📊 AI-Powered Sales Analysis Report

Created using Python & AI – No Power BI, No Excel!

```
import pandas as pd

# Try loading with encoding fix
df = pd.read_csv("/kaggle/sales_data_sample.csv", encoding='latin1')

# Preview first few rows
df.head()
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	...	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRITORY
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipped	1	2	2003	...	897 Long Airport Avenue	NaN	NYC	NY	10022	USA	NaN
1	10121	34	81.35	5	2765.90	5/7/2003 0:00	Shipped	2	5	2003	...	59 rue de l'Abbaye	NaN	Reims	NaN	51100	France	EMEA
2	10134	41	94.74	2	3884.34	7/1/2003 0:00	Shipped	3	7	2003	...	27 rue du Colonel Pierre Avia	NaN	Paris	NaN	75508	France	EMEA
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipped	3	8	2003	...	78934 Hillside Dr.	NaN	Pasadena	CA	90003	USA	NaN
4	10159	49	100.00	14	5205.27	10/10/2003 0:00	Shipped	4	10	2003	...	7734 Strong St.	NaN	San Francisco	CA	NaN	USA	NaN

5 rows × 25 columns

📂 Dataset Overview

(A quick look at the sales data used)

```
# Convert ORDERDATE to datetime
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'], errors='coerce')

# Convert SALES to numeric just in case
df['SALES'] = pd.to_numeric(df['SALES'], errors='coerce')

# Drop rows with missing ORDERDATE or SALES
df.dropna(subset=['ORDERDATE', 'SALES'], inplace=True)

# Confirm the cleaned data
print("✅ Cleaned data shape:", df.shape)
df.info()
```

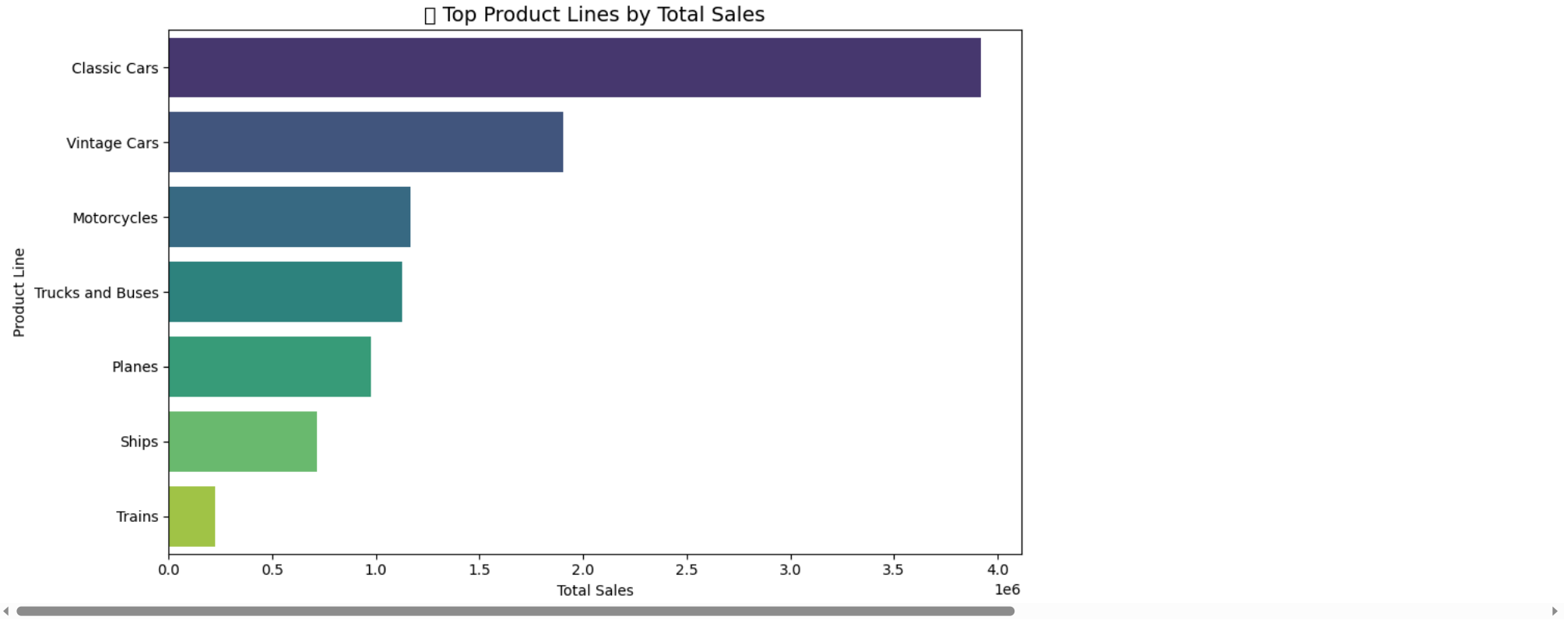
```
✅ Cleaned data shape: (2823, 25)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ORDERNUMBER       2823 non-null   int64
 1   QUANTITYORDERED   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   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  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: datetime64[ns](1), float64(2), int64(7), object(15)
memory usage: 551.5+ KB
```

📅 Monthly Sales Trend

(Line chart showing monthly performance)

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

# Aggregate total sales by PRODUCTLINE
top_products = df.groupby('PRODUCTLINE')['SALES'].sum().sort_values(ascending=False)

# Plot
plt.figure(figsize=(10,6))
sns.barplot(x=top_products.values, y=top_products.index, palette='viridis')
plt.title("💼 Top Product Lines by Total Sales", fontsize=14)
plt.xlabel("Total Sales")
plt.ylabel("Product Line")
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-27-227311645.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=top_products.values, y=top_products.index, palette='viridis')
/tmp/ipython-input-27-227311645.py:13: UserWarning: Glyph 128188 (\N{BRIEFCASE}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 128188 (\N{BRIEFCASE}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



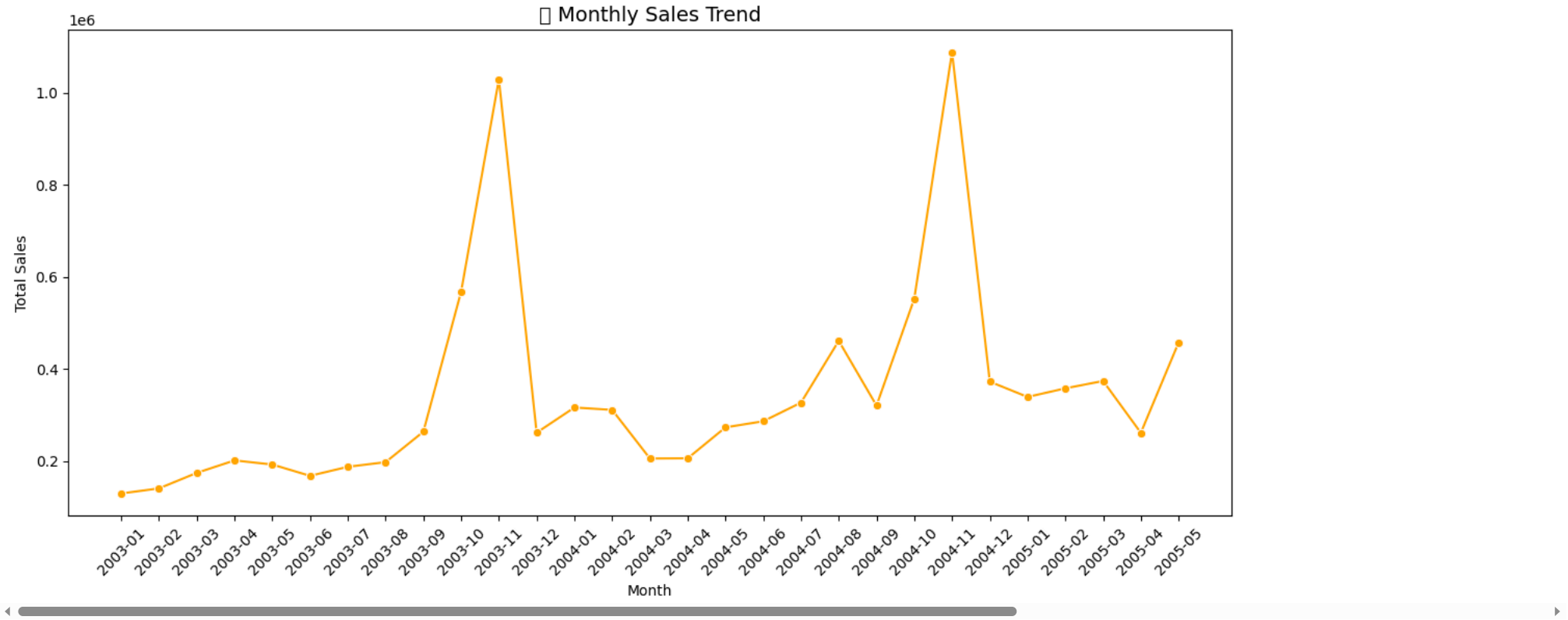
Country-wise Sales Distribution

(Pie chart showing top regions by sales)

```
# Extract Year-Month for grouping
df['YEAR_MONTH'] = df['ORDERDATE'].dt.to_period('M')

# Group by Year-Month and sum SALES
monthly_sales = df.groupby('YEAR_MONTH')['SALES'].sum().reset_index()
monthly_sales['YEAR_MONTH'] = monthly_sales['YEAR_MONTH'].astype(str)  # Convert for plotting

# Plot
plt.figure(figsize=(12,6))
```

```
sns.lineplot(x='YEAR_MONTH', y='SALES', data=monthly_sales, marker='o', color='orange')
plt.xticks(rotation=45)
plt.title("🗓️ Monthly Sales Trend", fontsize=14)
plt.xlabel("Month")
plt.ylabel("Total Sales")
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-38-384848279.py:15: UserWarning: Glyph 128198 (\N{TEAR-OFF CALENDAR}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 128198 (\N{TEAR-OFF CALENDAR}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



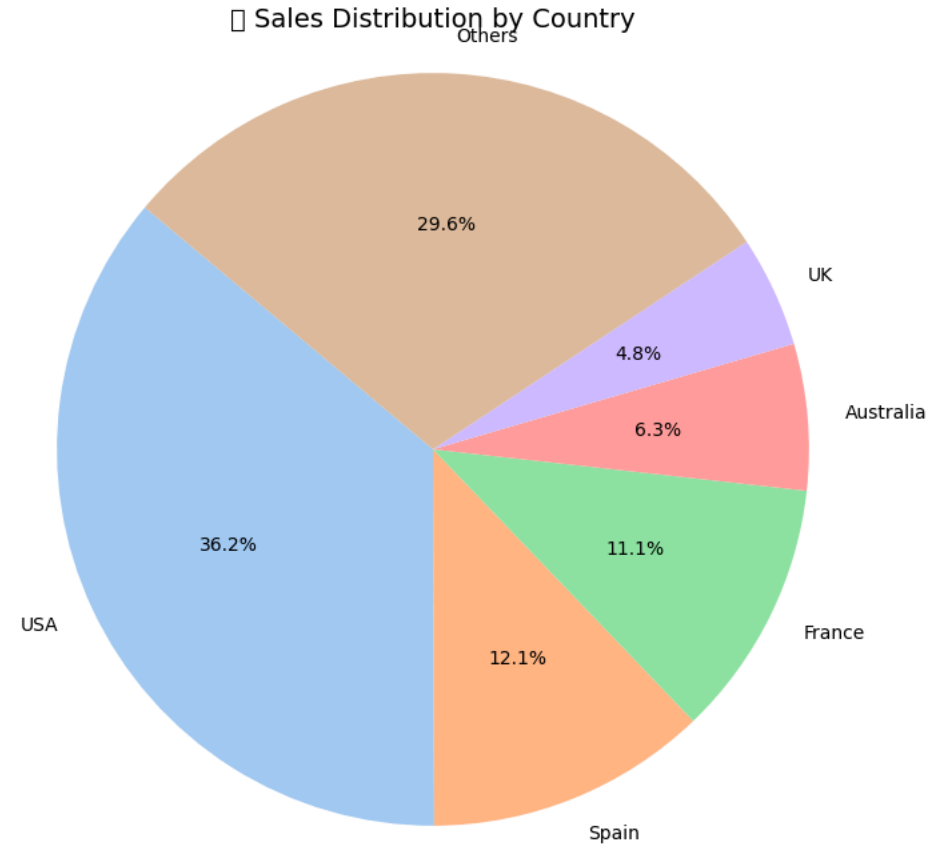
Deal Size Analysis

(Sales split by Small, Medium, Large deals)

```
# Group sales by COUNTRY
country_sales = df.groupby('COUNTRY')['SALES'].sum().sort_values(ascending=False)

# Separate top 5 and group the rest as 'Others'
top_countries = country_sales.head(5)
others_total = country_sales[5:].sum()
```

```
# Combine using pd.concat
final_data = pd.concat([top_countries, pd.Series({'Others': others_total})])

# Plot
plt.figure(figsize=(8,8))
plt.pie(final_data, labels=final_data.index, autopct='%1.1f%%', startangle=140, colors=sns.color_palette('pastel'))
plt.title("🌍 Sales Distribution by Country", fontsize=14)
plt.axis('equal')  # Keeps it a circle
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 127757 (\N{EARTH GLOBE EUROPE-AFRICA}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



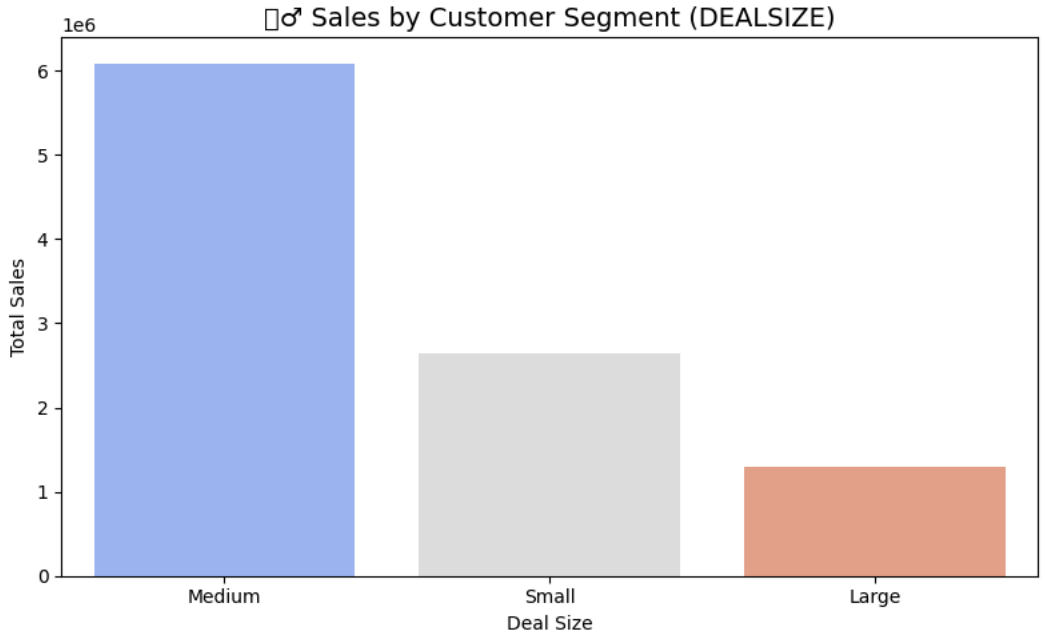
🧍 Deal Size Analysis

(Sales split by Small, Medium, Large deals)

```
# Group by DEALSIZE and sum SALES
```

```
# Group by DEALSIZE and sum SALES
segment_sales = df.groupby('DEALSIZE')['SALES'].sum().sort_values(ascending=False)

# Plot
plt.figure(figsize=(8,5))
sns.barplot(x=segment_sales.index, y=segment_sales.values, palette='coolwarm')
plt.title("🧍 Sales by Customer Segment (DEALSIZE)", fontsize=14)
plt.xlabel("Deal Size")
plt.ylabel("Total Sales")
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-30-1109330637.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=segment_sales.index, y=segment_sales.values, palette='coolwarm')
/tmp/ipython-input-30-1109330637.py:10: UserWarning: Glyph 129485 (\N{STANDING PERSON}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 129485 (\N{STANDING PERSON}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



🧠 AI-Generated Insights

(Smart summary generated using logic/AI)

```
# Calculate key stats from monthly sales
peak_month = monthly_sales.loc[monthly_sales['SALES'].idxmax(), 'YEAR_MONTH']
```

```
peak_month = monthly_sales.loc[monthly_sales['SALES'].idxmax(), 'YEAR_MONTH']
low_month = monthly_sales.loc[monthly_sales['SALES'].idxmin(), 'YEAR_MONTH']
avg_sales = monthly_sales['SALES'].mean()

# Generate summary text
print("🧠 AI Insight Summary:")
print(f"- 📈 Sales peaked in {peak_month}, showing highest revenue during that period.")
print(f"- 📉 The lowest sales were recorded in {low_month}.")
print(f"- 📊 Average monthly sales were approximately ${avg_sales:,.2f}.")
print("- 💡 Consider investigating what drove performance in peak vs. low months.")
```

```
🧠 AI Insight Summary:
- 📈 Sales peaked in 2004-11, showing highest revenue during that period.
- 📉 The lowest sales were recorded in 2003-01.
- 📊 Average monthly sales were approximately $345,952.72.
- 💡 Consider investigating what drove performance in peak vs. low months.
```