```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import FuncFormatter

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv(r'C:\Users\Mohamed Fawzi\Desktop\Product Sales
Analysis\sales_data_sample.csv', encoding='ISO-8859-1')

df.head(2)
```


{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"ORDERNUMBER","rawType":"int64","type":"integer"},
{"name":"QUANTITYORDERED","rawType":"int64","type":"integer"},
{"name":"PRICEEACH","rawType":"float64","type":"float"},
{"name":"ORDERLINENUMBER","rawType":"int64","type":"integer"},
{"name":"SALES","rawType":"float64","type":"float"},
{"name":"ORDERDATE","rawType":"object","type":"string"},
{"name":"STATUS","rawType":"object","type":"string"},
{"name":"QTR_ID","rawType":"int64","type":"integer"},
{"name":"MONTH_ID","rawType":"int64","type":"integer"},
{"name":"YEAR_ID","rawType":"int64","type":"integer"},
{"name":"PRODUCTLINE","rawType":"object","type":"string"},
{"name":"MSRP","rawType":"int64","type":"integer"},
{"name":"PRODUCTCODE","rawType":"object","type":"string"},
{"name":"CUSTOMERNAME","rawType":"object","type":"string"},
{"name":"PHONE","rawType":"object","type":"string"},
{"name":"ADDRESSLINE1","rawType":"object","type":"string"},
{"name":"ADDRESSLINE2","rawType":"object","type":"unknown"},
{"name":"CITY","rawType":"object","type":"string"},
{"name":"STATE","rawType":"object","type":"unknown"},
{"name":"POSTALCODE","rawType":"object","type":"string"},
{"name":"COUNTRY","rawType":"object","type":"string"},
{"name":"TERRITORY","rawType":"object","type":"unknown"},
{"name":"CONTACTLASTNAME","rawType":"object","type":"string"},
{"name":"CONTACTFIRSTNAME","rawType":"object","type":"string"},
{"name":"DEALSIZE","rawType":"object","type":"string"}],"ref":"c48b8ec
d-2e9b-4702-96a1-bda85aa9541c","rows":
[["0","10107","30","95.7","2","2871.0","2/24/2003
0:00","Shipped","1","2","2003","Motorcycles","95","S10_1678","Land of
Toys Inc.","2125557818","897 Long Airport
Avenue",null,"NYC","NY","10022","USA",null,"Yu","Kwai","Small"],
["1","10121","34","81.35","5","2765.9","5/7/2003
0:00","Shipped","2","5","2003","Motorcycles","95","S10_1678","Reims
Collectables","26.47.1555","59 rue de
l'Abbaye",null,"Reims",null,"51100","France","EMEA","Henriot","Paul","
Small"]],"shape":{"columns":25,"rows":2}}

Inspecting & Preparing Data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ORDERNUMBER       2823 non-null   int64  
 1   QUANTITYORDERED   2823 non-null   int64  
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64  
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object 
 6   STATUS            2823 non-null   object 
 7   QTR_ID            2823 non-null   int64  
 8   MONTH_ID          2823 non-null   int64  
 9   YEAR_ID           2823 non-null   int64  
 10  PRODUCTLINE       2823 non-null   object 
 11  MSRP              2823 non-null   int64  
 12  PRODUCTCODE       2823 non-null   object 
 13  CUSTOMERNAME      2823 non-null   object 
 14  PHONE             2823 non-null   object 
 15  ADDRESSLINE1      2823 non-null   object 
 16  ADDRESSLINE2      302 non-null    object 
 17  CITY              2823 non-null   object 
 18  STATE             1337 non-null   object 
 19  POSTALCODE        2747 non-null   object 
 20  COUNTRY           2823 non-null   object 
 21  TERRITORY         1749 non-null   object 
 22  CONTACTLASTNAME   2823 non-null   object 
 23  CONTACTFIRSTNAME  2823 non-null   object 
 24  DEALSIZE          2823 non-null   object 
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

```
df.describe()
```

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"ORDERNUMBER","rawType":"float64","type":"float"},
{"name":"QUANTITYORDERED","rawType":"float64","type":"float"},
{"name":"PRICEEACH","rawType":"float64","type":"float"},
{"name":"ORDERLINENUMBER","rawType":"float64","type":"float"},
{"name":"SALES","rawType":"float64","type":"float"},
{"name":"QTR_ID","rawType":"float64","type":"float"},
{"name":"MONTH_ID","rawType":"float64","type":"float"},
{"name":"YEAR_ID","rawType":"float64","type":"float"},
{"name":"MSRP","rawType":"float64","type":"float"}],"ref":"3460aab5-6111-497d-89fc-18c36d6ed705","rows":

[["count","2823.0","2823.0","2823.0","2823.0","2823.0","2823.0","2823.0","2823.0","2823.0"],
["mean","10258.725115125753","35.09280906836698","83.65854410201914","6.466170740347148","3553.889071909316","2.7176762309599716","7.0924548352816155","2003.8150903294368","100.71555083244775"],
["std","92.08547759571952","9.74144273706961","20.174276527840703","4.225840964690942","1841.8651057401805","1.203878088001747","3.656633307661775","0.6996701541300782","40.18791167720310 4"],
["min","10100.0","6.0","26.88","1.0","482.13","1.0","1.0","2003.0","33.0"],
["25%","10180.0","27.0","68.86","3.0","2203.4300000000003","2.0","4.0","2003.0","68.0"],
["50%","10262.0","35.0","95.7","6.0","3184.8","3.0","8.0","2004.0","99.0"],
["75%","10333.5","43.0","100.0","9.0","4508.0","4.0","11.0","2004.0","124.0"],
["max","10425.0","97.0","100.0","18.0","14082.8","4.0","12.0","2005.0","214.0"]],"shape":{"columns":9,"rows":8}}

```python
# I will rename the columns to be lowercase for easier access
df.columns = df.columns.str.lower()
df.columns
```

```
Index(['ordernumber', 'quantityordered', 'priceeach',
'orderlinenumber',
       'sales', 'orderdate', 'status', 'qtr_id', 'month_id',
'year_id',
       'productline', 'msrp', 'productcode', 'customername', 'phone',
       'addressline1', 'addressline2', 'city', 'state', 'postalcode',
       'country', 'territory', 'contactlastname', 'contactfirstname',
       'dealsize'],
      dtype='object')
```

```python
# Check for missing values
df.isnull().sum()
```

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"0","rawType":"int64","type":"integer"}],"ref":"026c4348-435f-41c7-a0cc-6aab1c197415","rows":[["ordernumber","0"],
["quantityordered","0"],["priceeach","0"],["orderlinenumber","0"],
["sales","0"],["orderdate","0"],["status","0"],["qtr_id","0"],
["month_id","0"],["year_id","0"],["productline","0"],["msrp","0"],
["productcode","0"],["customername","0"],["phone","0"],
["addressline1","0"],["addressline2","2521"],["city","0"],
["state","1486"],["postalcode","76"],["country","0"],
["territory","1074"],["contactlastname","0"],["contactfirstname","0"],
["dealsize","0"]],"shape":{"columns":1,"rows":25}}

- I noticed that the territory column missing values based on the country column are for USA and Canada. To handle these missing values we're going to map territory column based on the country column.

```
# define the mapping of countries to territories
territory_mapping = {
    "USA": "North America",
    "Spain": "EMEA",
    "France": "EMEA",
    "Australia": "APAC",
    "UK": "EMEA",
    "Italy": "EMEA",
    "Finland": "EMEA",
    "Norway": "EMEA",
    "Singapore": "APAC",
    "Canada": "North America",
    "Denmark": "EMEA",
    "Germany": "EMEA",
    "Sweden": "EMEA",
    "Austria": "EMEA",
    "Japan": "Japan",
    "Belgium": "EMEA",
    "Switzerland": "EMEA",
    "Philippines": "APAC",
    "Ireland": "EMEA"
}

# replace missing values in 'territory' column based on 'country'
df['territory'] = df['country'].map(territory_mapping)

#df.isna().sum()

# also I will replace the missing values in state column with
'Unknown'
df['state'].fillna('Unknown', inplace=True)
# and I will replace the missing values in postal code column with
'00000'
df['postalcode'].fillna('00000', inplace=True)

df.isna().sum()
```

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"0","rawType":"int64","type":"integer"}],"ref":"e3505ae9-e907-4a4f-8d04-62db8e74cdbe","rows":[["ordernumber","0"],
["quantityordered","0"],["priceeach","0"],["orderlinenumber","0"],
["sales","0"],["orderdate","0"],["status","0"],["qtr_id","0"],
["month_id","0"],["year_id","0"],["productline","0"],["msrp","0"],
["productcode","0"],["customername","0"],["phone","0"],
["addressline1","0"],["addressline2","2521"],["city","0"],
["state","0"],["postalcode","0"],["country","0"],["territory","0"],

["contactlastname","0"],["contactfirstname","0"],
["dealsize","0"]],"shape":{"columns":1,"rows":25}}

```python
# check for duplicates
df.duplicated().sum()
```

```
0
```

```python
# drop columns that are not needed for analysis
columns_to_drop = ['phone', 'addressline1', 'addressline2', ]
df.drop(columns=columns_to_drop, inplace=True)
```

```python
# converting the order date to datetime
df['orderdate'] = pd.to_datetime(df['orderdate'])
# then I will extract the name of the month
df['month'] = df['orderdate'].dt.month_name()
```

```python
df.info()
```

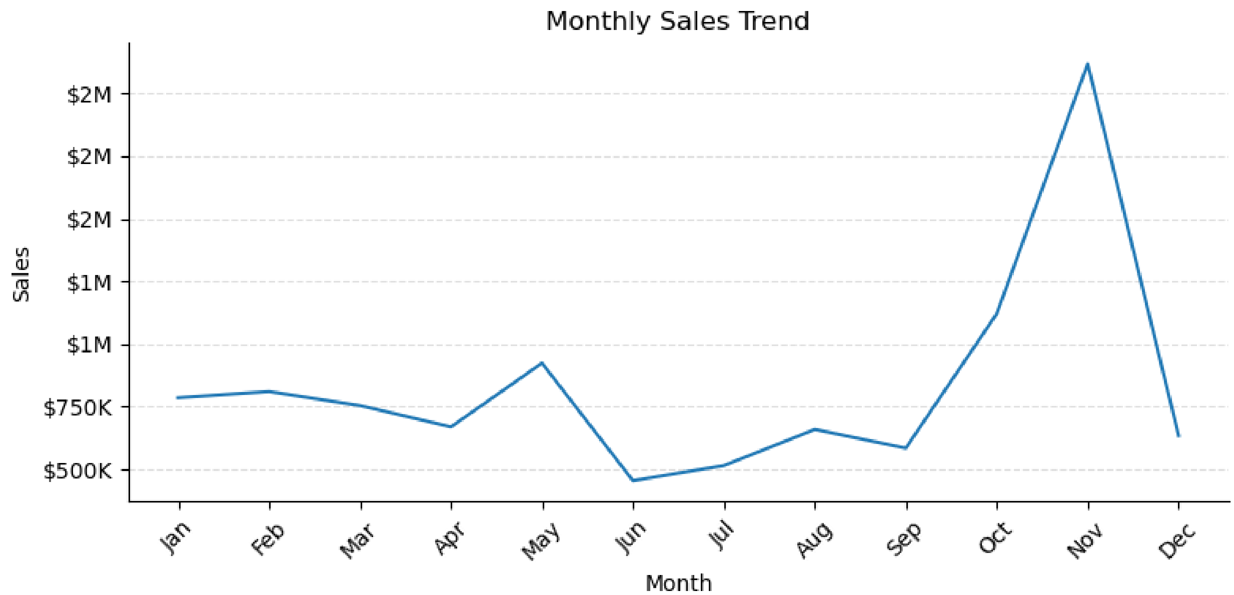
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ordernumber       2823 non-null   int64
 1   quantityordered   2823 non-null   int64
 2   priceeach         2823 non-null   float64
 3   orderlinenumber   2823 non-null   int64
 4   sales             2823 non-null   float64
 5   orderdate         2823 non-null   datetime64[ns]
 6   status            2823 non-null   object
 7   qtr_id            2823 non-null   int64
 8   month_id          2823 non-null   int64
 9   year_id           2823 non-null   int64
 10  productline       2823 non-null   object
 11  msrp              2823 non-null   int64
 12  productcode       2823 non-null   object
 13  customername      2823 non-null   object
 14  city              2823 non-null   object
 15  state             2823 non-null   object
 16  postalcode        2823 non-null   object
 17  country           2823 non-null   object
 18  territory         2823 non-null   object
 19  contactlastname   2823 non-null   object
 20  contactfirstname  2823 non-null   object
 21  dealsize          2823 non-null   object
 22  month             2823 non-null   object
dtypes: datetime64[ns](1), float64(2), int64(7), object(13)
memory usage: 507.4+ KB
```

```
#df.to_csv(r'C:\Users\Mohamed Fawzi\Desktop\Product Sales Analysis\
sales_data_cleaned.csv', index=False)
```

Time Series Analysis:

Monthly Sales Trend

```
monthly_trend = df.groupby('month')
['sales'].sum().reset_index().round(2)
month_order = ['January', 'February', 'March', 'April', 'May', 'June',
'July', 'August', 'September', 'October', 'November', 'December']
monthly_trend['month'] = pd.Categorical(monthly_trend['month'],
categories=month_order, ordered=True)
monthly_trend = monthly_trend.sort_values('month')
# convert month names to abbreviations to make the plot cleaner
monthly_trend['month'] = monthly_trend['month'].str[:3]


plt.figure(figsize=(8, 4))
sns.lineplot(
    data=monthly_trend,
    x='month',
    y='sales',
    palette='viridis'
    )

def currency(x, pos):
    if x >= 1e6:
        return '${:,.0f}M'.format(x * 1e-6)
    else:
        return '${:,.0f}K'.format(x * 1e-3)

plt.gca().yaxis.set_major_formatter(FuncFormatter(currency))

plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.4)

sns.despine()
plt.tight_layout()
plt.show()
```



- The sales performance across the year shows **significant variation**, indicating potential seasonality or promotional impact.
- **Key Observations**

  a. **Strongest Month:**

  - **November** experienced the highest spike in sales, surpassing **$2M**.
  - This likely corresponds with **seasonal promotions** such as Black Friday or early holiday shopping.

  b. **Second Strongest Month:**

  - **October** also shows a notable increase in sales, over **$1M**, suggesting the buildup to peak season begins early.

  c. **Weakest Month:**

  - **June** marked the lowest sales point, just below **$500K**.
  - This may indicate a **mid-year slump**, potentially due to lower consumer spending or limited marketing activities.

  d. **Consistent Performance:**

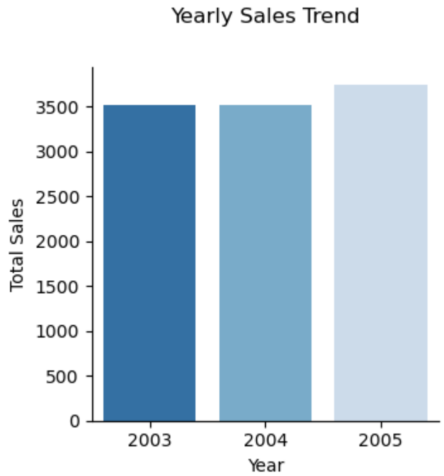
  - Sales remained relatively stable between **January and May**, ranging around **$700K–$900K**.
  - Indicates a **moderate baseline** level of activity in the first half of the year.

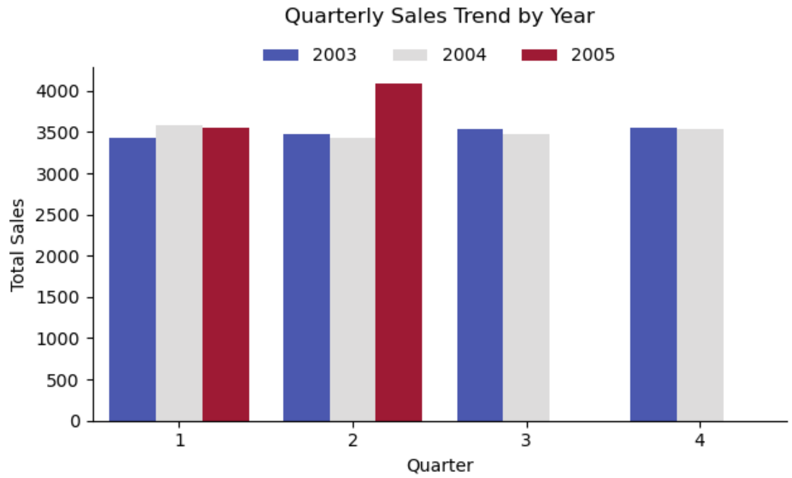
  e. **Sudden Drop in December:**

- After peaking in November, **sales dropped sharply in December**, down to levels similar to early in the year.

Quarterly & Yearly Trend

```
qrt_yr_trend = df.groupby(['year_id', 'qtr_id'])
['sales'].sum().reset_index()


import matplotlib.gridspec as gridspec

fig = plt.figure(figsize=(10, 4))
gs = gridspec.GridSpec(1, 2, width_ratios=[1, 2])  # Adjust width
ratios

# Yearly Sales Trend (smaller)
ax0 = fig.add_subplot(gs[0])
sns.barplot(data=df, x='year_id', y='sales', ax=ax0,
palette='Blues_r', ci=False)
ax0.set_title("Yearly Sales Trend", pad=25)
ax0.set_xlabel("Year")
ax0.set_ylabel("Total Sales")

# Quarterly Sales Trend (larger)
ax1 = fig.add_subplot(gs[1])
sns.barplot(data=df, x='qtr_id', y='sales', hue='year_id', ax=ax1,
palette='coolwarm', ci=False)
ax1.set_title("Quarterly Sales Trend by Year", pad=25)
ax1.set_xlabel("Quarter")
ax1.set_ylabel("Total Sales")
ax1.legend(loc='upper center', ncols=3, bbox_to_anchor=(0.5, 1.1),
frameon=False)

sns.despine()
plt.tight_layout()
plt.show()
```



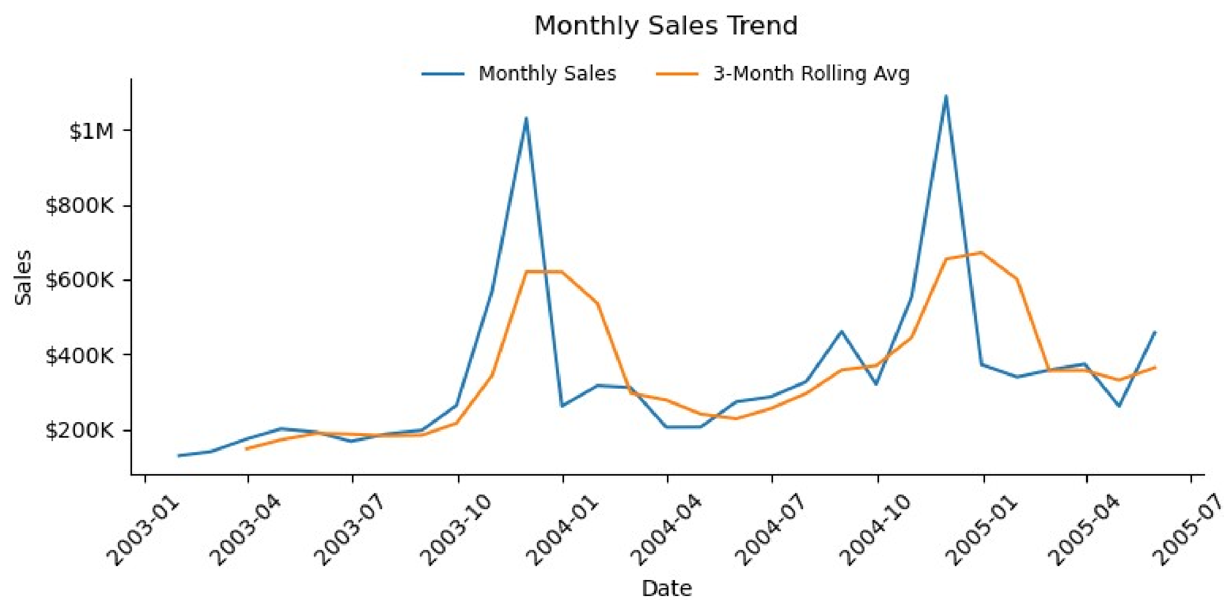


- **Yearly Sales Trend (2003–2005)**
  - **2003 and 2004** had nearly identical total sales, both around **$3,500**.
  - **2005** showed a slight improvement in total sales, crossing the **$3,700** mark.
  - Indicates a **steady but modest growth** trend over the 3-year period.
- **Quarterly Sales Trend by Year**
  - **Quarter 1**:
    - All three years performed well.
    - **2005** leads slightly, showing a strong start to the year.
  - **Quarter 2**:
    - **2005** significantly outperformed previous years, with the **highest single-quarter performance** across all years.
    - Both 2003 and 2004 maintained similar levels to Quarter 1.
  - **Quarter 3**:
    - Sales remained steady for all years, with minimal variation.
    - Suggests consistent performance during mid-year months.
  - **Quarter 4**:
    - Very similar sales values across all years.
    - Indicates **stable end-of-year sales**, possibly driven by recurring seasonal patterns.

```
df_advanced = df.copy() # Create a copy of the original DataFrame for advanced analysis
# Convert 'orderdate' to datetime format
df_advanced['orderdate'] = pd.to_datetime(df_advanced['orderdate'])

# Resample data to daily/weekly/monthly level for trend analysis
df_advanced.set_index('orderdate', inplace=True)
monthly_sales = df_advanced['sales'].resample('M').sum()
```

- Now let's analyze the monthly sales trends and their 3-month rolling average, which will help to identify trends in noisy data.

```
rolling_avg = monthly_sales.rolling(window=3).mean()

plt.figure(figsize=(8, 4))

sns.lineplot(x=monthly_sales.index, y=monthly_sales.values,
label='Monthly Sales')
sns.lineplot(x=rolling_avg.index, y=rolling_avg.values, label='3-Month
Rolling Avg')

plt.gca().yaxis.set_major_formatter(FuncFormatter(currency)) #
function definied earlier...

plt.title('Monthly Sales Trend', pad=20)
plt.xlabel('Date')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.legend(ncols=2, fontsize=9, loc='upper center',
bbox_to_anchor=(0.5, 0, 0, 1.08), frameon=False)

sns.despine()
plt.tight_layout()
plt.show()
```



- The chart visualizes monthly sales from **2003 to mid-2005** alongside a **3-month rolling average**, which smooths short-term fluctuations.
- **Key Observations**
    a. **Seasonal Peaks**:
        - Major sales spikes are seen at the **end of each year**:
        - **Late 2003** and **late 2004** both show sharp peaks exceeding **$1M**.

- These may correlate with **holiday seasons or end-of-year promotions**.

b. **Rolling Average Smoothing**:
   - The **3-month rolling average** clearly highlights **underlying trends**, removing noise from monthly fluctuations.
   - Peaks in the rolling average lag slightly behind actual monthly spikes, as expected with smoothing.

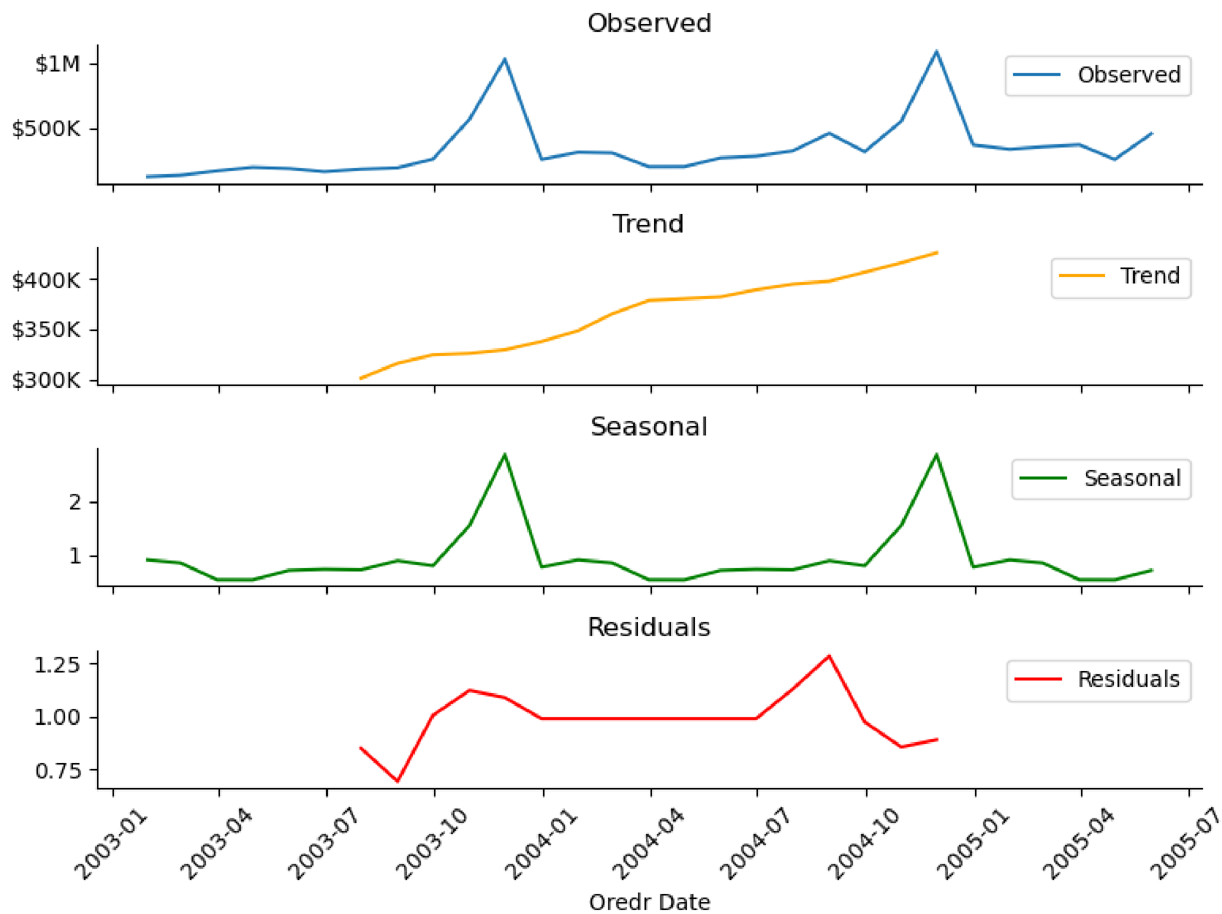
c. **Post-Peak Drop-offs**:
   - Following each year-end spike, there's a **notable sales decline** in the first quarter.
   - Indicates a **post-holiday slump**, consistent with reduced consumer demand after major sales periods.

d. **Mid-Year Stability**:
   - Sales during **mid-year months (Q2 and Q3)** show **relatively stable or modest growth**, with smaller fluctuations compared to year-end.

- Now, let's perform a seasonal decomposition of monthly sales using a multiplicative model to break it into 4 components, **Observes**, **Trend**, **Seasonal**, and **Residual**:
  - **Observed** : The raw monthly sales values.
  - **Trend** : Long-term movement in sales.
  - **Seasonal** : Repeating patterns within a year.
  - **Residuals** : Random noise or unexplained variability.

```python
from statsmodels.tsa.seasonal import seasonal_decompose

result = seasonal_decompose(monthly_sales, model='multiplicative')

fig, axes = plt.subplots(4, 1, figsize=(8, 6), sharex=True)

sns.lineplot(x=monthly_sales.index, y=monthly_sales.values,
ax=axes[0], label='Observed')
axes[0].set_title('Observed')

sns.lineplot(x=result.trend.index, y=result.trend.values, ax=axes[1],
label='Trend', color='orange')
axes[1].set_title('Trend')

sns.lineplot(x=result.seasonal.index, y=result.seasonal.values,
ax=axes[2], label='Seasonal', color='green')
axes[2].set_title('Seasonal')

sns.lineplot(x=result.resid.index, y=result.resid.values, ax=axes[3],
label='Residuals', color='red')
axes[3].set_title('Residuals')

axes[0].yaxis.set_major_formatter(FuncFormatter(currency))
axes[1].yaxis.set_major_formatter(FuncFormatter(currency))

plt.xlabel('Oredr Date')
```

```
plt.xticks(rotation=45)

sns.despine()
plt.tight_layout()
plt.show()
```



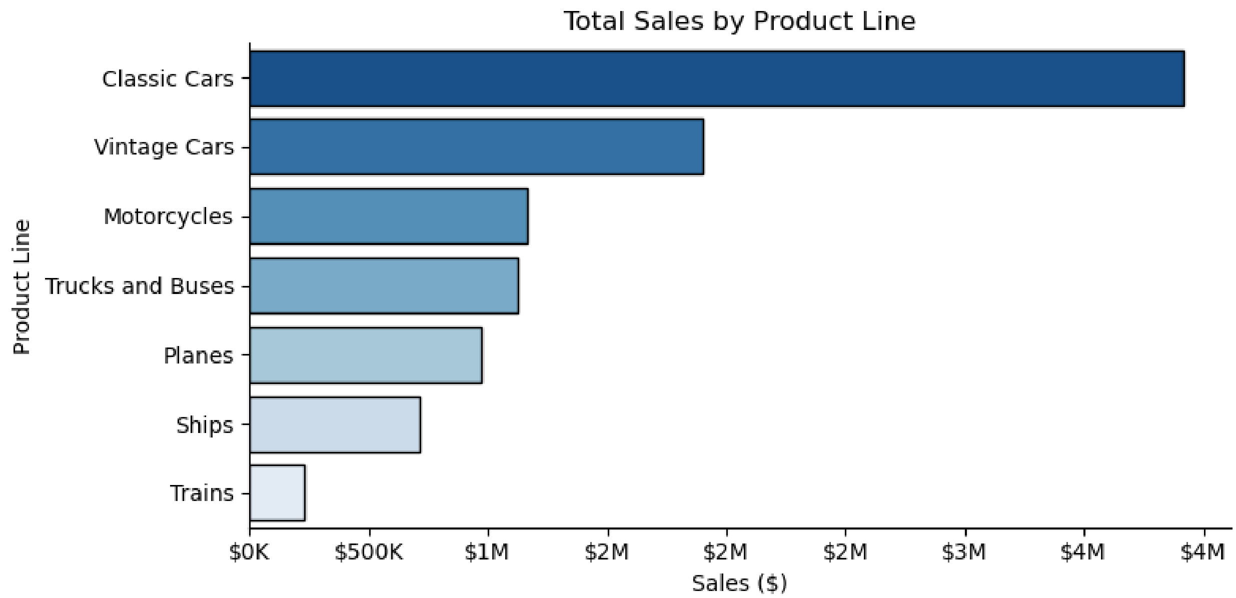
- **Key Insights**: This decomposition breaks down monthly sales into four components: **Observed**, **Trend**, **Seasonal**, and **Residuals**.
  - Observed
    - The raw sales data shows clear **spikes in late 2003 and late 2004**, crossing **$1M**.
    - Reflects **sharp seasonal patterns** with significant peaks and dips, consistent with earlier monthly sales trend findings.
  - Trend
    - A steady upward movement is seen from **early 2003 to mid-2005**.
    - The **trend component gradually increases**, rising from just above **$300K** to over **$400K**, suggesting **long-term growth** in sales performance.
  - Seasonal

        - Shows a repeating **seasonal structure**, with **notable spikes** aligning with end-of-year months (likely Q4).
        - Seasonal values occasionally exceed **2x the baseline**, emphasizing the strong **seasonality effect** on total sales.
    - Residuals
        - Residuals mostly hover between **0.75 and 1.25**, indicating **relatively low variance** beyond seasonality and trend.
        - A few brief periods (e.g., around **late 2004**) show higher residuals, suggesting **unexpected deviations**, possibly due to one-time events or promotions.

Product Performance Analysis:

- Now, we'll analyze the products line performance by sales

```
product_sales = df.groupby('productline')
['sales'].sum().reset_index().sort_values(by='sales', ascending=False)

plt.figure(figsize=(8, 4))
sns.barplot(
    data=product_sales,
    x='sales',
    y='productline',
    palette='Blues_r',
    edgecolor='black'
)

def currency(x, pos):
    if x >= 1e6:
        return '${:,.0f}M'.format(x * 1e-6)
    else:
        return '${:,.0f}K'.format(x * 1e-3)

plt.gca().xaxis.set_major_formatter(FuncFormatter(currency))

plt.title('Total Sales by Product Line',)
plt.xlabel('Sales ($)',)
plt.ylabel('Product Line',)

sns.despine()
plt.tight_layout()
plt.show()
```



- **Key Insights:**
  - This horizontal bar chart shows the total sales by product line, highlighting the best and worst-performing categories.
  - Top Performing Product Lines
    - **Classic Cars** dominate with sales near **$4M**, accounting for a significant portion of total revenue.
    - **Vintage Cars** follow at a distant second with just under **$2M** in sales.
  - Mid-Tier Product Lines
    - **Motorcycles** and **Trucks and Buses** perform similarly, each generating around **$1.3M**.
    - **Planes** also show solid performance with approximately **$1M** in total sales.
  - Low Performing Product Lines
    - **Ships** show moderate performance, falling below $1M.
    - **Trains** contribute the least, with sales below **$500K**.
- Let's explore the relationship between **Unit Price** and **Quantity**

```
# Price vs Quantity Ordered
plt.figure(figsize=(8, 4))
sns.scatterplot(
    data=df,
    x='priceeach',
    y='quantityordered',
```

```
    hue='productline',
    palette='Set1',
    alpha=0.7,
    edgecolor='black',
)


plt.title('Price vs Quantity Ordered', pad=15)
plt.xlabel('Price ($)')
plt.ylabel('Quantity Ordered')

plt.legend(title='Product Line', loc='center right',
bbox_to_anchor=(1.08, 0.0, 0.25, 1.0), fontsize=9)

sns.despine()
plt.tight_layout()
plt.show()
```

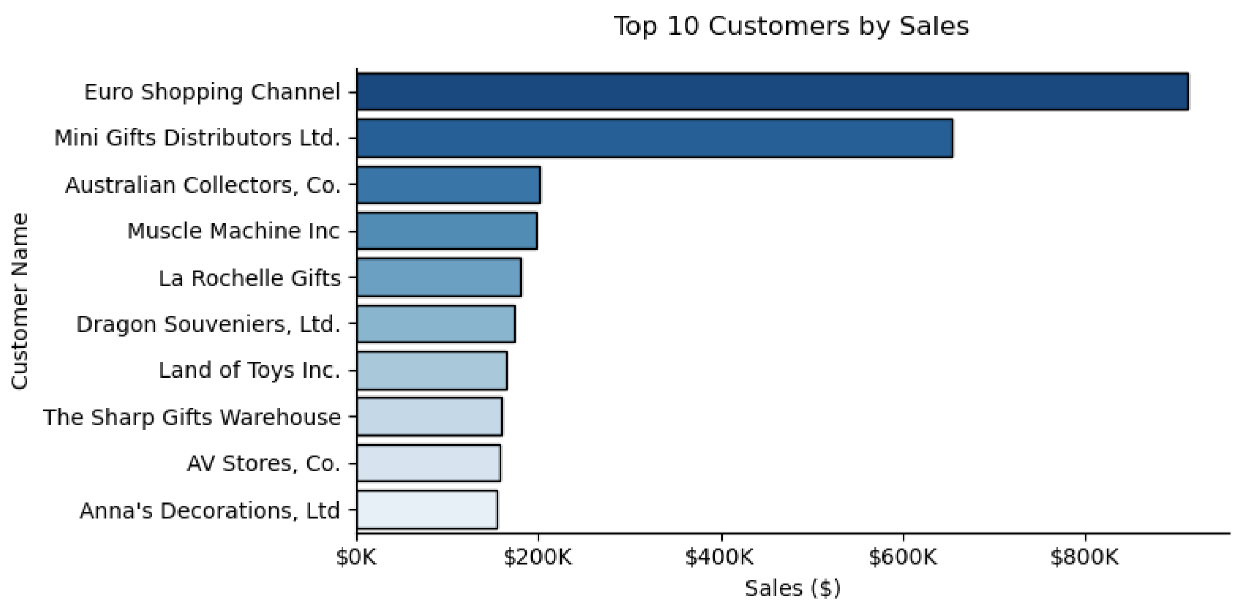
- This scatter plot explores the relationship between **unit price** and **quantity ordered** across various product lines.
- **Observations:**
  - Price Range
    - Prices span from **$30 to $100**, with a noticeable **cluster near the $100 mark**, likely due to premium products (e.g., **Classic Cars**).
  - Quantity Ordered
    - Most orders fall between **20 and 50 units**, regardless of price.
    - A few outliers show quantities exceeding **80–100 units**, particularly around the **$90–$100** range.

- Distribution by Product Line
    - The plot shows **no strong negative correlation** between price and quantity, which suggests:
    - Customers are **still ordering high-priced items** in reasonable quantities.
- Product Line Highlights
    - **Classic Cars** and **Vintage Cars** appear frequently in the high-price range ($90–$100), reinforcing their revenue dominance.
    - **Motorcycles** and **Trucks and Buses** span a wide price range, showing **diverse product offerings**.
    - **Trains** and **Ships** are less dense in the plot, supporting earlier insights on lower sales.

Customer Analysis:

- We'll analyze **Top Customer** by **Sales**

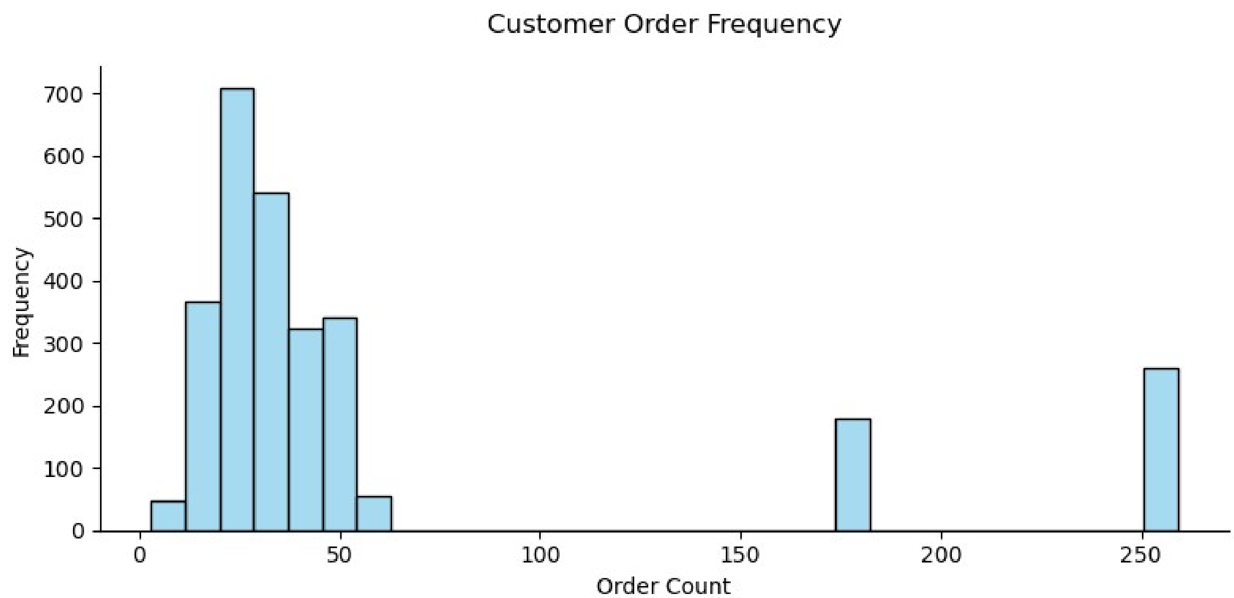
```
# top 10 customer by sales
top_customer = df.groupby('customername')
['sales'].sum().reset_index().sort_values(by='sales',
ascending=False).head(10)

plt.figure(figsize=(8, 4))
sns.barplot(
    data=top_customer,
    x='sales',
    y='customername',
    palette='Blues_r',
    edgecolor='black'
)
def currency(x, pos):
    if x >= 1e6:
        return '${:,.0f}M'.format(x * 1e-6)
    else:
        return '${:,.0f}K'.format(x * 1e-3)

plt.gca().xaxis.set_major_formatter(FuncFormatter(currency))

plt.title('Top 10 Customers by Sales', pad=15)
plt.xlabel('Sales ($)')
plt.ylabel('Customer Name')

sns.despine()
plt.tight_layout()
plt.show()
```



- This horizontal bar chart displays the top 10 customers ranked by total sales volume.
- **Key Insights:**
- Highest Revenue Contributors
  - **Euro Shopping Channel** leads by a wide margin, generating nearly **$900K+** in sales.
  - **Mini Gifts Distributors Ltd.** is second, contributing approximately **$700K+**.
- Sales Drop-Off
  - There’s a significant gap between the **top 2 customers** and the remaining ones.
  - Other top customers such as **Australian Collectors, Co.**, **Muscle Machine Inc**, and **La Rochelle Gifts** cluster around the **$200K mark**.
- Long Tail
  - The bottom 5 of the top 10 still represent key business, each bringing in **$150K–$200K**, suggesting a moderately strong mid-tier customer base.
- Let's see how frequently customers place orders by plotting the distribution of counts per customer

```
# Order Frequency
order_frequency = df['customername'].value_counts().reset_index()
order_frequency.columns = ['customername', 'order_count']
plt.figure(figsize=(8, 4))
```

```
sns.histplot(
    data=order_frequency,
    x='order_count',
    bins=30,
    color='skyblue',
    edgecolor='black',
    weights='order_count',

)

plt.title('Customer Order Frequency', pad=15)
plt.xlabel('Order Count')
plt.ylabel('Frequency')

sns.despine()
plt.tight_layout()
plt.show()
```



- This histogram shows the distribution of customers based on how many orders they've placed.

Key Observations:

- Majority of Customers
  - Most customers placed between **20 and 60 orders**, with the **peak frequency around 30 orders**.
  - These customers make up the **bulk of the customer base**, representing typical purchasing behavior.
- Outliers

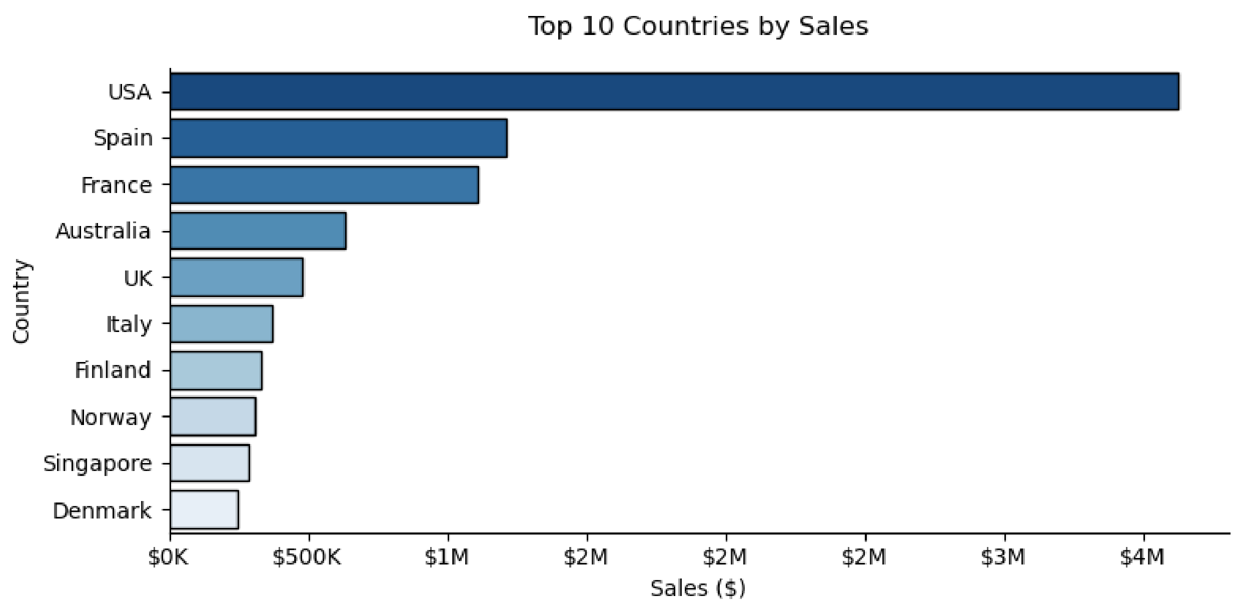
- A small number of customers placed **extremely high numbers of orders**, with clear outliers at:
- **~175 orders**
- **~250 orders**
- These may represent **VIP customers, resellers, or subscription accounts**.

- Distribution Shape
  - The distribution is **right-skewed**, suggesting that while most customers place a moderate number of orders, a few place significantly more.

Geographical Analysis Analysis:

- Let's see Top 10 coutries by Sales

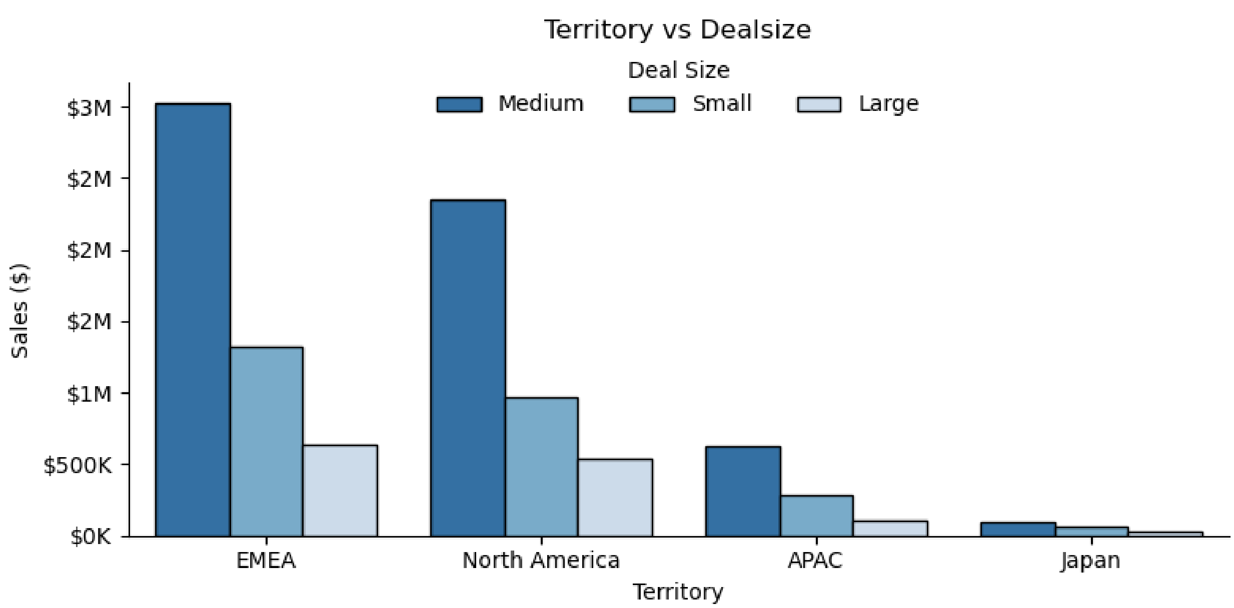
```
# Sales by Country
sales_by_country = df.groupby('country')
['sales'].sum().reset_index().sort_values(by='sales',
ascending=False).head(10)

plt.figure(figsize=(8, 4))
sns.barplot(
    data=sales_by_country,
    x='sales',
    y='country',
    palette='Blues_r',
    edgecolor='black'
)

def currency(x, pos):
    """The two args are the value and tick position"""
    if x >= 1e6:
        return '${:,.0f}M'.format(x * 1e-6)
    else:
        return '${:,.0f}K'.format(x * 1e-3)

plt.gca().xaxis.set_major_formatter(FuncFormatter(currency))

plt.title('Top 10 Countries by Sales', pad=15)
plt.xlabel('Sales ($)')
plt.ylabel('Country')

sns.despine()
plt.tight_layout()
plt.show()
```



- **Key Highlights:**
- Top Contributor
  - **USA** dominates with over **$4M in sales**, far ahead of all other countries.
  - This makes it the **primary market** and a crucial driver of revenue.
- Other Strong Performers
  - **Spain** and **France** follow, each generating more than **$1M** in sales.
  - They represent significant but secondary markets.
- Mid-Tier Markets
  - **Australia** and **UK** show moderate sales volumes (~$500K–$800K).
  - These may benefit from targeted growth strategies.
- Long-Tail Countries
  - Countries like **Italy**, **Finland**, **Norway**, **Singapore**, and **Denmark** show lower sales figures (< $500K).
  - These markets may require localization, awareness campaigns, or partner distribution strategies.
- Now, let's see how sales performance varies across territories based on the deal size categories:

```
# Territory vs Deal Size
territory_dealsize = df.groupby(['territory', 'dealsize'])
['sales'].sum().reset_index()
territory_dealsize.sort_values(by='sales', ascending=False,
```

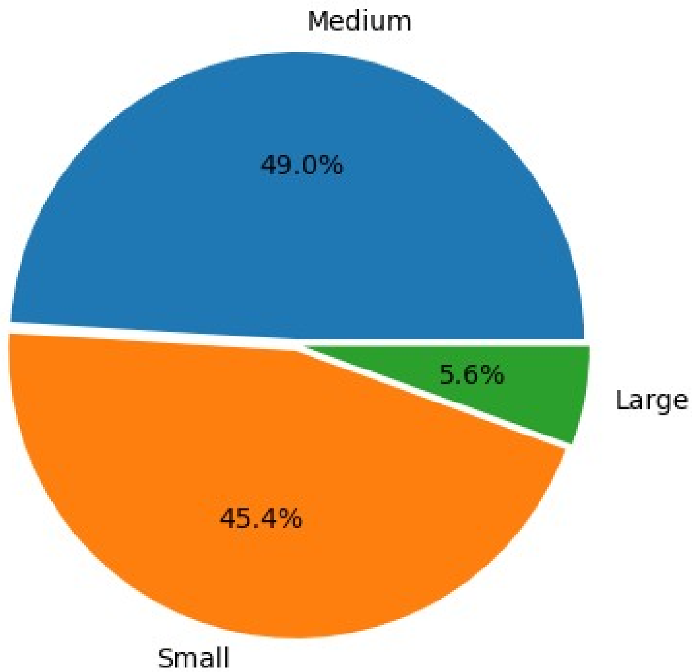
```
inplace=True)

plt.figure(figsize=(8, 4))
sns.barplot(
    data=territory_dealsize,
    x='territory',
    y='sales',
    hue='dealsize',
    palette='Blues_r',
    edgecolor='black'
)

plt.gca().yaxis.set_major_formatter(FuncFormatter(currency)) # the
same currency function as above

plt.title('Territory vs Dealsize', pad=20)
plt.xlabel('Territory')
plt.ylabel('Sales ($)')
plt.legend(title='Deal Size', loc='upper center', ncols=3,
frameon=False, bbox_to_anchor=(0, 0.0, 1, 1.09))

sns.despine()
plt.tight_layout()
plt.show()
```



- **Insights:**
- EMEA (Europe, Middle East & Africa)
  - **Highest overall sales:** Over **$5M** total.
  - **Medium-sized deals** dominate (~$3M), followed by Small and Large.

    - Suggests strong mid-market presence and upsell potential.
- North America
    - Total sales ~**$4M**, with a similar pattern to EMEA.
    - Medium deals lead, followed by Small, then Large.
    - Comparable to EMEA but slightly lower across all deal sizes.
- APAC (Asia-Pacific)
    - Noticeably lower sales (~$1.2M).
    - Still, Medium deals are the largest segment.
    - Potential growth territory; might benefit from more regional focus.
- Japan
    - Very small market footprint (~$100K total).
    - All deal sizes contribute marginally.
    - Requires strategic decisions: invest or maintain minimal presence.

```
# Deal Size Distribution
dealsize_dist = df['dealsize'].value_counts().reset_index()
dealsize_dist.columns = ['status', 'count']


plt.figure(figsize=(8, 4))
plt.pie(
    dealsize_dist['count'],
    labels=dealsize_dist['status'],
    autopct='%1.1f%%',
    explode=(0.02, 0.02, 0.02),

)

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



Deal Size Distribution

This pie chart illustrates the **proportion of total deals** by size category.

- **Medium Deals**: **49.0%**
  → Nearly **half** of all deals fall into this category.
  → Core revenue driver and the most consistent segment.

- **Small Deals**: **45.4%**
  → Also a **significant contributor**, slightly below medium deals.
  → Indicates a healthy volume of transactions, possibly with lower individual value.

- **Large Deals**: **5.6%**
  → A **small but notable share**.
  → These may represent strategic accounts with high individual value but low frequency.

```
# Group by country and calculate average deal size
geo_analysis = df.groupby('country').agg({
    'sales': 'sum',
    'ordernumber': 'nunique',
    'dealsize': lambda x: x.value_counts(normalize=True)['Small']  # %
of small deals
}).reset_index()
```

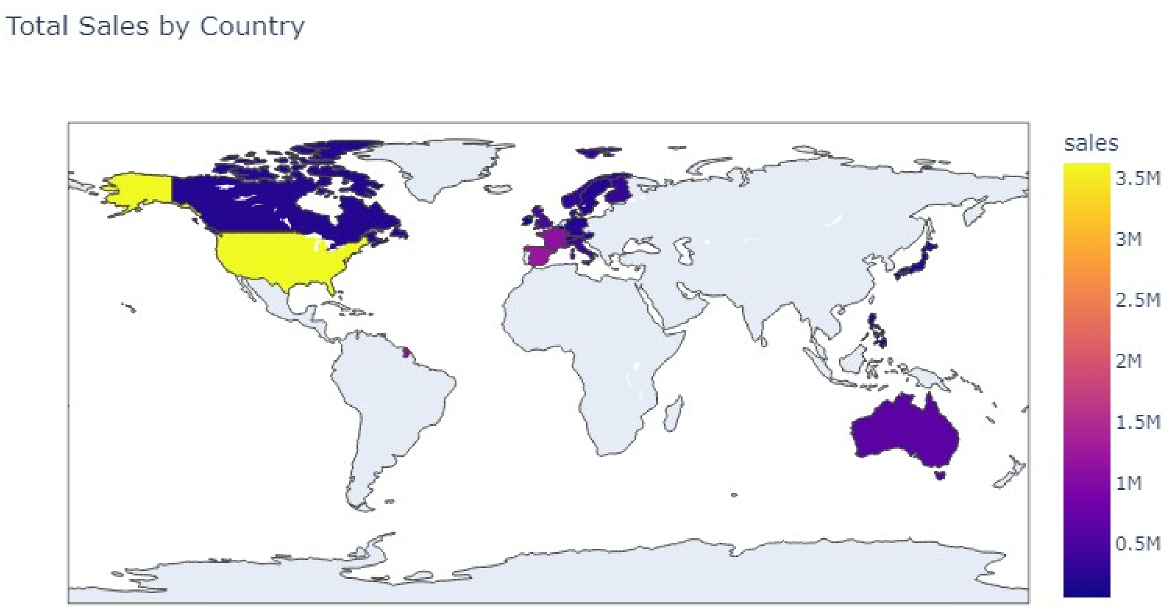
```python
# Visualize using geospatial plots
import plotly.express as px
fig = px.choropleth(
    geo_analysis,
    locations='country',
    locationmode='country names',
    color='sales',
    hover_name='country',
    title='Total Sales by Country',
    width=800,
    height=500,
)
fig.show()
```


```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"coloraxis":"coloraxis","geo":"geo","hovertemplate":"<b>%
{hovertext}</b><br><br>country=%{location}<br>sales=%{z}<extra></
extra>","hovertext":
["Australia","Austria","Belgium","Canada","Denmark","Finland","France"
,"Germany","Ireland","Italy","Japan","Norway","Philippines","Singapore
","Spain","Sweden","Switzerland","UK","USA"],"locationmode":"country
names","locations":
["Australia","Austria","Belgium","Canada","Denmark","Finland","France"
,"Germany","Ireland","Italy","Japan","Norway","Philippines","Singapore
","Spain","Sweden","Switzerland","UK","USA"],"name":"","type":"chorop
eth","z":{"bdata":"MzMzM74+I0HXo3A9dKoIQbgehevJd/
pArkfhenRaC0EzMzMzKfwNQT0K16O3HRRBUrgehYTzMEGF61G4wOkKQSlcj8KNM+xA16Nw
PUneFkGuR+F6PvgGQc3MzMwexBJB4XoUrvvz9kA9CtejoZsRQbgehevGjDJB4XoUrvGiCU
Fcj8L1GL38QHE9CteBOh1BpHA9auetS0E=","dtype":"f8"}}],"layout":
{"coloraxis":{"colorbar":{"title":{"text":"sales"}},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"geo":{"center":
{},"domain":{"x":[0,1],"y":[0,1]}},"height":500,"legend":
{"tracegroupgap":0},"template":{"data":{"bar":[{"error_x":
{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
```

```
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermap":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermap"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
```

```
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
```

```
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"title":
{"text":"Total Sales by Country"},"width":800}}
```

This choropleth map visually represents **total sales distribution** across various countries.

Key Highlights:

- **Top Performing Countries**:
  - **USA**: Highest sales volume, indicated by **bright yellow**.
  - **Spain**: Among the top, with strong sales in **lighter shades**.
  - **France**, **UK**, **Australia**: Also notable contributors.
- **Mid-range Performers**:

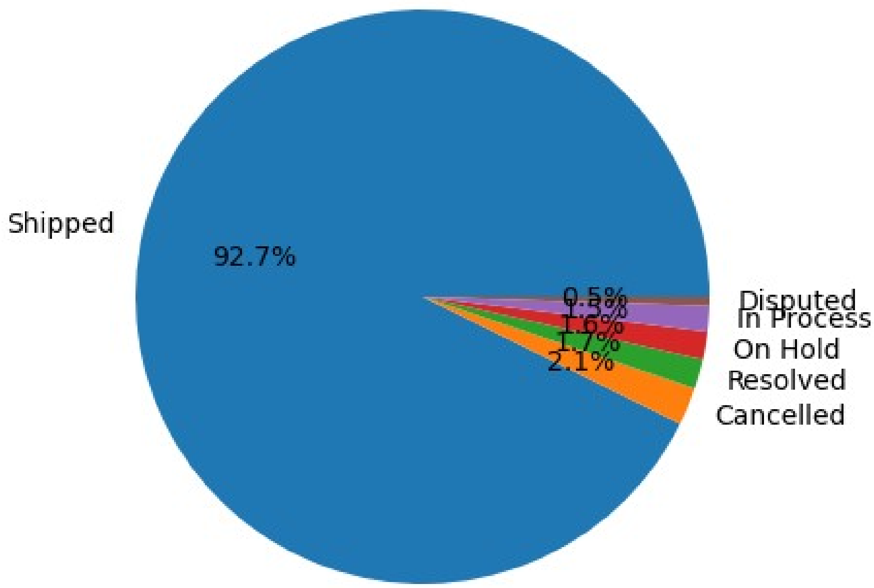
    - **Northern and Western Europe**: Countries like **Finland, Norway, and Italy** show **moderate sales**.

    - 🇨🇦 **Canada**: Noteworthy performance in **North America**.
- **Lower Sales Regions**:
    - Some parts of **Asia** and **South America** show **limited or no activity**, indicating potential markets for future expansion.

Order Status Analysis:

- Now I will use pie chart to show the proportion of each order status relative to the total number of orders. This will help us to understand the current state of orders such as how many are shipped, pending or cancelled.

```
# Status Distribution
status_distribution = df['status'].value_counts().reset_index()
#status_distribution.columns = ['status', 'count']

plt.figure(figsize=(8, 4))
plt.pie(
    status_distribution['count'],
    labels=status_distribution['status'],
    autopct='%1.1f%%',

)

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

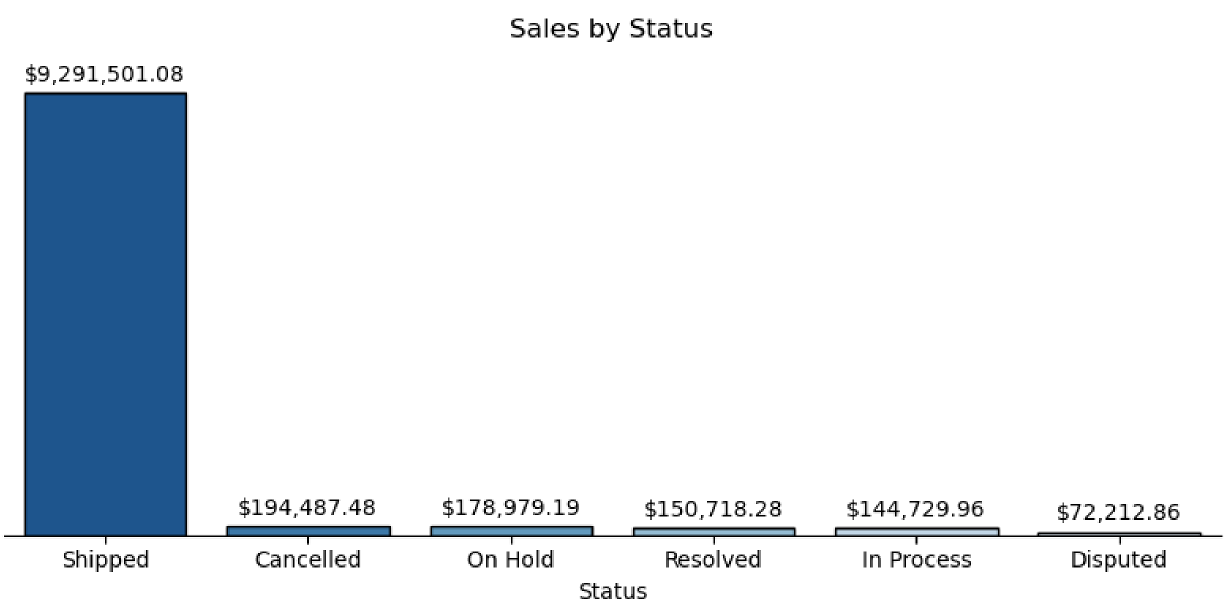


- **Key Findings:**
  - **Shipped**:
    - Dominates the distribution with **92.7%** of all orders.
    - Indicates highly efficient fulfillment and delivery processes.
  - **Other Statuses**:
    - **Cancelled**: ~2.1% — relatively low, suggesting good order accuracy.
    - **Resolved**: ~1.1% — issues addressed effectively.
    - **On Hold**: ~1.0% — potential minor delays or pending issues.
    - **In Process**: ~1.5% — active handling of some orders.
    - **Disputed**: ~0.5% — minimal disputes, reflecting customer satisfaction.

- Now, I want to see the total revenue is actually realized versus how much is tied up in orders that may be at risk of cancellation or dispute.

```
sales_by_status = df.groupby('status')['sales'].sum().reset_index()
sales_by_status = sales_by_status.sort_values(by='sales',
ascending=False)
```

```
plt.figure(figsize=(8, 4))

ax = sns.barplot(
    data=sales_by_status,
    x='status',
    y='sales',
    palette='Blues_r',
    edgecolor='black'
)

#plt.gca().yaxis.set_major_formatter(FuncFormatter(currency))

for container in ax.containers:
    labels = [f'${float(v.get_height()):,}' for v in container]
    ax.bar_label(container, labels=labels, padding=3)


plt.title('Sales by Status', pad=15)
plt.xlabel('Status')
ax.set_yticks([])  # Removes y-ticks
plt.ylabel('')

sns.despine(left=True)
plt.tight_layout()
plt.show()
```



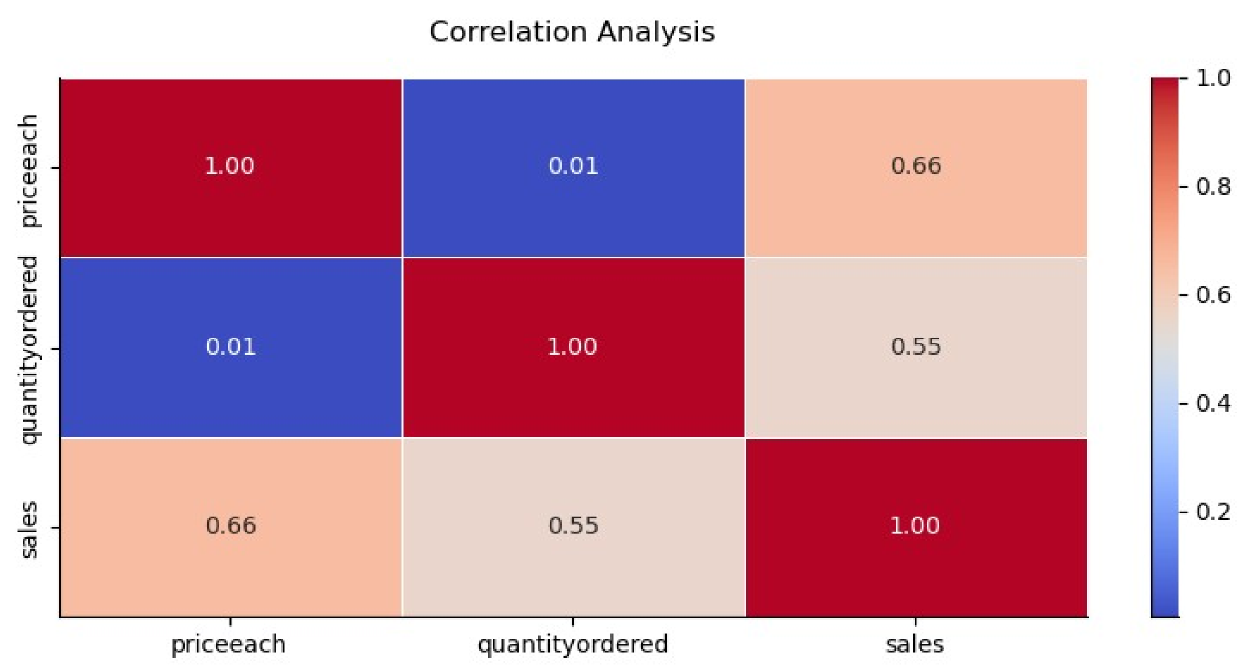
- **Insights:**
  - **Shipped orders dominate revenue**, confirming a **well-functioning sales pipeline**.
  - The relatively **small financial impact** from **Cancelled** and **Disputed** orders suggests **low revenue leakage** due to operational or customer service issues.

- Continued monitoring of **On Hold** and **In Process** orders is advisable to avoid potential delays in revenue recognition.

Correlation:

- Now I will generate a correlation heatmap showing the relationships between the price of each unit, the orders quantity, and sales to see how strongly each pair is related.

```
numerical_columns = ['priceeach', 'quantityordered', 'sales']
plt.figure(figsize=(8, 4))
sns.heatmap(
    data=df[numerical_columns].corr(),
    annot=True,
    cmap='coolwarm',
    linewidths=0.5,
    fmt='.2f',
)

plt.title('Correlation Analysis', pad=15)

sns.despine()
plt.tight_layout()
plt.show()
```



- This heatmap illustrates the **strength and direction of linear relationships** between key sales variables.
- Key Correlation Values:

Variable Pair	Correlation
Price Each ↔ Sales	**0.66** → **moderate positive** relationship. Higher unit prices generally lead to higher total sales.
Quantity Ordered ↔ Sales	**0.55** → **moderate positive** relationship. Larger order quantities contribute to greater sales.
Price Each ↔ Quantity Ordered	**0.01** → **no correlation**. Product price does not significantly influence quantity ordered.

- Insights:
  - **Sales are driven by both price and quantity**, but **price** has a **slightly stronger impact**.
  - The **lack of correlation** between **price** and **quantity ordered** suggests that **purchasing decisions** might be **price inelastic** — customers buy similar quantities regardless of price changes.
  - Marketing strategies could focus on **bundling or volume incentives** to leverage the moderate quantity-sales relationship.
