

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv('/content/drive/MyDrive/MLProject1/sales_data_sample.csv', encoding='unicode_escape')
```

df.head()

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	...	ADDRESSL
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipped	1	2	2003	...	897 Airport Av
1	10121	34	81.35	5	2765.90	5/7/2003 0:00	Shipped	2	5	2003	...	59 n l'At
2	10134	41	94.74	2	3884.34	7/1/2003 0:00	Shipped	3	7	2003	...	27 n Colonel F
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipped	3	8	2003	...	78934 Hi
4	10159	49	100.00	14	5205.27	10/10/2003 0:00	Shipped	4	10	2003	...	7734 S

5 rows × 25 columns

df.dtypes

	0
ORDERNUMBER	int64
QUANTITYORDERED	int64
PRICEEACH	float64
ORDERLINENUMBER	int64
SALES	float64
ORDERDATE	object
STATUS	object
QTR_ID	int64
MONTH_ID	int64
YEAR_ID	int64
PRODUCTLINE	object
MSRP	int64
PRODUCTCODE	object
CUSTOMERNAME	object
PHONE	object
ADDRESSLINE1	object
ADDRESSLINE2	object
CITY	object
STATE	object
POSTALCODE	object
COUNTRY	object
TERRITORY	object
CONTACTLASTNAME	object
CONTACTFIRSTNAME	object
DEALSIZE	object

dtype: object

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')
```

df.isnull().sum()

	0
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

dtype: int64

```
#Deleting columns that have 30% missing values
missing_percentage = df.isnull().mean() * 100
threshold = 30
```

```
# Drop the columns
columns_to_drop = missing_percentage[missing_percentage >
threshold].index
```

```
# Drop the columns
df = df.drop(columns=columns_to_drop)
```

```
# Display the resulting dataframe after removing columns with too many missing values
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")
```

```
Columns dropped: []
Remaining columns: Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER',
'SALES', 'ORDERDATE', 'STATUS', 'QTR_ID', 'MONTH_ID', 'YEAR_ID',
'PRODUCTLINE', 'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE',
'ADDRESSLINE1', 'CITY', 'POSTALCODE', 'COUNTRY', 'CONTACTLASTNAME',
'CONTACTFIRSTNAME', 'DEALSIZE', 'HighSales'],
dtype='object')
```

```
#Pre-processing on dataset
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
```

```
df[column] = label_encoder.fit_transform(df[column])
df.head()
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	...	CUSTOMERNAME
0	10107	30	95.70	2	2871.00	112	5	1	2	2003	...	
1	10121	34	81.35	5	2765.90	185	5	2	5	2003	...	
2	10134	41	94.74	2	3884.34	204	5	3	7	2003	...	
3	10145	45	83.26	6	3746.70	226	5	3	8	2003	...	
4	10159	49	100.00	14	5205.27	23	5	4	10	2003	...	

5 rows × 23 columns

```
for column in df.select_dtypes(include=['float64']).columns: mean_value = df[column].mean()
```

```
df[column] = df[column].fillna(mean_value)
```

```
for column in df.select_dtypes(include=['object']).columns: mode_value = df[column].mode()[0]
```

```
mode_value = df[column].mode()[0]
```

```
df[column] = df[column].fillna(mode_value)
```

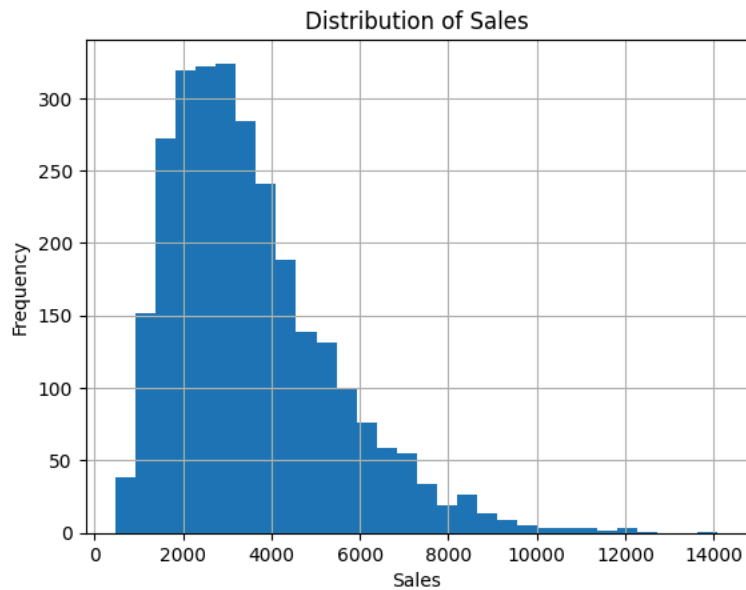
```
df.isnull().sum()
```

	0
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
CITY	0
POSTALCODE	0
COUNTRY	0
CONTACTLASTNAME	0
CONTACTFIRSTNAME	0
DEALSIZE	0

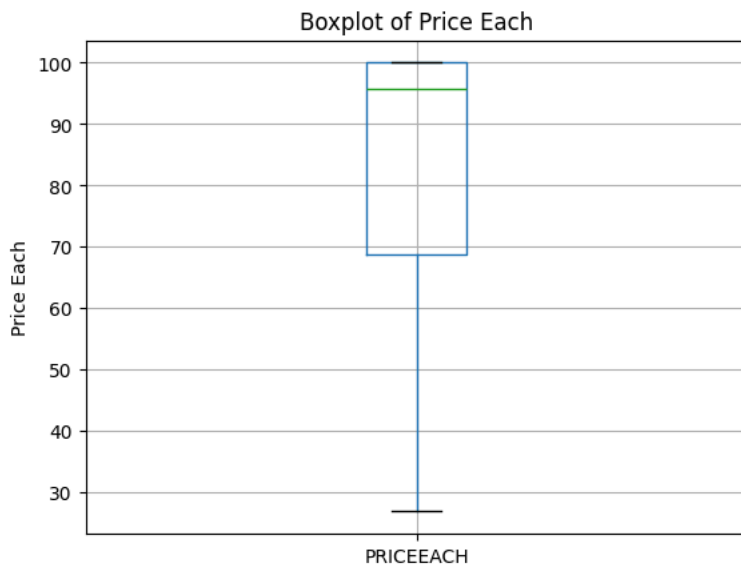
dtype: int64

```
# Visualizations
# Histogram of Sales
plt.figure()
```

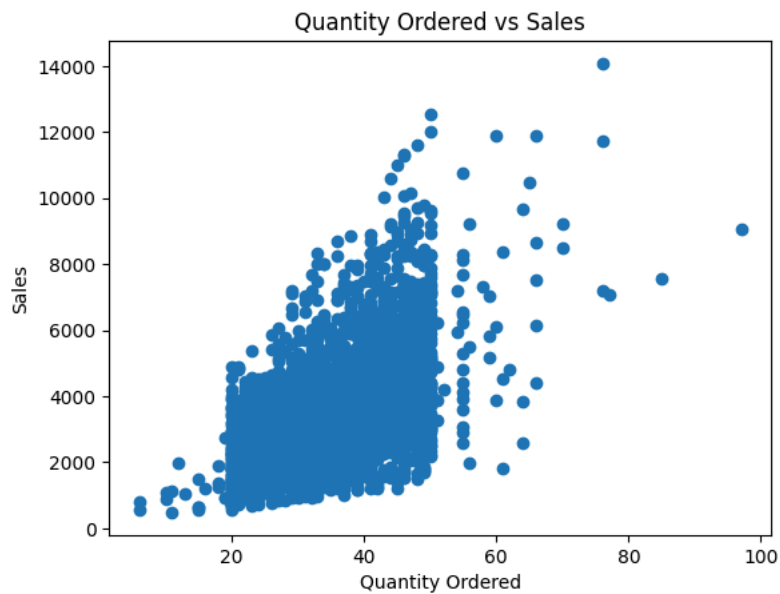
```
df['SALES'].hist(bins=30)
plt.title('Distribution of Sales')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



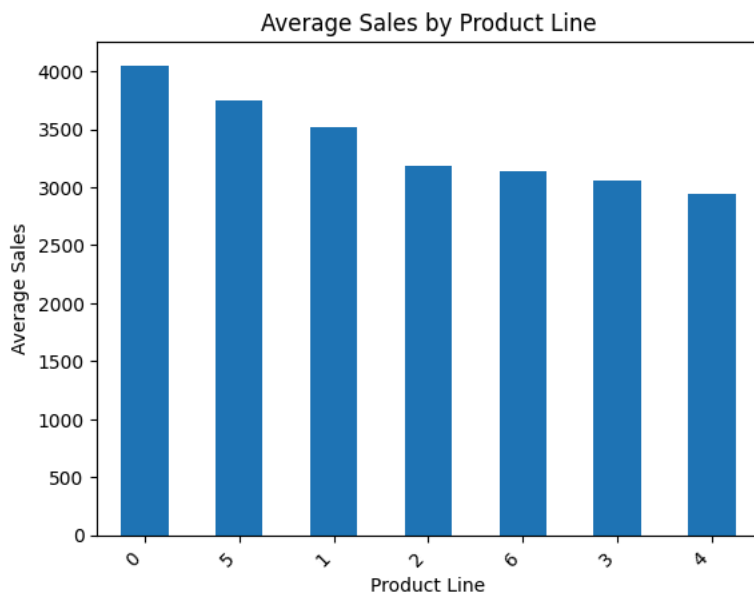
```
# Boxplot of Price Each
plt.figure()
df.boxplot(column='PRICEEACH')
plt.title('Boxplot of Price Each')
plt.ylabel('Price Each')
plt.show()
```



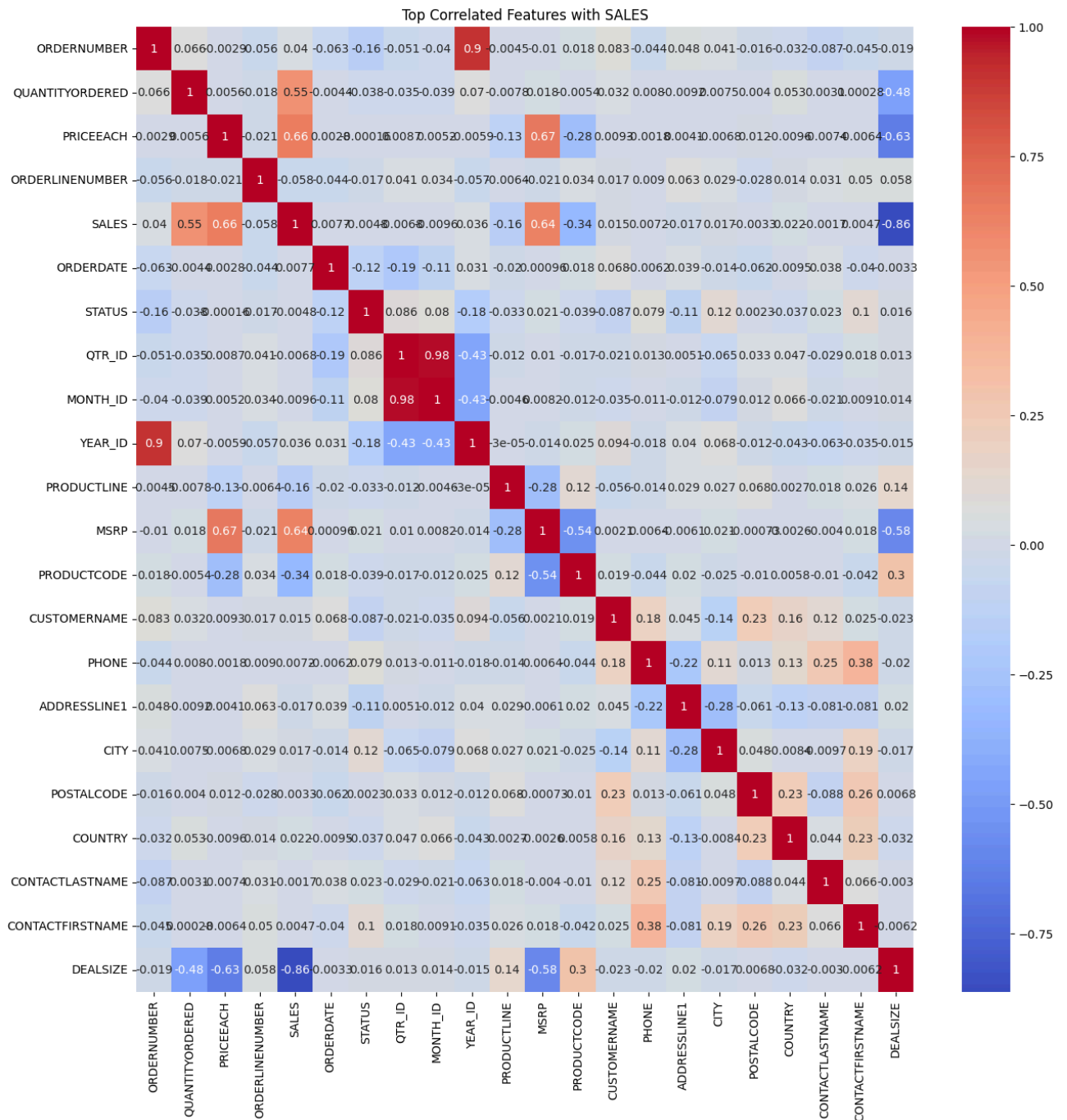
```
# Scatter plot Quantity Ordered vs Sales
plt.figure()
plt.scatter(df['QUANTITYORDERED'], df['SALES'])
plt.title('Quantity Ordered vs Sales')
plt.xlabel('Quantity Ordered')
plt.ylabel('Sales')
plt.show()
```



```
# Bar chart of average sales by product line
avg_sales = df.groupby('PRODUCTLINE')['SALES'].mean().sort_values(ascending=False)
plt.figure()
avg_sales.plot(kind='bar')
plt.title('Average Sales by Product Line')
plt.xlabel('Product Line')
plt.ylabel('Average Sales')
plt.xticks(rotation=45, ha='right')
plt.show()
```



```
plt.figure(figsize=(15, 15))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Top Correlated Features with SALES')
plt.show()
```



```
df['COUNTRY'].value_counts()
```

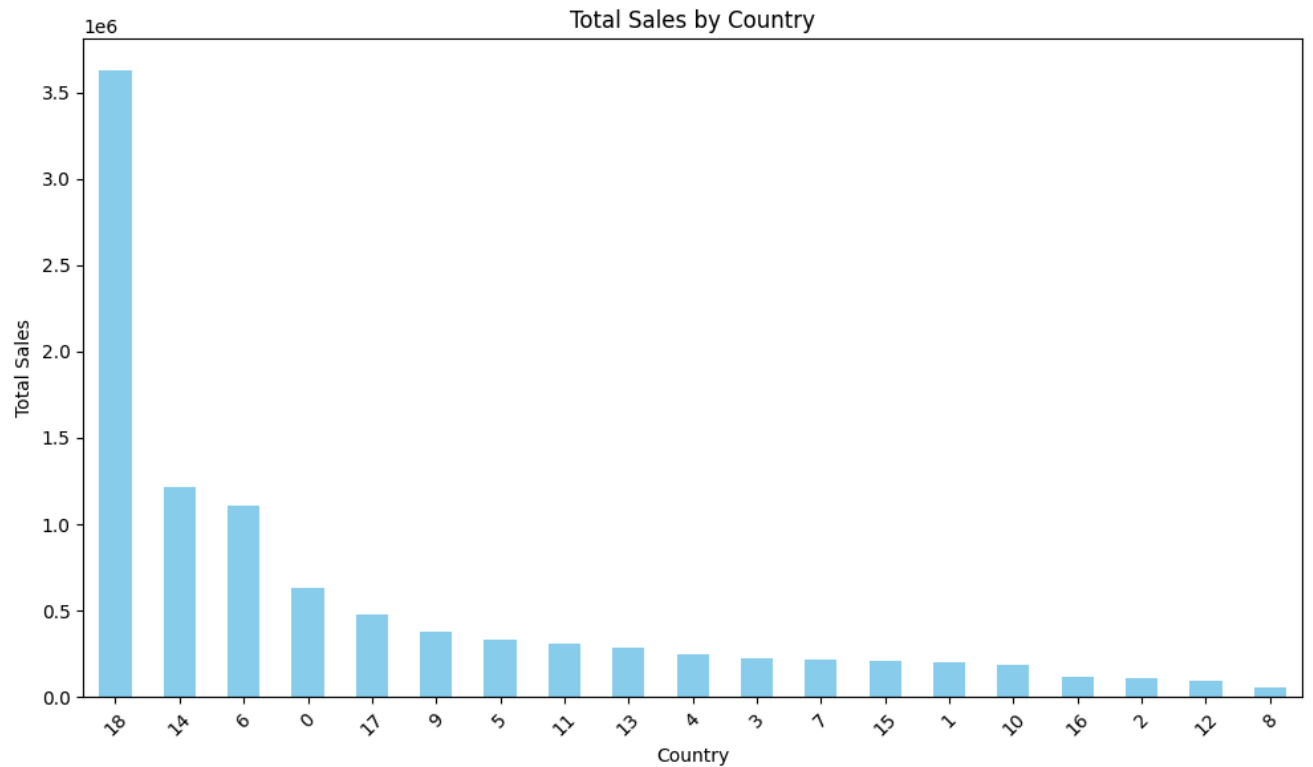
	count
COUNTRY	
18	1004
14	342
6	314
0	185
17	144
9	113
5	92
11	85
13	79
3	70
4	63
7	62
15	57
1	55
10	52
2	33
16	31
12	26
8	16

dtype: int64

```
sales_by_country = df.groupby('COUNTRY')['SALES'].sum().sort_values(ascending=False)

plt.figure(figsize=(10, 6))
sales_by_country.plot(kind='bar', color='skyblue')

# Add chart details
plt.title('Total Sales by Country')
plt.xlabel('Country')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```
avg_sales_country = df.groupby('COUNTRY')['SALES'].mean().sort_values(ascending=False)
```

```
# Plot
```

```
plt.figure(figsize=(10, 6))
```

```
avg_sales_country.plot(kind='bar', color='black')
```

```
plt.title('Average Sales by Country')
```

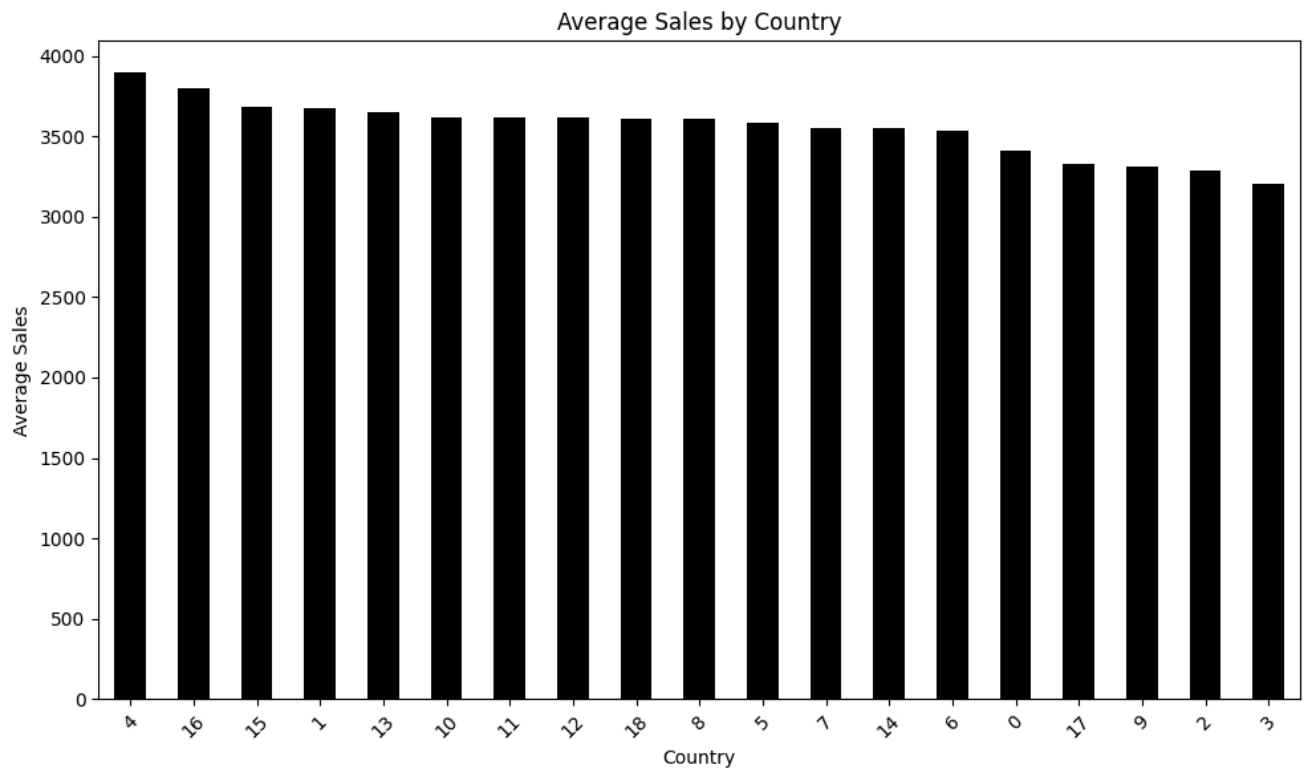
```
plt.xlabel('Country')
```

```
plt.ylabel('Average Sales')
```

```
plt.xticks(rotation=45)
```

```
plt.tight_layout()
```

```
plt.show()
```



```
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns.tolist()

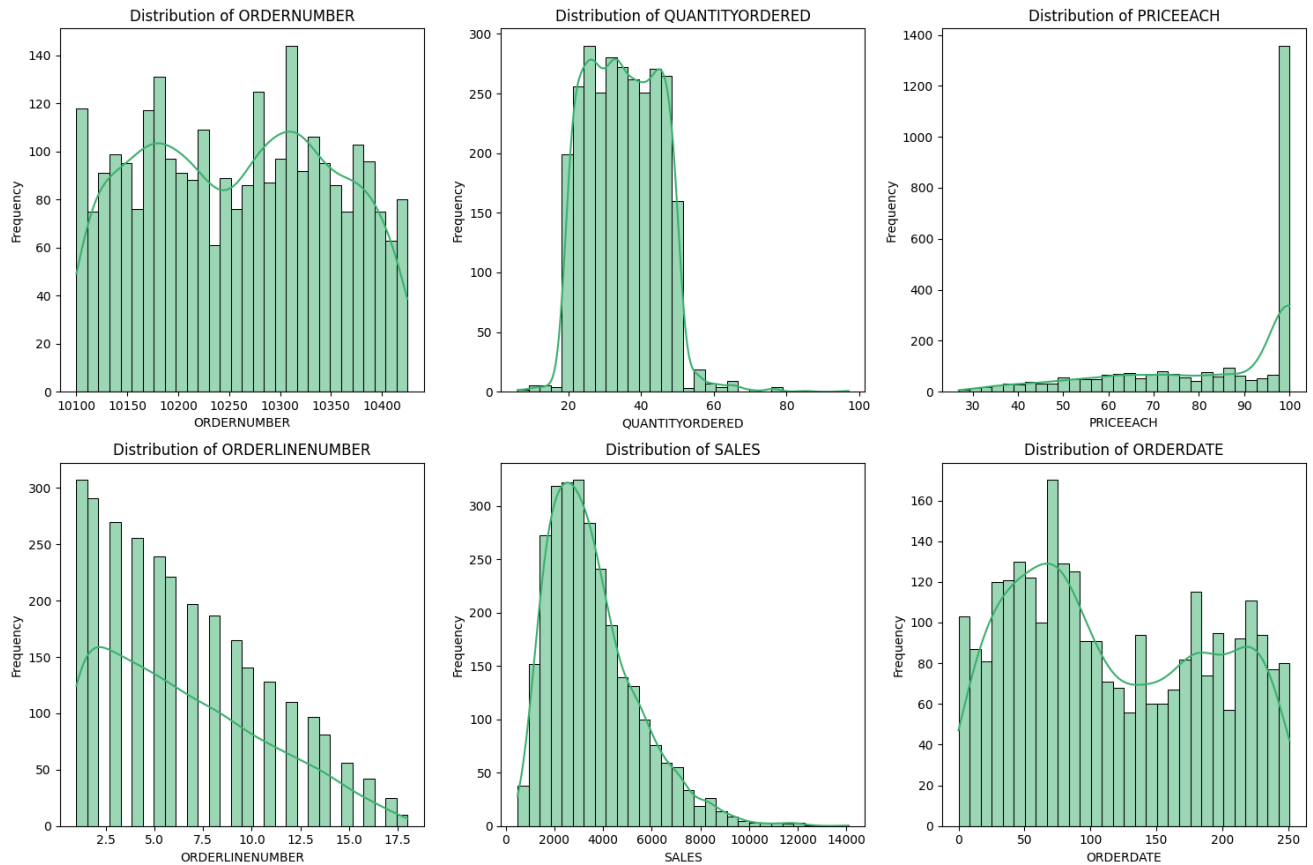
print("Numerical columns detected:", numerical_cols)

# Plot histograms
plt.figure(figsize=(15, 10))

for i, col in enumerate(numerical_cols[:6]): # Display up to 6 columns in a 2x3 grid
    plt.subplot(2, 3, i + 1)
    sns.histplot(df[col], kde=True, bins=30, color='mediumseagreen')
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')

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

Numerical columns detected: ['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER', 'SALES', 'ORDERDATE', 'STATUS', '']



```
# Ensure there's a SALES column
if 'SALES' not in df.columns:
    raise ValueError("No 'SALES' column found in sales_data_sample.csv. Please provide a SALES column.")

# Create a binary target: HighSales (1) if SALES > median, else 0
median_sales = df['SALES'].median()
df['HighSales'] = (df['SALES'] > median_sales).astype(int)

# Detect categorical columns (non-numeric, or small-number ints)
categorical_cols = df.select_dtypes(include=['object', 'category']).columns.tolist()

# also include integer columns with low cardinality (likely categorical codes)
int_cols = df.select_dtypes(include=['int64']).columns.tolist()
for c in int_cols:
    if df[c].nunique() <= 10 and c not in categorical_cols and c != 'HighSales':
        categorical_cols.append(c)

# Limit to first 9 to fit a 3x3 grid (adjust as needed)
categorical_cols = categorical_cols[:9]

print("Categorical columns used for plotting:", categorical_cols)

# Setup subplots (3x3 grid)
n = len(categorical_cols)
nrows, ncols = 3, 3
fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(18, 12))
axes = axes.flatten()

for i, col in enumerate(categorical_cols):
    ax = axes[i]

    # Compute normalized crosstab: proportion of HighSales (1) and LowSales (0) per category
    ct = pd.crosstab(df[col].fillna('Missing'), df['HighSales'], normalize='index')

    # Reindex columns to ensure both 0 and 1 exist
```

```

for val in [0, 1]:
    if val not in ct.columns:
        ct[val] = 0.0
ct = ct[[0, 1]] # order: LowSales then HighSales

# Plot stacked bar chart showing proportions per category
ct.plot(kind='bar', stacked=True, ax=ax, width=0.8)

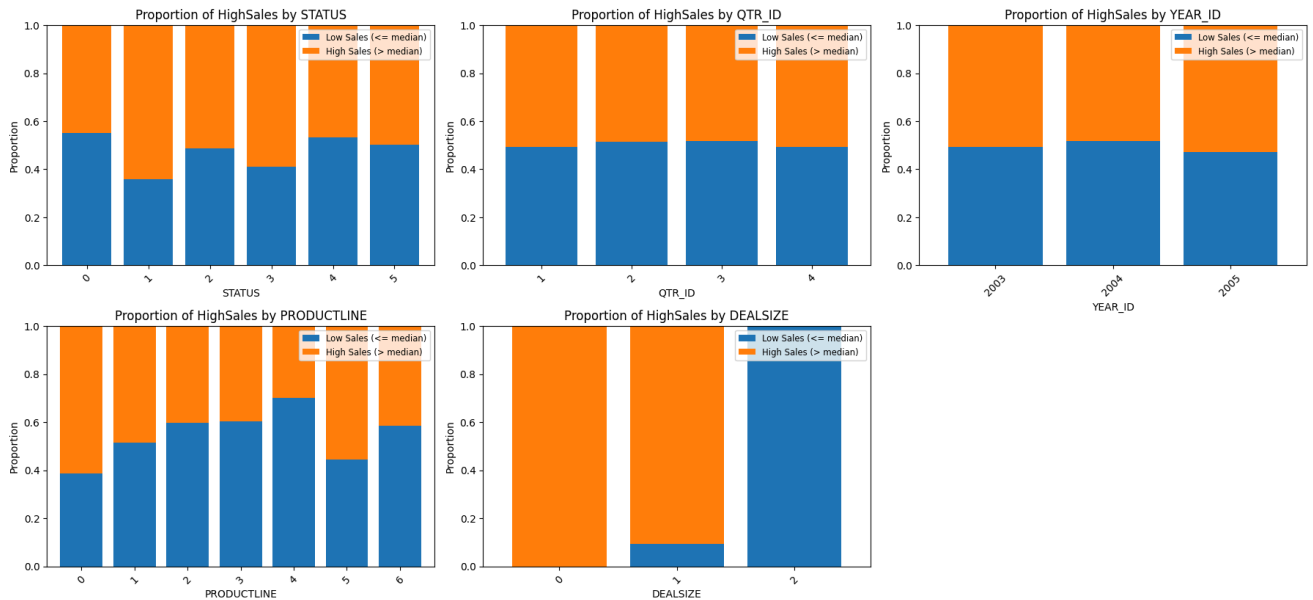
ax.set_title(f'Proportion of HighSales by {col}')
ax.set_xlabel(col)
ax.set_ylabel('Proportion')
ax.legend(['Low Sales (<= median)', 'High Sales (> median)'], loc='upper right', fontsize='small')
ax.set_ylim(0, 1)
ax.tick_params(axis='x', rotation=45)

# Remove unused axes if fewer than grid size
for j in range(n, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.suptitle('Categorical Features vs High Sales (stacked proportions)', fontsize=16, y=1.02)
plt.show()

```

Categorical columns used for plotting: ['STATUS', 'QTR_ID', 'YEAR_ID', 'PRODUCTLINE', 'DEALSIZE']
Categorical Features vs High Sales (stacked proportions)



```

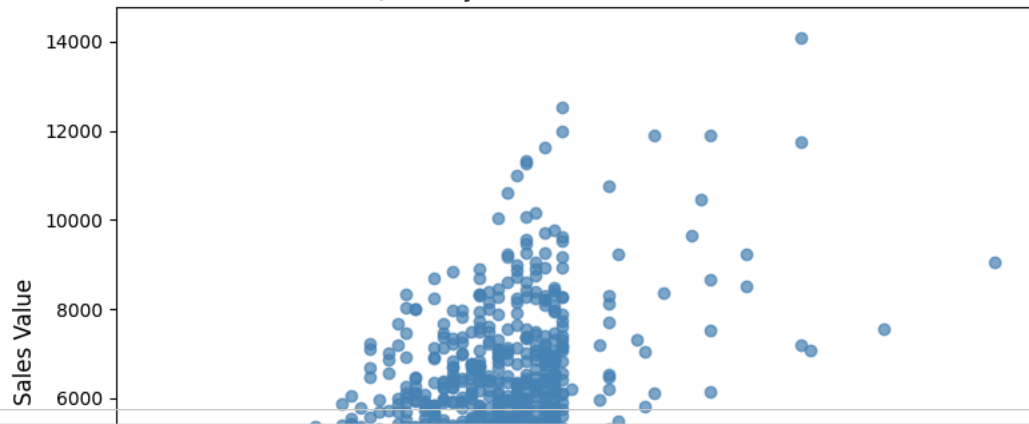
# Create scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(df["QUANTITYORDERED"], df["SALES"], color='steelblue', alpha=0.7)

plt.title("Quantity Ordered vs Sales Value", fontsize=14)
plt.xlabel("Quantity Ordered", fontsize=12)
plt.ylabel("Sales Value", fontsize=12)

plt.tight_layout()
plt.show()

```

Quantity Ordered vs Sales Value



```
# Create boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(x='DEALSIZE', y='SALES', data=df, palette='Set2')

plt.title("Sales Distribution by Deal Size", fontsize=14)
plt.xlabel("Deal Size", fontsize=12)
plt.ylabel("Sales Value", fontsize=12)
plt.tight_layout()
plt.show()
```