Dashboard_Python

August 20, 2025

[1]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import numpy as np
```

[2]:

```python
df = pd.read_excel('SuperStore_Dashboard_pyt.xlsx')
df
```

[2]:

```
       Row ID         Order ID  Year Month-Year Order Date  Ship Date  \
0       42433     AG-2011-2040  2011   Jan 2011 2011-01-01 2011-01-06
1       22253    IN-2011-47883  2011   Jan 2011 2011-01-01 2011-01-08
2       48883     HU-2011-1220  2011   Jan 2011 2011-01-01 2011-01-05
3       11731  IT-2011-3647632  2011   Jan 2011 2011-01-01 2011-01-05
4       22255    IN-2011-47883  2011   Jan 2011 2011-01-01 2011-01-08
…       …              …   …        …        …        …
51285   32593   CA-2014-115427  2014   Dec 2014 2014-12-31 2015-01-04
51286   47594     MO-2014-2560  2014   Dec 2014 2014-12-31 2015-01-05
51287    8857   MX-2014-110527  2014   Dec 2014 2014-12-31 2015-01-02
51288    6852   MX-2014-114783  2014   Dec 2014 2014-12-31 2015-01-06
51289   36388   CA-2014-156720  2014   Dec 2014 2014-12-31 2015-01-04

       Shipping Days       Ship Mode Customer ID     Customer Name  …  \
0                  5  Standard Class    TB-11280   Toby Braunhardt  …
1                  7  Standard Class    JH-15985       Joseph Holt  …
2                  4    Second Class      AT-735     Annie Thurman  …
3                  4    Second Class    EM-14140      Eugene Moren  …
4                  7  Standard Class    JH-15985       Joseph Holt  …
…              …             …         …               …  …
51285              4  Standard Class    EB-13975        Erica Bern  …
51286              5  Standard Class     LP-7095         Liz Preis  …
51287              2    Second Class    CM-12190  Charlotte Melton  …
51288              6  Standard Class    TD-20995     Tamara Dahlen  …
51289              4  Standard Class    JM-15580     Jill Matthias  …

             Product ID         Category Sub-Category  \
0      OFF-TEN-10000025  Office Supplies      Storage
1       OFF-SU-10000618  Office Supplies     Supplies
```

```
2      OFF-TEN-10001585  Office Supplies      Storage
3       OFF-PA-10001492  Office Supplies        Paper
4       FUR-FU-10003447        Furniture  Furnishings
…                   …              …          …
51285   OFF-BI-10002103  Office Supplies      Binders
51286  OFF-WIL-10001069  Office Supplies      Binders
51287   OFF-LA-10004182  Office Supplies       Labels
51288   OFF-LA-10000413  Office Supplies       Labels
51289   OFF-FA-10003472  Office Supplies    Fasteners

                                     Product Name    Sales Quantity  \
0                              Tenex Lockers, Blue  408.300        2
1                         Acme Trimmer, High Speed  120.366        3
2                          Tenex Box, Single Width   66.120        4
3                       Enermax Note Cards, Premium   44.865        3
4                         Eldon Light Bulb, Duo Pack  113.670        5
…                                                …        …       …
51285  Cardinal Slant-D Ring Binder, Heavy Gauge Vinyl   13.904        2
51286          Wilson Jones Hole Reinforcements, Clear    3.990        1
51287           Hon Color Coded Labels, 5000 Label Set   26.400        3
51288           Hon Legal Exhibit Labels, Alphabetical    7.120        1
51289                              Bagged Rubber Bands    3.024        3

      Discount    Profit Shipping Cost Order Priority
0          0.0  106.1400         35.46         Medium
1          0.1   36.0360          9.72         Medium
2          0.0   29.6400          8.17           High
3          0.5  -26.0550          4.82           High
4          0.1   37.7700          4.70         Medium
…          …        …            …            …
51285      0.2    4.5188          0.89         Medium
51286      0.0    0.4200          0.49         Medium
51287      0.0   12.3600          0.35         Medium
51288      0.0    0.5600          0.20         Medium
51289      0.2   -0.6048          0.17         Medium

[51290 rows x 27 columns]
```

[4]:

```
df.columns
```

[4]:

```
Index(['Row ID', 'Order ID', 'Year', 'Month-Year', 'Order Date', 'Ship Date',
       'Shipping Days', 'Ship Mode', 'Customer ID', 'Customer Name', 'Segment',
       'City', 'State', 'Country', 'Postal Code', 'Market', 'Region',
       'Product ID', 'Category', 'Sub-Category', 'Product Name', 'Sales',
       'Quantity', 'Discount', 'Profit', 'Shipping Cost', 'Order Priority'],
      dtype='object')
```

```
[20]: df.isnull().sum()
```

```
[20]: Row ID                0
      Order ID              0
      Year                  0
      Month-Year            0
      Order Date            0
      Ship Date             0
      Shipping Days         0
      Ship Mode             0
      Customer ID           0
      Customer Name         0
      Segment               0
      City                  0
      State                 0
      Country               0
      Postal Code       41296
      Market                0
      Region                0
      Product ID            0
      Category              0
      Sub-Category          0
      Product Name          0
      Sales                 0
      Quantity              0
      Discount              0
      Profit                0
      Shipping Cost         0
      Order Priority        0
      dtype: int64
```

```
[22]: print(df.shape)
```

```
(51290, 27)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 27 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          51290 non-null  int64
 1   Order ID        51290 non-null  object
 2   Year            51290 non-null  int64
 3   Month-Year      51290 non-null  object
 4   Order Date      51290 non-null  datetime64[ns]
 5   Ship Date       51290 non-null  datetime64[ns]
 6   Shipping Days   51290 non-null  int64
 7   Ship Mode       51290 non-null  object
 8   Customer ID     51290 non-null  object
 9   Customer Name   51290 non-null  object
 10  Segment         51290 non-null  object
 11  City            51290 non-null  object
 12  State           51290 non-null  object
 13  Country         51290 non-null  object
 14  Postal Code     9994 non-null   float64
 15  Market          51290 non-null  object
 16  Region          51290 non-null  object
 17  Product ID      51290 non-null  object
 18  Category        51290 non-null  object
 19  Sub-Category    51290 non-null  object
 20  Product Name    51290 non-null  object
 21  Sales           51290 non-null  float64
 22  Quantity        51290 non-null  int64
 23  Discount        51290 non-null  float64
 24  Profit          51290 non-null  float64
 25  Shipping Cost   51290 non-null  float64
 26  Order Priority  51290 non-null  object
dtypes: datetime64[ns](2), float64(5), int64(4), object(16)
memory usage: 10.6+ MB
```

[29]:
```
df.duplicated().sum()
```

[29]:
```
0
```

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

[30]:
```
             Row ID          Year                     Order Date  \
count  51290.00000  51290.000000                          51290
mean   25645.50000   2012.777208  2013-05-11 21:26:49.155780864
min        1.00000   2011.000000            2011-01-01 00:00:00
25%    12823.25000   2012.000000            2012-06-19 00:00:00
50%    25645.50000   2013.000000            2013-07-08 00:00:00
75%    38467.75000   2014.000000            2014-05-22 00:00:00
max    51290.00000   2014.000000            2014-12-31 00:00:00
std    14806.29199      1.098931                            NaN

                           Ship Date  Shipping Days   Postal Code  \
count                          51290   51290.000000   9994.000000
mean   2013-05-15 20:42:42.745174528       3.969370  55190.379428
min              2011-01-03 00:00:00       0.000000   1040.000000
25%              2012-06-23 00:00:00       3.000000  23223.000000
50%              2013-07-12 00:00:00       4.000000  56430.500000
75%              2014-05-26 00:00:00       5.000000  90008.000000
```

```
max              2015-01-07 00:00:00       7.000000  99301.000000
std                              NaN       1.729437  32063.693350

              Sales      Quantity      Discount        Profit  Shipping Cost
count  51290.000000  51290.000000  51290.000000  51290.000000   51290.000000
mean     246.490581      3.476545      0.142908     28.610982      26.375915
min        0.444000      1.000000      0.000000  -6599.978000       0.000000
25%       30.758625      2.000000      0.000000      0.000000       2.610000
50%       85.053000      3.000000      0.000000      9.240000       7.790000
75%      251.053200      5.000000      0.200000     36.810000      24.450000
max    22638.480000     14.000000      0.850000   8399.976000     933.570000
std      487.565361      2.278766      0.212280    174.340972      57.296804
```

[ ]:

```

```

[3]:

```
# KPIs
total_sales = df["Sales"].sum()
total_profit = df["Profit"].sum()
unique_customers = df["Customer Name"].nunique()
avg_shipping_days = df["Shipping Days"].mean()
profit_margin = (df["Profit"].sum() / df["Sales"].sum()) * 100
avg_order_value = df.groupby("Order ID")["Sales"].sum().mean()
total_orders = df["Order ID"].nunique()

# --- Sales Statistics ---
mean_sales = df["Sales"].mean()
median_sales = df["Sales"].median()
mode_sales = df["Sales"].mode()[0]
range_sales = df["Sales"].max() - df["Sales"].min()
variance_sales = df["Sales"].var()
std_dev_sales = df["Sales"].std()
q1_sales, q2_sales, q3_sales = df["Sales"].quantile([0.25, 0.5, 0.75])
```

[12]:

```
# Statistics KPIs
print("\033[1mTotal Sales:\033[0m", f"${total_sales:,.0f}")
print("\033[1mTotal Profit:\033[0m", f"${total_profit:,.0f}")
print("\033[1mTotal Customers:\033[0m", unique_customers)
print("\033[1mTotal Orders:\033[0m", total_orders)
print("\033[1mAvg. Shipping Days:\033[0m", avg_shipping_days)
print("\033[1mProfit Margin:\033[0m", profit_margin)
print("\033[1mAvg. Order Values:\033[0m", avg_order_value)
print("\n")
# Sales Statistics
print("\033[1mSales Mean:\033[0m", mean_sales)
print("\033[1mSales Median:\033[0m",f"{median_sales:,.2f}")
print("\033[1mSales Mode:\033[0m", mode_sales)
print("\033[1mSales Range:\033[0m", range_sales)
```

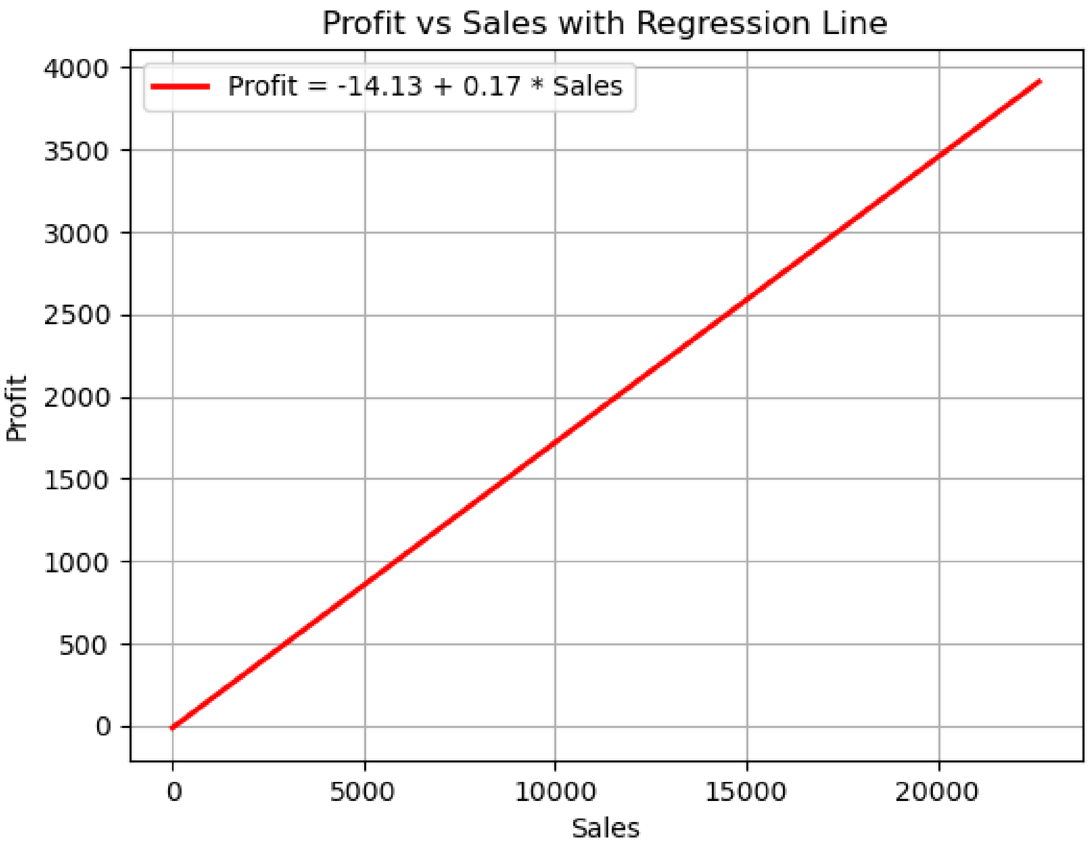
```python
print("\033[1mSales Variance:\033[0m", f"{variance_sales:,.2f}")
print("\033[1mSales Std Dev.:\033[0m", f"{std_dev_sales:,.2f}")
print("\033[1mQuartiles (Q1, Q2, Q3):\033[0m", f"{q1_sales:,.2f}, {q2_sales:,.
 ↪2f}, {q3_sales:,.2f}")
```

```
Total Sales: $12,642,502
Total Profit: $1,467,457
Total Customers: 795
Total Orders: 25035
Avg. Shipping Days: 3.96937024761162
Profit Margin: 11.607332960995603
Avg. Order Values: 504.99308607469555


Sales Mean: 246.49058120257362
Sales Median: 85.05
Sales Mode: 12.96
Sales Range: 22638.036
Sales Variance: 237,719.98
Sales Std Dev.: 487.57
Quartiles (Q1, Q2, Q3): 30.76, 85.05, 251.05
```

[17]:

```python
# --- Regression Line: Profit ~ Sales ---
import statsmodels.api as sm
import matplotlib.pyplot as plt

X = sm.add_constant(df["Sales"])  # adds a column of 1s (for intercept)
y = df["Profit"]                  # dependent variable
model = sm.OLS(y, X).fit()        # fit Ordinary Least Squares regression

intercept, slope = model.params   # extract intercept & slope
regression_eq = f"Profit = {intercept:.2f} + {slope:.2f} * Sales"

print(f"Regression Line: {regression_eq}")

# Regression line
x_vals = np.linspace(df["Sales"].min(), df["Sales"].max(), 100)
y_vals = intercept + slope * x_vals
plt.plot(x_vals, y_vals, color="red", linewidth=2, label=regression_eq)

# Labels and title
plt.xlabel("Sales")
plt.ylabel("Profit")
plt.title("Profit vs Sales with Regression Line")
plt.legend()
plt.grid(True)
plt.show()
```

```
Regression Line: Profit = -14.13 + 0.17 * Sales
```



```
[6]: kpi_data = {
         "Total Sales": f"${total_sales:,.0f}",
         "Total Profit": f"${total_profit:,.0f}",
         "Avg Profit Margin": f"{profit_margin:.2%}",
         "Total Customers": unique_customers,
         "Total Orders": total_orders,
         "Shipping Days": f"{avg_shipping_days:.2}"
     }

     kpi_data
```

```
[6]: {'Total Sales': '$12,642,502',
      'Total Profit': '$1,467,457',
      'Avg Profit Margin': '1160.73%',
      'Total Customers': 795,
      'Total Orders': 25035,
      'Shipping Days': '4.0'}
```

[ ]:

```
 
```

[7]:

```
from IPython.display import display, HTML

kpi_html = f"""
<div style='display: flex; gap: 20px;'>
    <div style='background: #4CAF50; color: white; padding: 20px; border-radius:
  ↪ 8px;'>
        <h3>Total Sales</h3>
        <p style='font-size: 24px;'>{kpi_data['Total Sales']}</p>
    </div>
    <div style='background: #2196F3; color: white; padding: 20px; border-radius:
  ↪ 8px;'>
        <h3>Total Profit</h3>
        <p style='font-size: 24px;'>{kpi_data['Total Profit']}</p>
    </div>
    <div style='background: #2196F3; color: white; padding: 20px; border-radius:
  ↪ 8px;'>
        <h3>Total Orders</h3>
        <p style='font-size: 24px;'>{kpi_data['Total Orders']}</p>
    </div>
     <div style='background: #9C27B0; color: white; padding: 20px;␣
  ↪border-radius: 8px;'>
        <h3>Total Customers</h3>
        <p style='font-size: 24px;'>{kpi_data['Total Customers']}</p>
    </div>
     <div style='background: #9C27B0; color: white; padding: 20px;␣
  ↪border-radius: 8px;'>
        <h3>Avg. Shipping Days</h3>
        <p style='font-size: 24px;'>{kpi_data['Shipping Days']}</p>
    </div>
    <div style='background: #FF9800; color: white; padding: 20px; border-radius:
  ↪ 8px;'>
        <h3>Avg Profit Margin</h3>
        <p style='font-size: 24px;'>{kpi_data['Avg Profit Margin']}</p>
    </div>

</div>
"""

display(HTML(kpi_html))
```

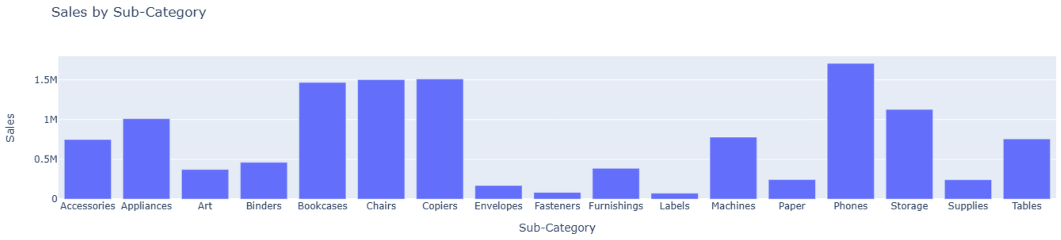
```
<IPython.core.display.HTML object>
```

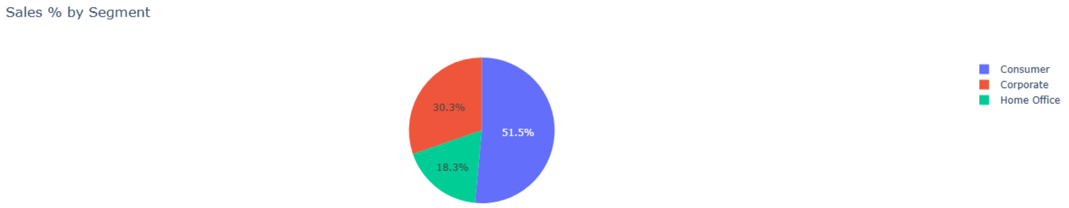
[ ]:

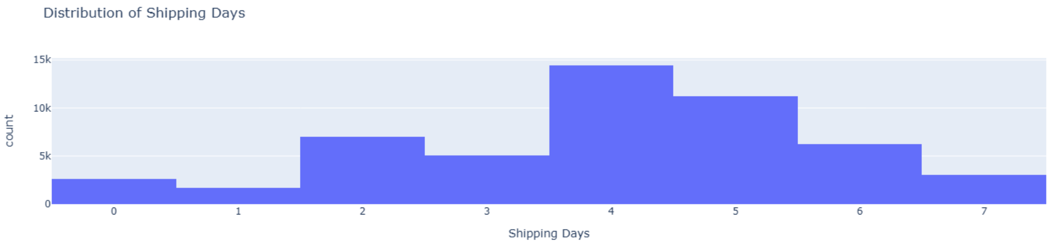
```
 
```

[17]:

```python
import plotly.express as px

sales_by_category = px.bar(
    df.groupby('Sub-Category', as_index=False)['Sales'].sum(),
    x='Sub-Category', y='Sales', title="Sales by Sub-Category"
)

sales_by_segment = px.pie(
    df, names='Segment', values='Sales', title="Sales % by Segment"
)

shipping_days_hist = px.histogram(
    df, x='Shipping Days', title="Distribution of Shipping Days"
)


sales_by_category.show()
sales_by_segment.show()
shipping_days_hist.show()
```



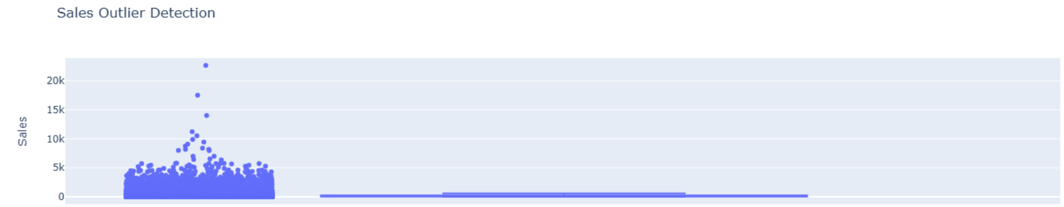


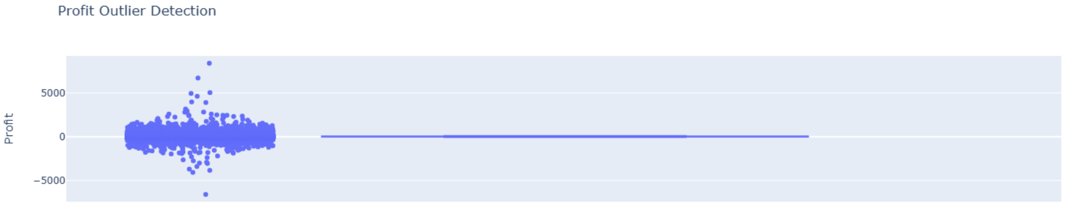


```
[15]: Q1 = df['Sales'].quantile(0.25)
      Q3 = df['Sales'].quantile(0.75)
      IQR = Q3 - Q1

      lower_bound = Q1 - 1.5 * IQR
      upper_bound = Q3 + 1.5 * IQR

      # Boolean mask for outliers
      outliers = (df['Sales'] < lower_bound) | (df['Sales'] > upper_bound)
      print(f"No. of outliers in Sales: {outliers.sum()}")
```

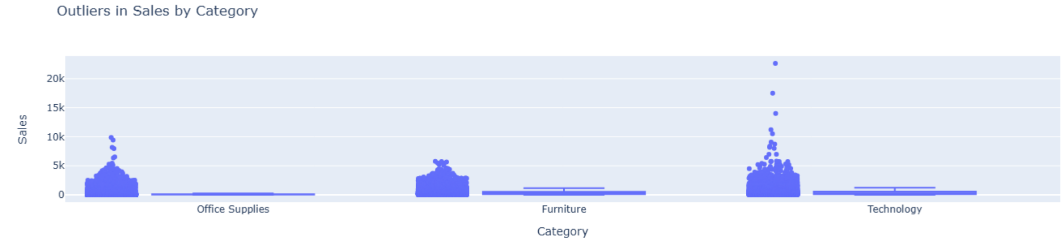
```
No. of outliers in Sales: 5655
```

```
[18]: #Visualize Outliers with a Boxplot
      boxplot_sales = px.box(
          df,
          y="Sales",
          points="all",  # show all points including outliers
          title="Sales Outlier Detection"
      )

      boxplot_profit = px.box(
          df,
          y="Profit",
          points="all",
          title="Profit Outlier Detection"
      )

      boxplot_sales.show()
      boxplot_profit.show()
```

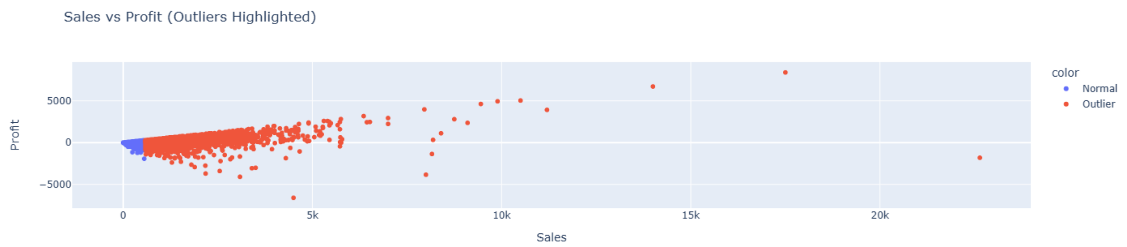




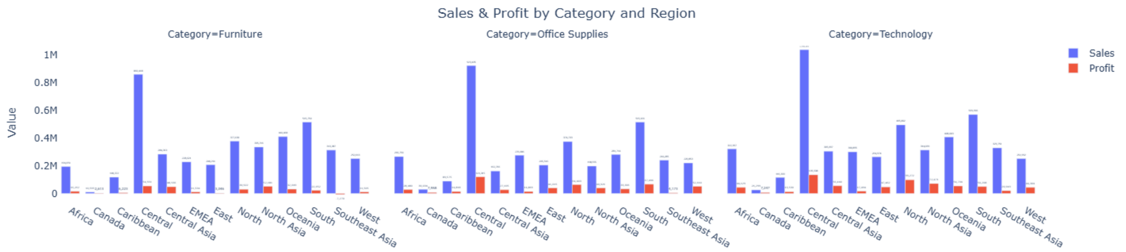
[20]:
```
# Side-by-Side Boxplot by Category (to see outliers per group)
boxplot_by_category = px.box(
    df,
    x="Category",
    y="Sales",
    points="all",
    title="Outliers in Sales by Category"
)
boxplot_by_category.show()
```



[21]:
```
# Highlight Outliers in a Scatter Plot
scatter_outliers = px.scatter(
    df,
    x="Sales",
    y="Profit",
    color=outliers.map({True: "Outlier", False: "Normal"}),
    title="Sales vs Profit (Outliers Highlighted)"
)
scatter_outliers.show()
```

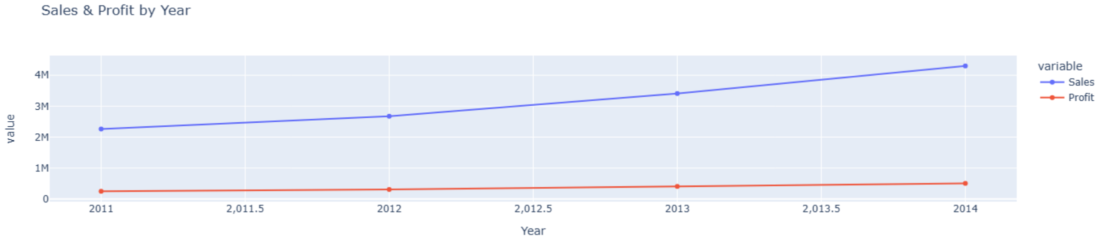


[26]:
```
# Sales & Profit by Category and Region
sales_profit_category_region = px.bar(
    df.groupby(['Category', 'Region'], as_index=False)[['Sales', 'Profit']].
 ↪sum().melt(
        id_vars=['Category', 'Region'], value_vars=['Sales', 'Profit']
    ),
    x="Region", y="value", color="variable",
    barmode="group", facet_col="Category",  # facet by Category instead
    title="Sales & Profit by Category and Region",
    text="value"  # add labels
)

# Update layout for cleaner look
sales_profit_category_region.update_traces(texttemplate='%{text:,.0f}',␣
 ↪textposition="outside")
sales_profit_category_region.update_layout(
    legend_title_text="",  # remove extra title
    yaxis_title="Value",
    xaxis_title="Region",
    bargap=0.25,  # space between bars
    plot_bgcolor="white",
    title_x=0.5,  # center title
    font=dict(size=12),
    margin=dict(l=50, r=20, t=60, b=80)
)

sales_profit_category_region.show()
```



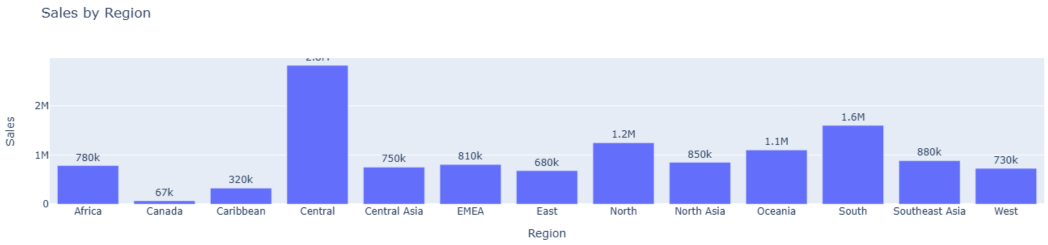
[22]:

```python
# Sales & Profit by Year
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Year'] = df['Order Date'].dt.year

sales_profit_year = px.line(
    df.groupby('Year', as_index=False)[['Sales', 'Profit']].sum().melt(
        id_vars='Year', value_vars=['Sales', 'Profit']
    ),
    x="Year", y="value", color="variable",
    markers=True,
    title="Sales & Profit by Year"
)
sales_profit_year.show()
```



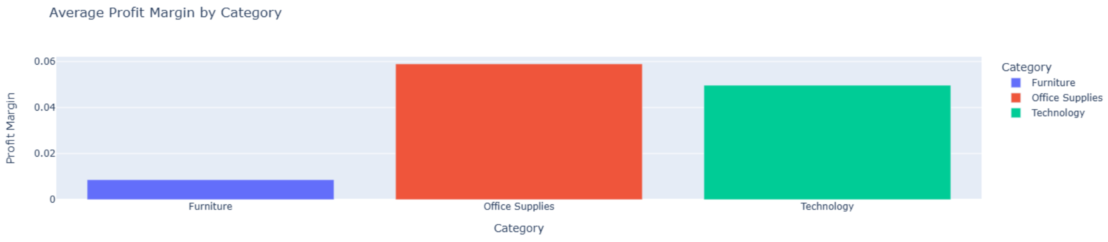
[24]:

```python
# Sales by Region
sales_region = px.bar(
    df.groupby('Region', as_index=False)['Sales'].sum(),
    x="Region", y="Sales", text="Sales",
    title="Sales by Region"
)
sales_region.update_traces(texttemplate='%{text:.2s}', textposition="outside")
sales_region.show()
```



[25]:

```python
# Average Profit Margin by Category
df['Profit Margin'] = df['Profit'] / df['Sales']

profit_margin_category = px.bar(
    df.groupby('Category', as_index=False)['Profit Margin'].mean(),
    x="Category", y="Profit Margin", color="Category",
    title="Average Profit Margin by Category"
)
profit_margin_category.show()
```



[18]:

```python
df.to_excel("superstore_cleaned.xlsx", index=False)
```

[ ]:

```python

```