Exploratory Data Analysis (EDA) on Retail Sales Data

In this project, we will work with a dataset containing information about retail sales. The goal is to perform exploratory data analysis (EDA) to uncover patterns, trends, and insights that can help the retail business make informed decisions.

Import Library

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

Import Dataset

```
retail=pd.read_csv('/content/drive/MyDrive/kaggle_API/retail_sales_dataset.csv')
```

Will Display the first 10 rows

```
retail.head(10)
```

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
0	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
1	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
3	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
4	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
5	6	2023-	CUST006	Female	45	Beauty	1	30	30

Next steps: Generate code with retail View recommended plots

Checking for missing values if any

```
retail.isnull().sum()
```

```
Transaction ID      0
Date                0
Customer ID         0
Gender              0
Age                 0
Product Category    0
Quantity            0
Price per Unit      0
Total Amount        0
dtype: int64
```

For correct data types we'll use:

```
retail['Date']=pd.to_datetime(retail['Date'])
```

Remove any duplicates

```
retail.drop_duplicates(inplace=True)
```

```
retail.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Transaction ID    1000 non-null   int64
 1   Date              1000 non-null   datetime64[ns]
 2   Customer ID       1000 non-null   object
 3   Gender            1000 non-null   object
 4   Age               1000 non-null   int64
 5   Product Category  1000 non-null   object
 6   Quantity          1000 non-null   int64
 7   Price per Unit    1000 non-null   int64
 8   Total Amount      1000 non-null   int64
dtypes: datetime64[ns](1), int64(5), object(3)
memory usage: 70.4+ KB
```

```
retail.columns
```

```
Index(['Transaction ID', 'Date', 'Customer ID', 'Gender', 'Age',
       'Product Category', 'Quantity', 'Price per Unit', 'Total Amount'],
      dtype='object')
```

```
retail['Sales']=np.random.randint(low=100,high=1000,size=len(retail))
```

Now Descriptive Statistics

Calculating basic statistics:

```
retail.describe()
```

	Transaction ID	Date	Age	Quantity	Price per Unit	Total Amount	Sales
count	1000.000000	1000	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	2023-07-03 00:25:55.200000256	41.39200	2.514000	179.890000	456.000000	548.120000
min	1.000000	2023-01-01 00:00:00	18.00000	1.000000	25.000000	25.000000	100.000000
25%	250.750000	2023-04-08 00:00:00	29.00000	1.000000	30.000000	60.000000	329.750000
50%	500.500000	2023-06-29 12:00:00	42.00000	3.000000	50.000000	135.000000	542.000000
75%	750.250000	2023-10-04 00:00:00	53.00000	4.000000	300.000000	900.000000	769.750000
max	1000.000000	2024-01-01 00:00:00	64.00000	4.000000	500.000000	2000.000000	999.000000
std	288.819436	NaN	13.68143	1.132734	189.681356	559.997632	259.702963

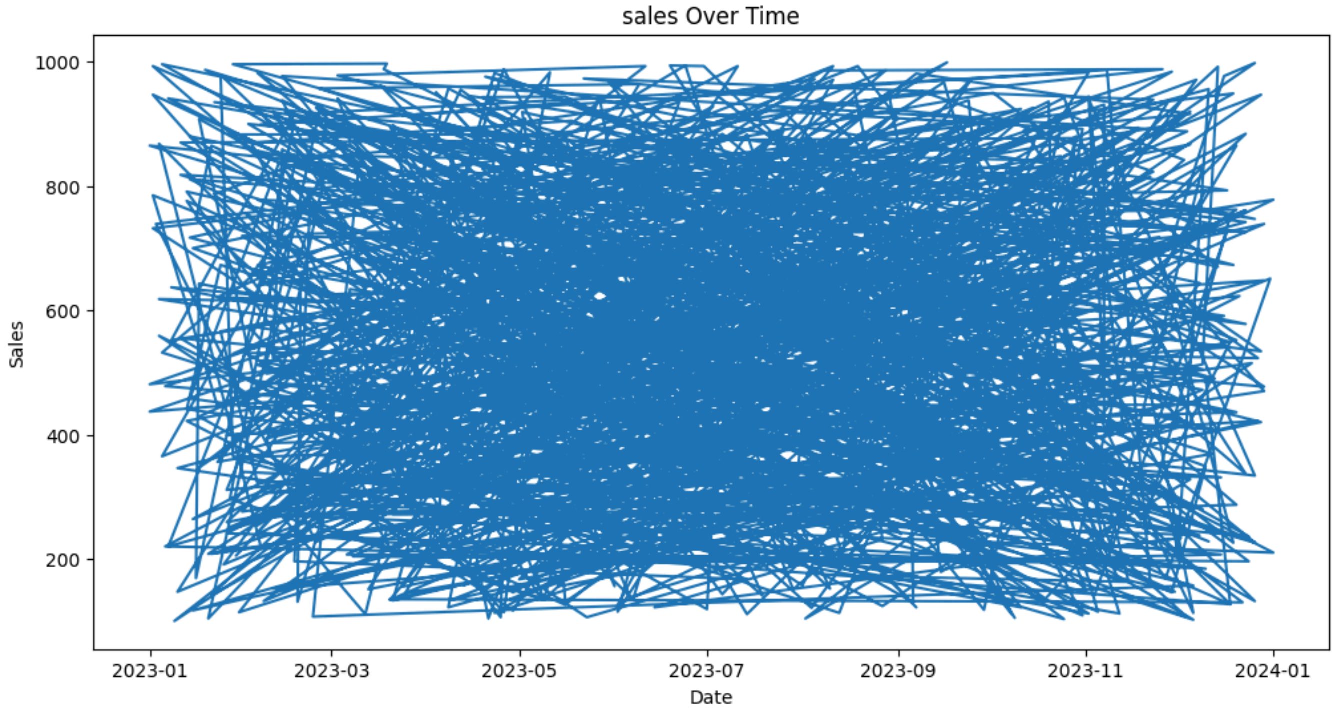
Time Series Analysis

will set the date column as the index

```
retail.set_index('Date',inplace=True)
```

We'll plot the sales over time

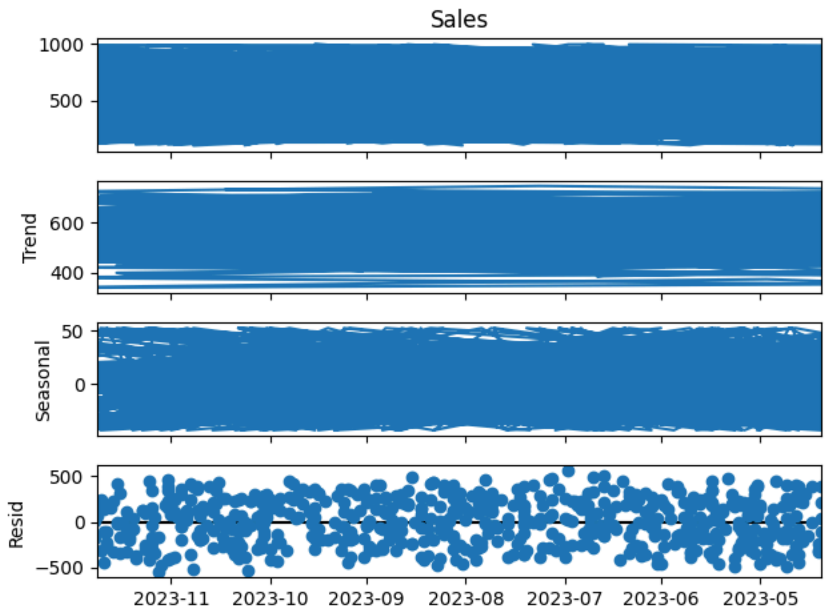
```
plt.figure(figsize=(12,6))
plt.plot(retail.index,retail['Sales'])
plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('sales Over Time')
plt.show()
```




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

```
result=seasonal_decompose(retail['Sales'],model='additive',period=12)
```

```
result.plot()
plt.show()
```




```
retail['ProductID']=np.random.randint(low=0,high=100,size=len(retail))
```

Now we'll do Product and Customer Analysis

Customer Demographics

```
customer_retail=retail[['Customer ID','Gender','Age']].drop_duplicates()
```

Calculating average purchase value per customer:

```
avg_purchase_value=retail.groupby('Customer ID')['Sales'].mean()
```

Top selling products

```
top_selling_products=retail.groupby('ProductID')['Sales'].sum().sort_values(ascending=False).head(10)
```

```
print("Top-Selling Products:")
print(top_selling_products)
```

```
Top-Selling Products:
ProductID
56    10731
57    10262
83     9946
25     9680
81     9489
64     9053
58     8380
69     8219
86     8103
84     8097
Name: Sales, dtype: int64
```

```
retail.dtypes
```

```
Transaction ID       int64
Customer ID         object
Gender              object
Age                  int64
Product Category    object
Quantity             int64
Price per Unit       int64
Total Amount         int64
Sales                int64
ProductID            int64
dtype: object
```

```
unique_customers = retail['Customer ID'].nunique()
```

```
print("Number of unique customers:", unique_customers)
```

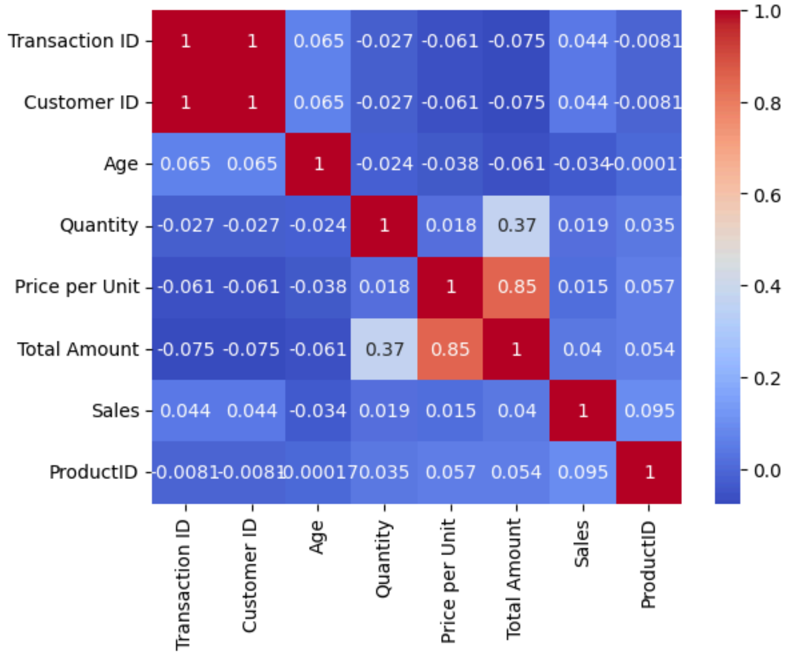
```
Number of unique customers: 1000
```

```
retail['Customer ID'] = retail['Customer ID'].str.replace('CUST', '').astype(int)
```

```
retail['Customer ID']=retail['Customer ID'].astype(float)
```

```
retail=retail.select_dtypes(include=[np.number])
```

```
sns.heatmap(retail.corr(),annot=True,cmap='coolwarm')
plt.show()
```



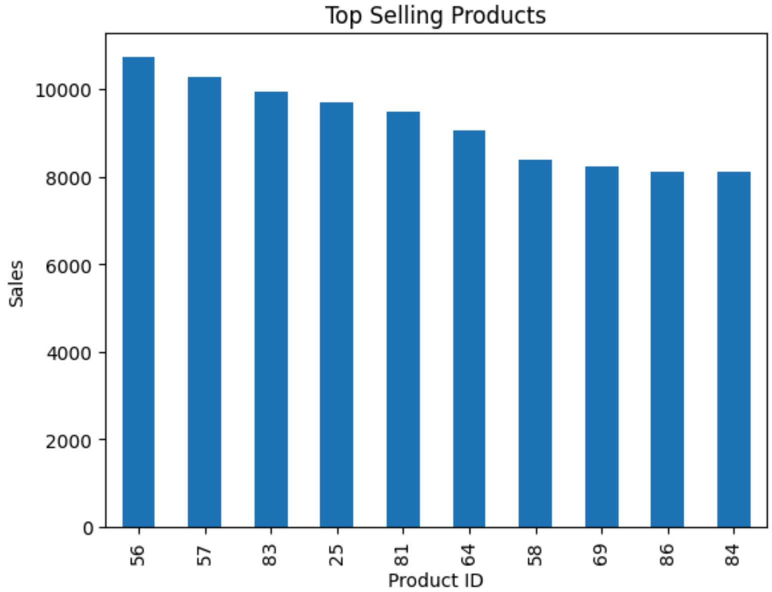
For Visualization

Creating a barchat for top selling products

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

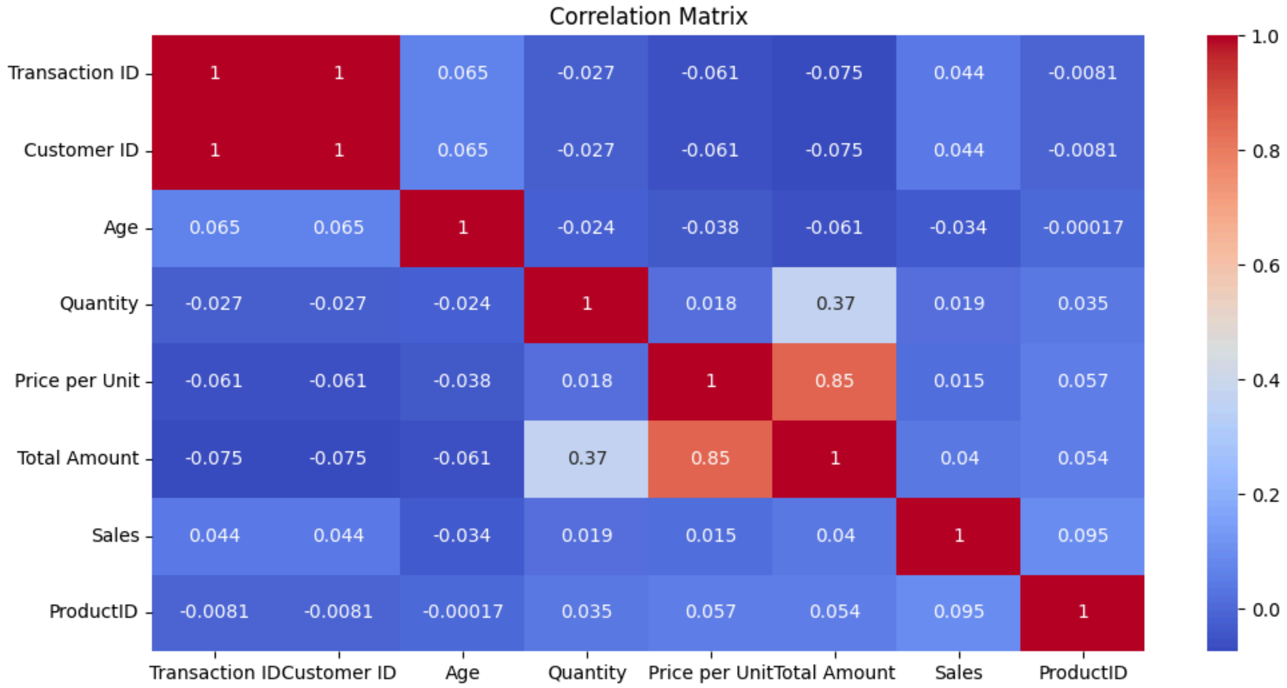
```
<Figure size 1200x600 with 0 Axes>
<Figure size 1200x600 with 0 Axes>
```

```
top_selling_products.plot(kind='bar')
plt.xlabel('Product ID')
plt.ylabel('Sales')
plt.title('Top Selling Products')
plt.show()
```



Now a heatmap showing correlation matrix

```
plt.figure(figsize=(12,6))
sns.heatmap(retail.corr(),annot=True,cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



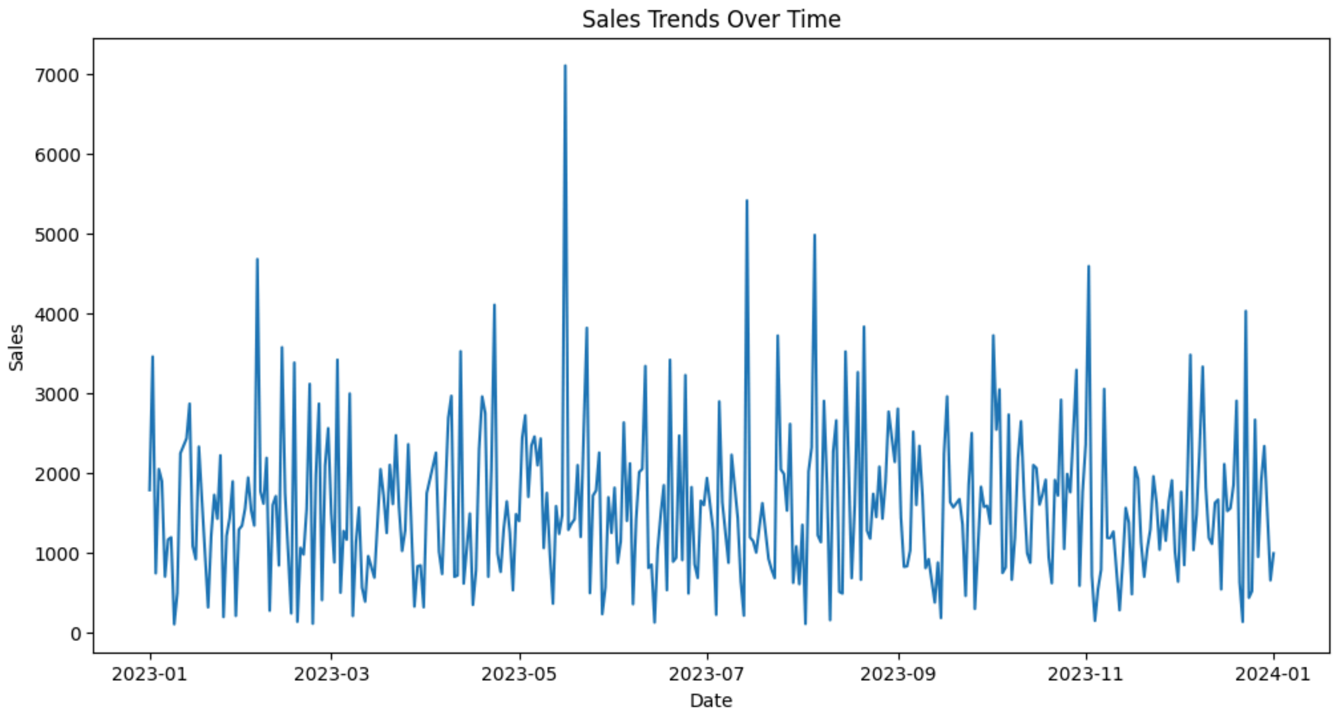
Now Sales Trend Analysis

Aggregate sales by date

```
daily_sales=retail.groupby('Date').agg({'Sales':'sum'}).reset_index()
```

Plotting Sales trends over time

```
plt.figure(figsize=(12,6))
plt.plot(daily_sales['Date'],daily_sales['Sales'])
plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales Trends Over Time')
plt.show()
```



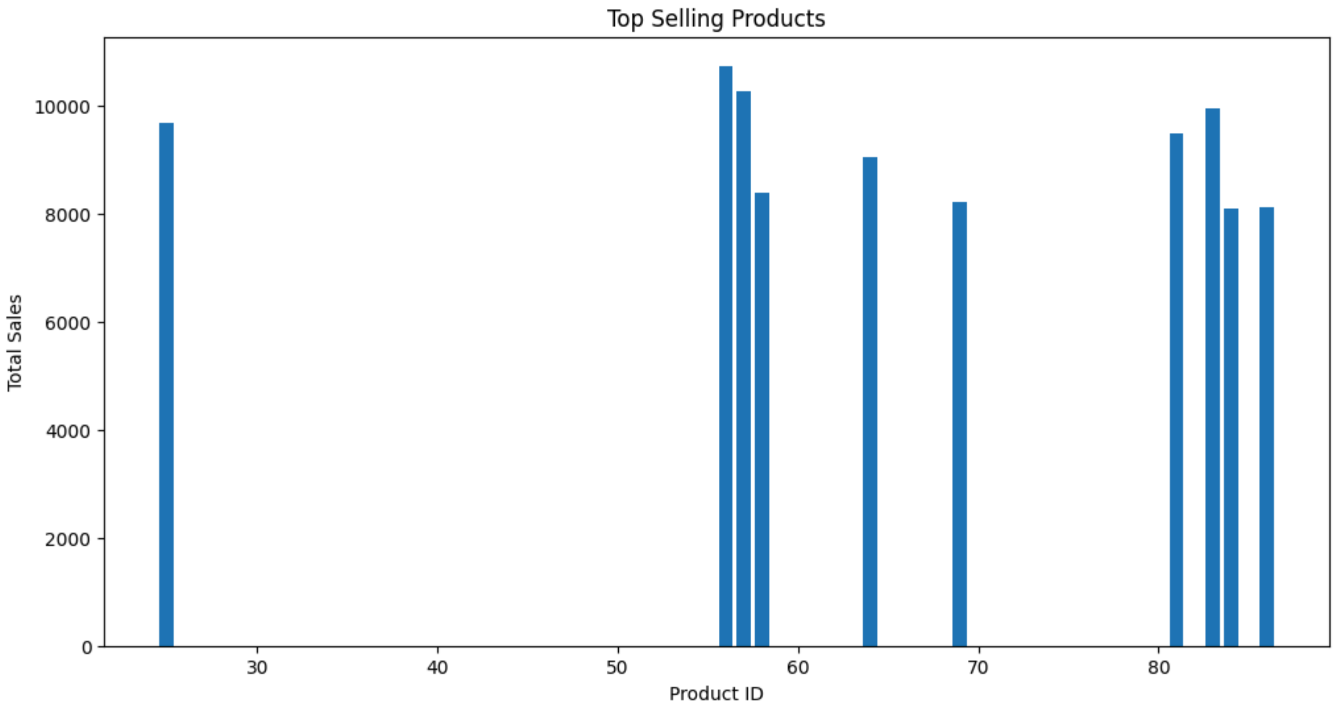
Now Product Performance

Identifying Top selling products

```
top_products=retail.groupby('ProductID').agg({'Sales':'sum'}).sort_values(by='Sales',ascending=False).head(10).reset_index()
```

Plotting Top selling Products

```
plt.figure(figsize=(12,6))
plt.bar(top_products['ProductID'],top_products['Sales'])
plt.xlabel('Product ID')
plt.ylabel('Total Sales')
plt.title('Top Selling Products')
plt.show()
```




```
retail['Gender']=np.random.randint(low=0,high=1000,size=len(retail))
```

Customer segmentation based on demographics

```
customer_segments=retail.groupby(['Customer ID','Age','Gender']).size().reset_index(name='PurchaseCount')
```

Displaying Customer Segments

```
print(customer_segments.head())
```

```
   Customer ID  Age  Gender  PurchaseCount
0          1.0   34     281              1
1          2.0   26     422              1
2          3.0   50     444              1
3          4.0   37     634              1
4          5.0   30     391              1
```

```
retail['Customer_segments']=np.random.randint(low=0,high=1000,size=len(retail))
```

Marketing Strategies

Analyzing customer segments for targeting marketing

```
young_customers=customer_segments[customer_segments['Age']<30]
male_customers=customer_segments[customer_segments['Gender']=='Male']
```

Example of Targeted marketing

```
print("Number of young customers:", len(young_customers))
print("Number of male customers:", len(male_customers))
```

```
Number of young customers: 251
Number of male customers: 0
```

Seasonal Trends-Identifying peak sales periods

```
peak_sales_periods=daily_sales[daily_sales['Sales']>daily_sales['Sales'].quantile(0.8)]
```

```
print("Peak Sales Periods:",peak_sales_periods['Date'].dt.month.value_counts().sort_index())
```

```
Peak Sales Periods: Date
1     4
2     6
3     4
4     7
5     8
6     5
7     4
8     9
9     5
10    7
11    3
12    7
Name: count, dtype: int64
```

Using insights for marketing strategies