960
High Speed Automatic Electric Letter Opener
13100.240
Global Troy Executive Leather Low-Back Tilter
```

```
10019.600
Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manual Bind
8134.336
GuestStacker Chair with Chrome Finish Legs
8030.016
Name: Sales, dtype: float64
```

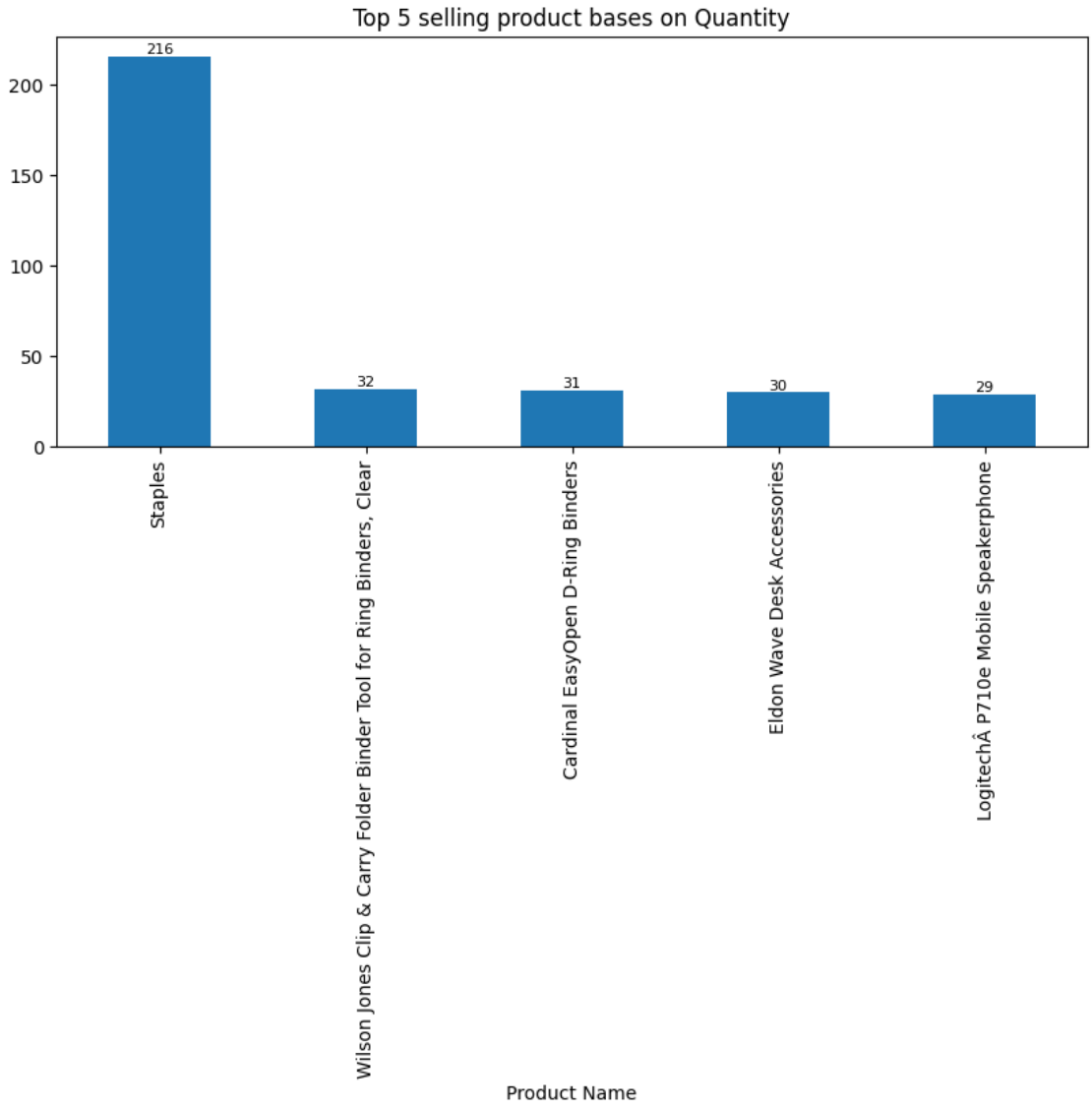
[63]:
```
top5_prod_sale.plot(kind = 'bar', figsize =(10,4))
plt.bar_label(plt.gca().containers[0],fontsize=8)
plt.title("Top 5 selling product bases on Sales")
plt.show()
```



[54]:
```
# Top 5 selling product bases on Quantity
top5_prod_sale_qunty = df.groupby('Product Name')['Quantity'].sum()
top5_prod_sale_qunty = top5_prod_sale_qunty.sort_values(ascending=False)
top5_prod_sale_qunty = top5_prod_sale_qunty[:5]
top5_prod_sale_qunty
```

[54]:
```
Product Name
Staples                                                                 216
Wilson Jones Clip & Carry Folder Binder Tool for Ring Binders, Clear     32
Cardinal EasyOpen D-Ring Binders                                         31
Eldon Wave Desk Accessories                                              30
LogitechÂ P710e Mobile Speakerphone                                      29
Name: Quantity, dtype: int64
```

[64]:
```
top5_prod_sale_qunty.plot(kind = 'bar', figsize = (10,4))
plt.bar_label(plt.gca().containers[0],fontsize=8)
plt.title("Top 5 selling product bases on Quantity")
plt.show()
```



5.2 2.Which product has the highest profit?

[56]:
```
# top 5 profit generated products
top5_porf_prod = df.groupby('Product Name')['Profit'].sum()
top5_porf_prod = top5_porf_prod.sort_values(ascending=False)
top5_porf_prod[:5]
```

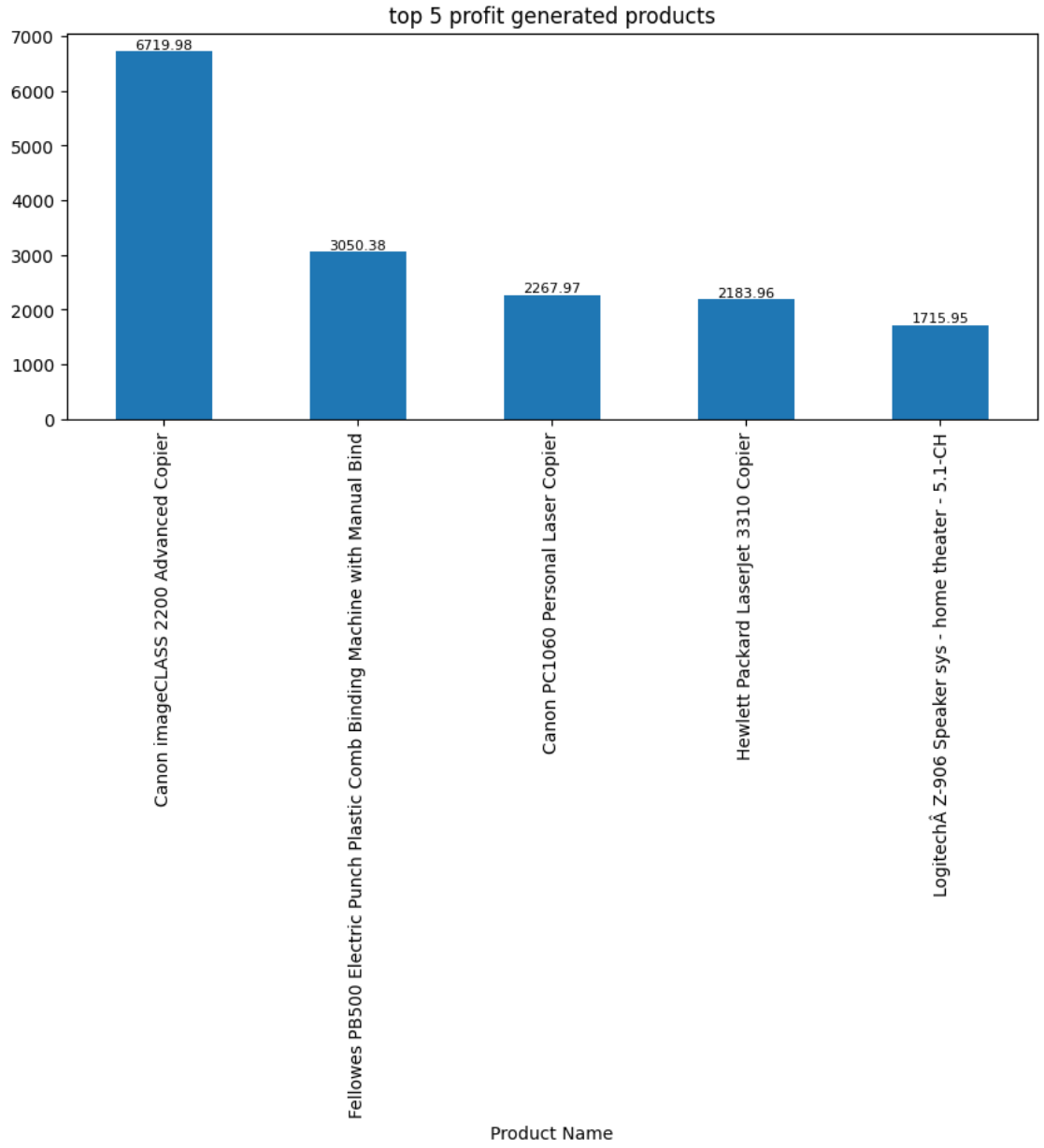
[56]:
```
Product Name
Canon imageCLASS 2200 Advanced Copier
6719.9808
Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manual Bind
3050.3760
```

```
Canon PC1060 Personal Laser Copier
2267.9676
Hewlett Packard LaserJet 3310 Copier
2183.9636
LogitechÂ Z-906 Speaker sys - home theater - 5.1-CH
1715.9480
Name: Profit, dtype: float64
```

[57]:
```
print(top5_porf_prod.idxmax(),top5_porf_prod.max())
```

```
Canon imageCLASS 2200 Advanced Copier 6719.9808
```

[65]:
```
top5_porf_prod = top5_porf_prod[:5]
top5_porf_prod.plot(kind = 'bar', figsize = (10,4))
plt.bar_label(plt.gca().containers[0],fontsize=8)
plt.title("top 5 profit generated products")
plt.show()
```



6 Category-wise Analysis:

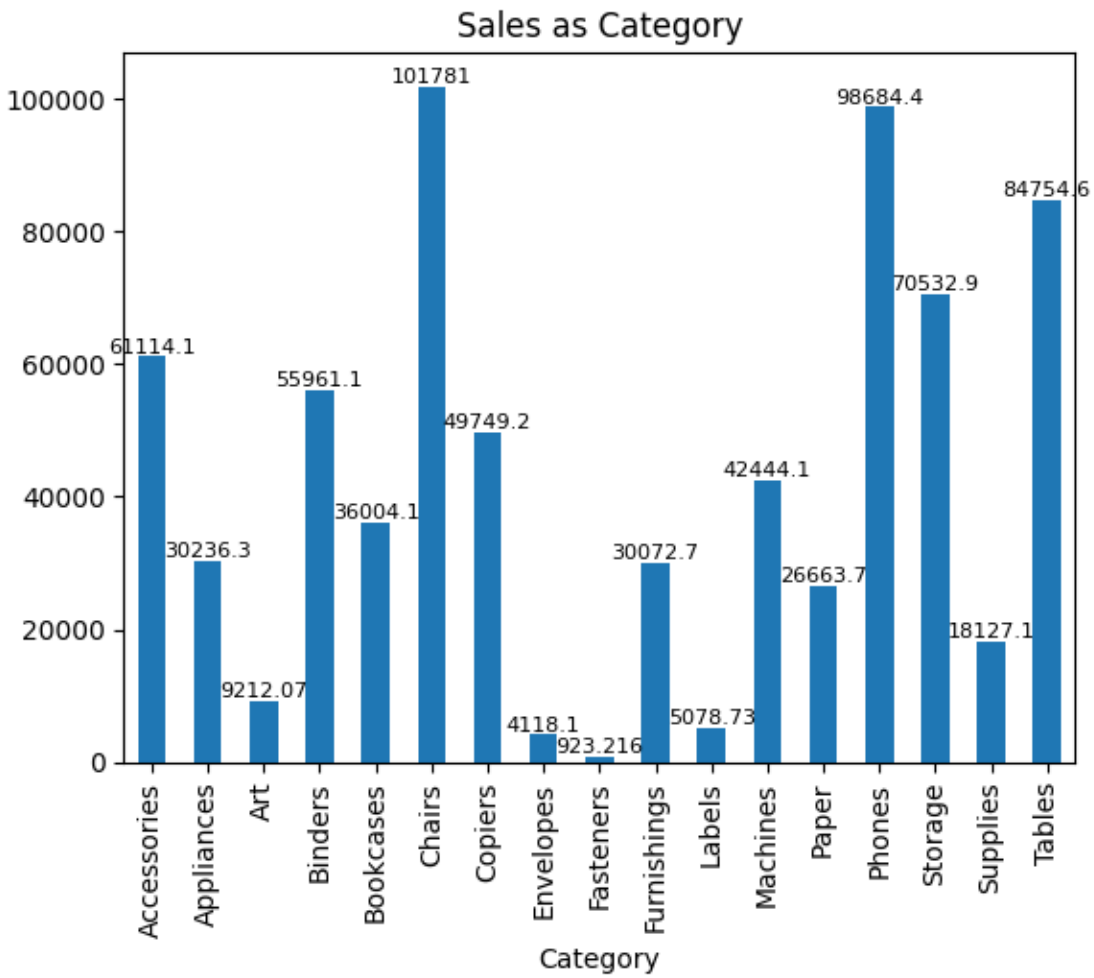
6.1 1.How do sales and profits vary across different product categories?

[67]:
```
most_cate_sold = df.groupby('Category')['Sales'].sum()
most_cate_sold
```

[67]:
```
Category
Accessories     61114.1160
```

```
Appliances      30236.3360
Art              9212.0660
Binders         55961.1130
Bookcases       36004.1235
Chairs         101781.3280
Copiers         49749.2420
Envelopes        4118.1000
Fasteners         923.2160
Furnishings     30072.7300
Labels           5078.7260
Machines        42444.1220
Paper           26663.7180
Phones          98684.3520
Storage         70532.8520
Supplies        18127.1220
Tables          84754.5620
Name: Sales, dtype: float64
```

[70]:
```
most_cate_sold.plot(kind = 'bar')
plt.bar_label(plt.gca().containers[0],fontsize=8)
plt.title("Sales as Category")
plt.show()
```

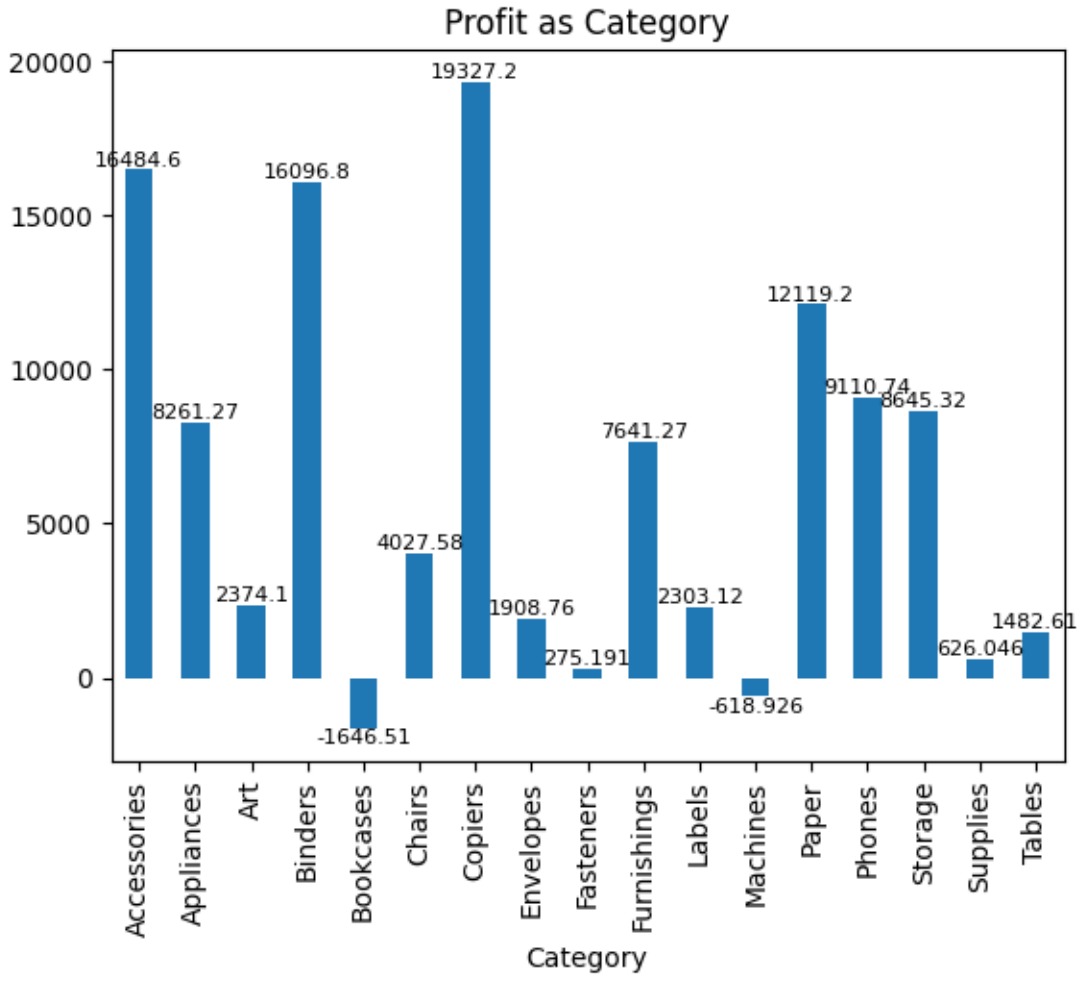


[74]:
```
most_cate_prof = df.groupby('Category')['Profit'].sum()
most_cate_prof
```

[74]:
```
Category
Accessories    16484.5983
Appliances      8261.2699
Art             2374.0970
Binders        16096.8016
Bookcases      -1646.5117
Chairs          4027.5843
Copiers        19327.2351
Envelopes       1908.7624
Fasteners        275.1907
Furnishings     7641.2704
Labels          2303.1223
Machines        -618.9264
```

```
Paper          12119.2364
Phones          9110.7426
Storage         8645.3222
Supplies         626.0465
Tables          1482.6073
Name: Profit, dtype: float64
```

[75]:
```
import matplotlib.pyplot as plt
most_cate_prof.plot(kind = 'bar')
plt.bar_label(plt.gca().containers[0], fontsize=8)
plt.title("Profit as Category")
plt.show()
```

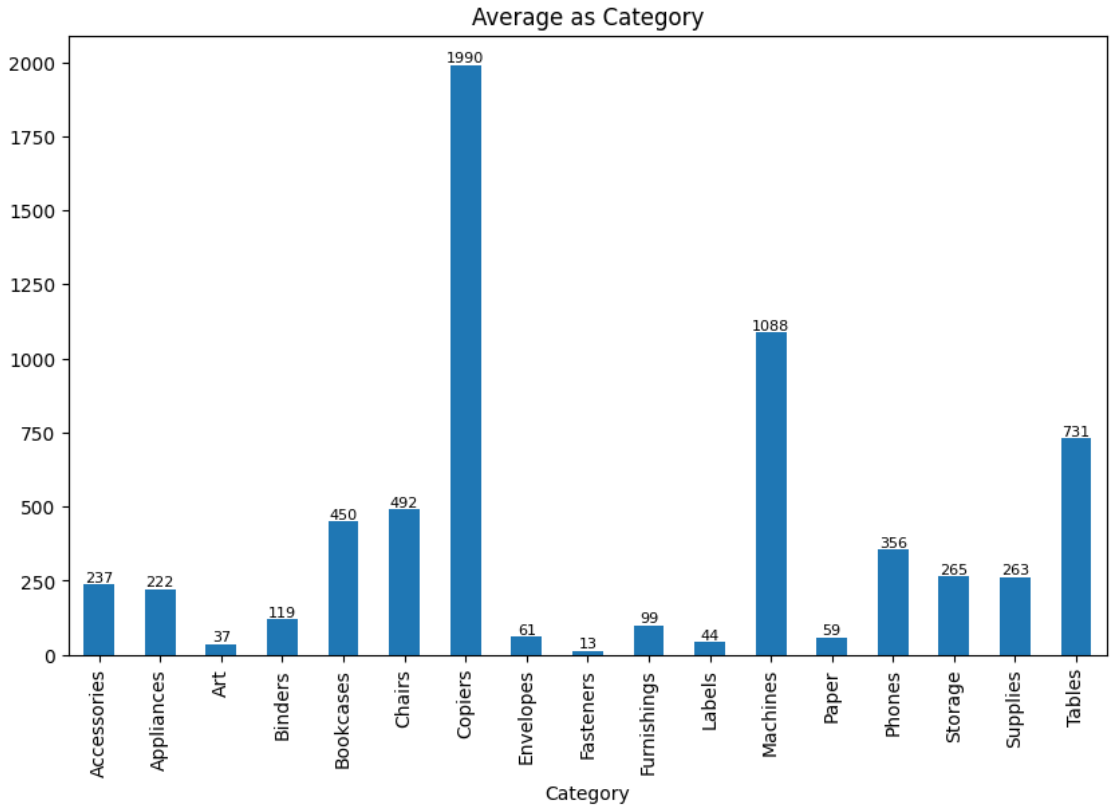


6.2 2. Which category has the highest average sales?

[77]:
```
high_cate_avg = df.groupby('Category')['Sales'].mean()
high_cate_avg = np.round(high_cate_avg)
high_cate_avg
```

[77]:
```
Category
Accessories     237.0
Appliances      222.0
Art              37.0
Binders         119.0
Bookcases       450.0
Chairs          492.0
Copiers        1990.0
Envelopes        61.0
Fasteners        13.0
Furnishings      99.0
Labels           44.0
Machines       1088.0
Paper            59.0
Phones          356.0
Storage         265.0
Supplies        263.0
Tables          731.0
Name: Sales, dtype: float64
```

[80]:
```
high_cate_avg.plot(kind='bar', figsize=(10,6))
plt.bar_label(plt.gca().containers[0], fontsize=8)
plt.title("Average as Category")
plt.show()
```


Average as Category
2000
1750
1500
1250
1000
750
500
250
0
237
222
37
119
450
492
1990
61
13
99
44
1088
59
356
265
263
731
Accessories
Appliances
Art
Binders
Bookcases
Chairs
Copiers
Envelopes
Fasteners
Furnishings
Labels
Machines
Paper
Phones
Storage
Supplies
Tables
Category