1. Data Loading & Initial Exploration

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime

# Load the dataset
df = pd.read_csv('/Users/Dell/OneDrive/Desktop/Ecommerce_Strategic_Assignment_Dataset.csv')

# Display first 5 rows
print("First 5 rows:")
display(df.head())

# Check basic info
print("\nDataset info:")
df.info()

# Check for missing values
print("\nMissing values per column:")
print(df.isnull().sum())

# Basic statistics
print("\nDescriptive statistics:")
display(df.describe())
```

```
First 5 rows:
```

	Date	Traffic_Source	Campaign	Customer_Type	Visitors	Marketing_Spend	Add_to_Cart	Purchases	Revenue	Cart_Abandonment_Rate	Repeat_I
0	2023-01-01	Email Campaign	Flash Deals	New	1226	1469.85	261	46.0	6416.70	19.11	
1	2023-01-02	Social Media	Flash Deals	New	1559	4429.12	274	153.0	6692.86	49.04	
2	2023-01-03	Direct	Winter Promo	New	960	3763.82	69	238.0	4355.64	36.69	
3	2023-01-04	Direct	Flash Deals	New	1394	4770.05	162	187.0	5613.54	10.42	
4	2023-01-05	Organic Search	Summer Sale	Returning	1230	1720.68	189	117.0	4717.28	20.22	

```
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 450 entries, 0 to 449
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Date                   450 non-null    object 
 1   Traffic_Source         450 non-null    object 
 2   Campaign               450 non-null    object 
 3   Customer_Type          450 non-null    object 
 4   Visitors               450 non-null    int64  
 5   Marketing_Spend        405 non-null    float64
 6   Add_to_Cart            450 non-null    int64  
 7   Purchases              405 non-null    float64
 8   Revenue                405 non-null    float64
 9   Cart_Abandonment_Rate  450 non-null    float64
 10  Repeat_Purchase_Rate   450 non-null    float64
 11  Conversion_Rate        450 non-null    float64
dtypes: float64(6), int64(2), object(4)
memory usage: 42.3+ KB

Missing values per column:
Date                      0
Traffic_Source            0
Campaign                  0
Customer_Type             0
Visitors                  0
Marketing_Spend          45
Add_to_Cart               0
Purchases                45
Revenue                  45
Cart_Abandonment_Rate     0
Repeat_Purchase_Rate      0
Conversion_Rate           0
dtype: int64

Descriptive statistics:
```

	Visitors	Marketing_Spend	Add_to_Cart	Purchases	Revenue	Cart_Abandonment_Rate	Repeat_Purchase_Rate	Conversion_Rate
count	450.000000	405.000000	450.000000	405.000000	405.000000	450.000000	450.000000	450.000000
mean	1095.306667	2543.955111	275.624444	158.158025	5415.334741	29.528867	17.650378	5.640600
std	544.475596	1391.484598	130.688766	83.107885	2492.708899	11.835111	7.228703	2.575682
min	101.000000	112.720000	50.000000	21.000000	1002.990000	10.010000	5.050000	1.000000
25%	641.500000	1373.750000	164.000000	81.000000	3237.220000	18.882500	11.427500	3.425000
50%	1123.000000	2515.830000	274.500000	166.000000	5494.000000	29.495000	18.195000	5.740000
75%	1574.000000	3711.470000	382.750000	233.000000	7439.420000	39.877500	23.725000	7.925000
max	1999.000000	4986.550000	499.000000	299.000000	9959.320000	49.880000	29.990000	10.000000

2. Data Cleaning

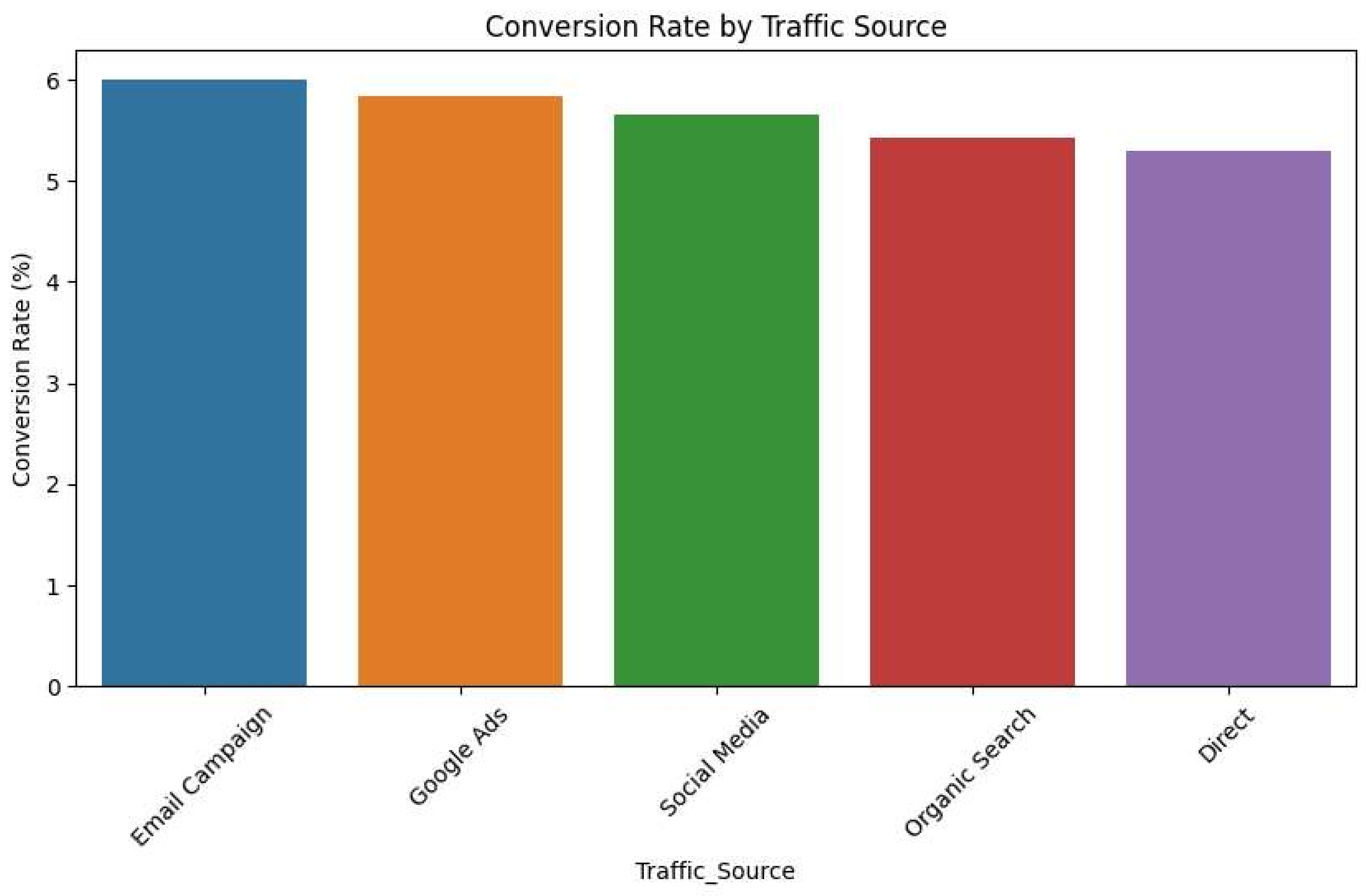
In [6]:
```
# Fill missing numerical values with median
df['Marketing_Spend'] = df['Marketing_Spend'].fillna(df['Marketing_Spend'].median())
df['Revenue'] = df['Revenue'].fillna(df['Revenue'].median())
df['Purchases'] = df['Purchases'].fillna(df['Purchases'].median())

# Convert 'Date' to datetime
df['Date'] = pd.to_datetime(df['Date'])
```

3. Key Performance Metrics

A. Conversion Rate by Traffic Source

In [7]:
```python
conversion_by_source = df.groupby('Traffic_Source')['Conversion_Rate'].mean().sort_values(ascending=False)
print("Avg. Conversion Rate by Traffic Source:\n", conversion_by_source)

# Plot
plt.figure(figsize=(10, 5))
sns.barplot(x=conversion_by_source.index, y=conversion_by_source.values)
plt.title("Conversion Rate by Traffic Source")
plt.ylabel("Conversion Rate (%)")
plt.xticks(rotation=45)
plt.show()
```

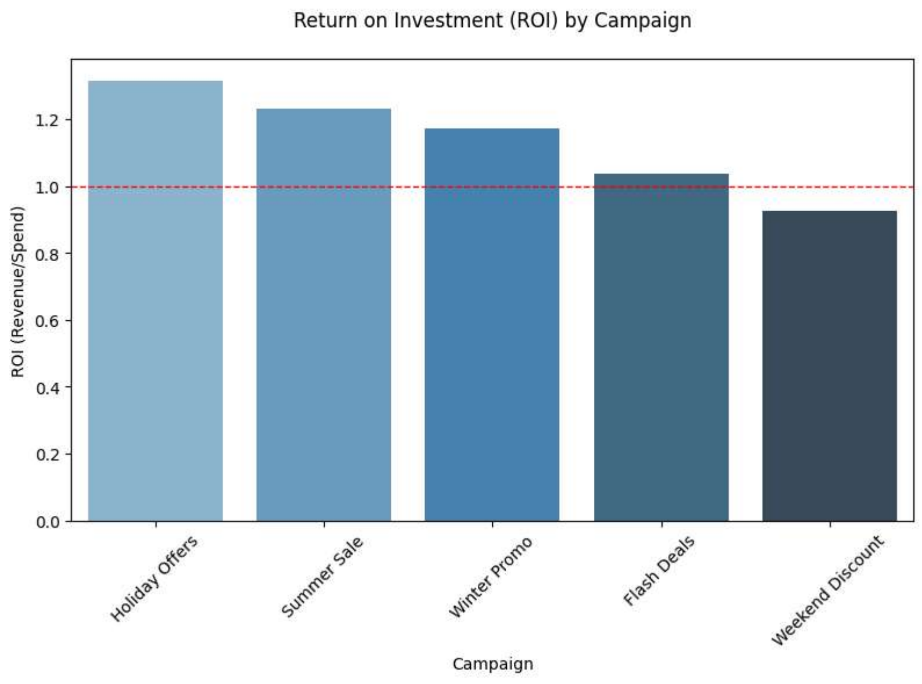
```
Avg. Conversion Rate by Traffic Source:
 Traffic_Source
Email Campaign    5.994138
Google Ads        5.838947
Social Media      5.661512
Organic Search    5.425316
Direct            5.306699
Name: Conversion_Rate, dtype: float64
```

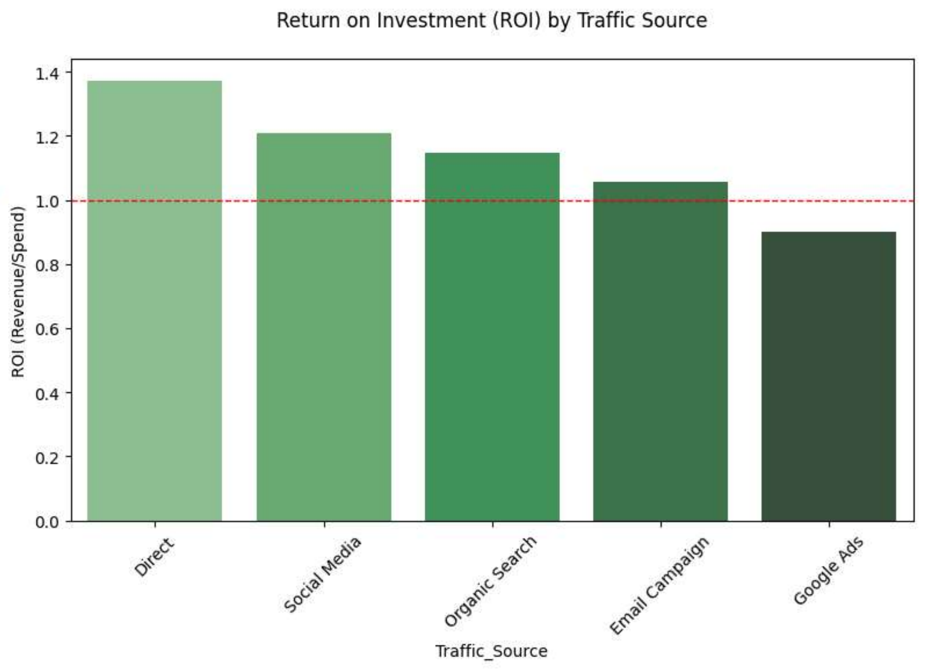


B. Revenue vs. Marketing Spend

In [16]:
```
# Calculate ROI by Campaign
roi_by_campaign = df.groupby('Campaign').apply(
    lambda x: (x['Revenue'].sum() - x['Marketing_Spend'].sum()) / x['Marketing_Spend'].sum()
).sort_values(ascending=False)

# Calculate ROI by Traffic Source
roi_by_traffic = df.groupby('Traffic_Source').apply(
    lambda x: (x['Revenue'].sum() - x['Marketing_Spend'].sum()) / x['Marketing_Spend'].sum()
).sort_values(ascending=False)

print("ROI by Campaign:\n", roi_by_campaign)
print("\nROI by Traffic Source:\n", roi_by_traffic)

# Create figure with two subplots
plt.figure(figsize=(16, 6))

# ROI by Campaign plot
plt.subplot(1, 2, 1)
sns.barplot(x=roi_by_campaign.index, y=roi_by_campaign.values, palette="Blues_d")
plt.title("Return on Investment (ROI) by Campaign", pad=20)
plt.ylabel("ROI (Revenue/Spend)")
plt.xticks(rotation=45)
plt.axhline(y=1, color='red', linestyle='--', linewidth=1)  # Break-even line

# ROI by Traffic Source plot
plt.subplot(1, 2, 2)
sns.barplot(x=roi_by_traffic.index, y=roi_by_traffic.values, palette="Greens_d")
plt.title("Return on Investment (ROI) by Traffic Source", pad=20)
plt.ylabel("ROI (Revenue/Spend)")
plt.xticks(rotation=45)
plt.axhline(y=1, color='red', linestyle='--', linewidth=1)  # Break-even line

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

```
ROI by Campaign:
 Campaign
Holiday Offers      1.315419
Summer Sale         1.232844
Winter Promo        1.171792
Flash Deals         1.037087
Weekend Discount    0.926592
dtype: float64

ROI by Traffic Source:
 Traffic_Source
Direct            1.372385
Social Media      1.206972
Organic Search    1.147716
Email Campaign    1.056986
Google Ads        0.901510
dtype: float64
```

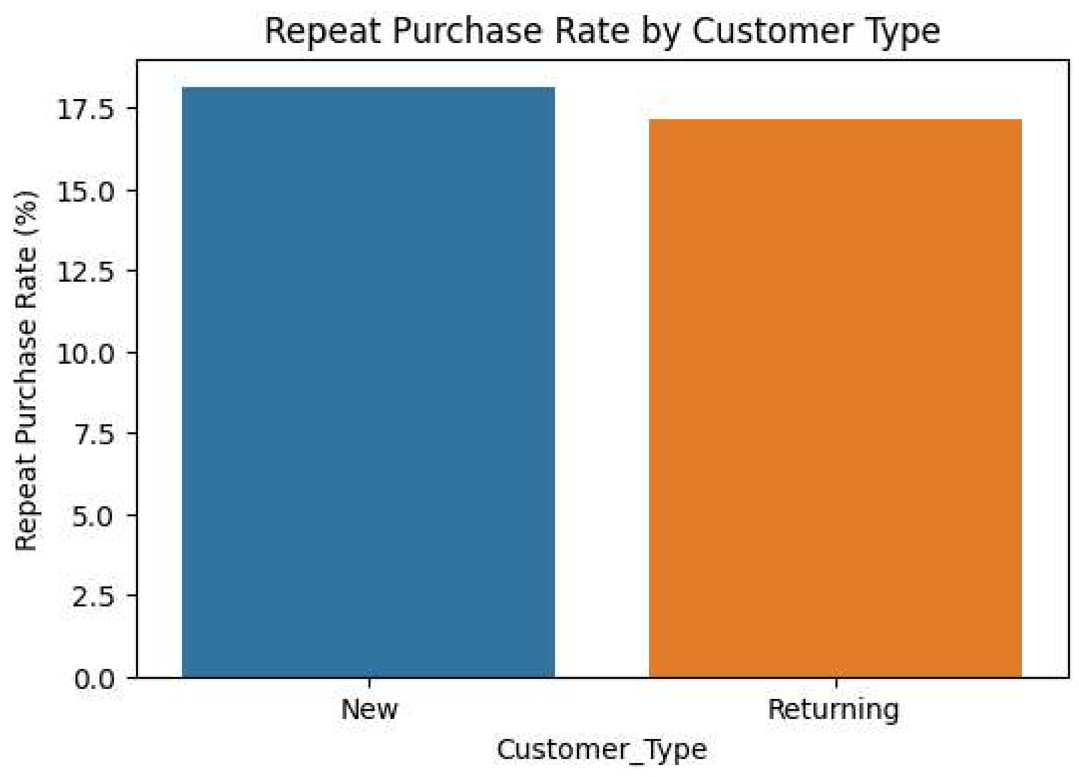




C. Customer Retention Analysis

In [10]:
```
repeat_purchase_rates = df.groupby('Customer_Type')['Repeat_Purchase_Rate'].mean()
print("Avg. Repeat Purchase Rate:\n", repeat_purchase_rates)

# Plot
plt.figure(figsize=(6, 4))
sns.barplot(x=repeat_purchase_rates.index, y=repeat_purchase_rates.values)
plt.title("Repeat Purchase Rate by Customer Type")
plt.ylabel("Repeat Purchase Rate (%)")
plt.show()
```

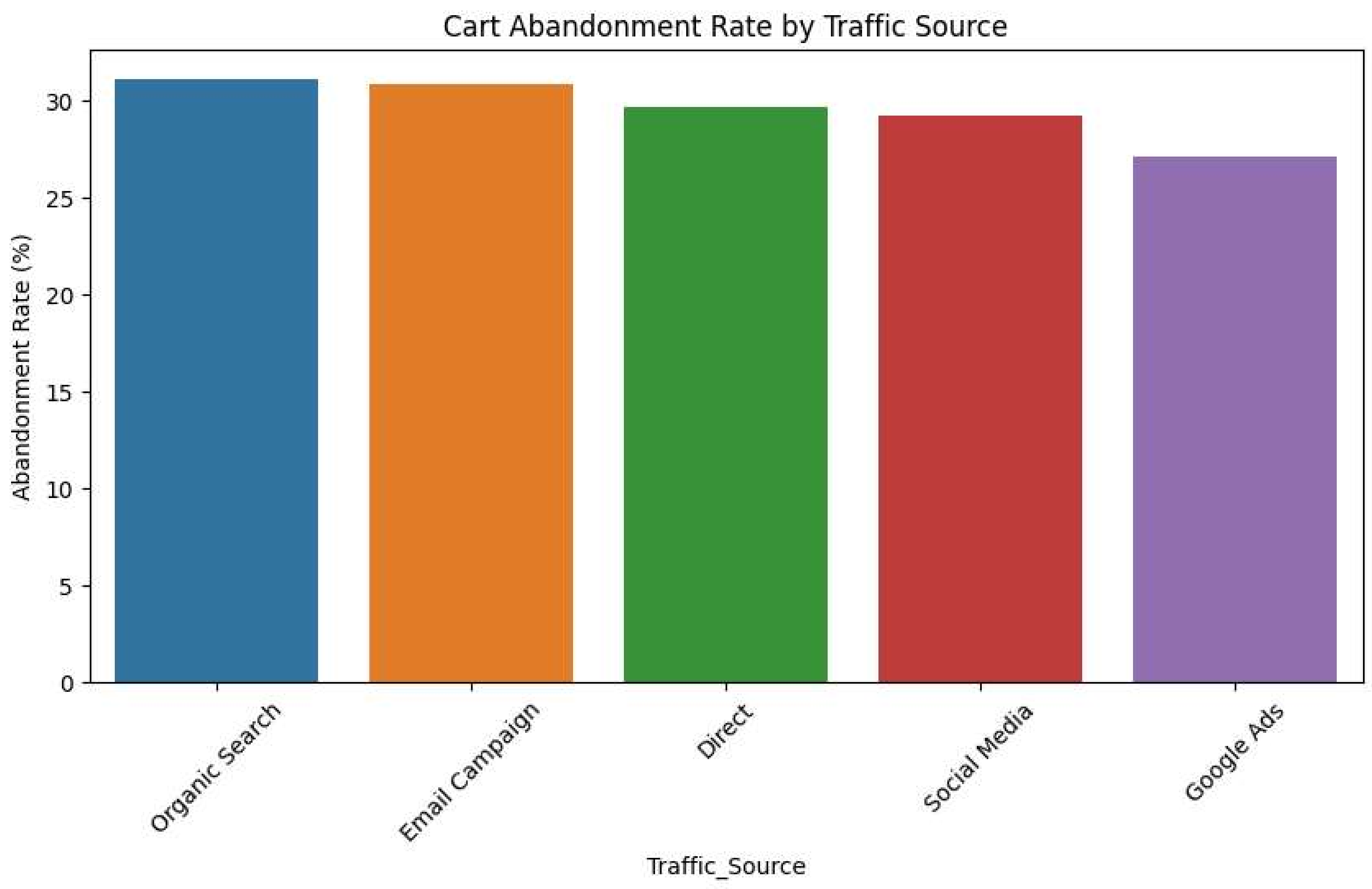
```
Avg. Repeat Purchase Rate:
 Customer_Type
New          18.116856
Returning    17.167014
Name: Repeat_Purchase_Rate, dtype: float64
```



D. Cart Abandonment Analysis

In [11]:
```
abandonment_by_source = df.groupby('Traffic_Source')['Cart_Abandonment_Rate'].mean().sort_values(ascending=False)
print("Avg. Cart Abandonment by Source:\n", abandonment_by_source)

# Plot
plt.figure(figsize=(10, 5))
sns.barplot(x=abandonment_by_source.index, y=abandonment_by_source.values)
plt.title("Cart Abandonment Rate by Traffic Source")
plt.ylabel("Abandonment Rate (%)")
plt.xticks(rotation=45)
plt.show()
```

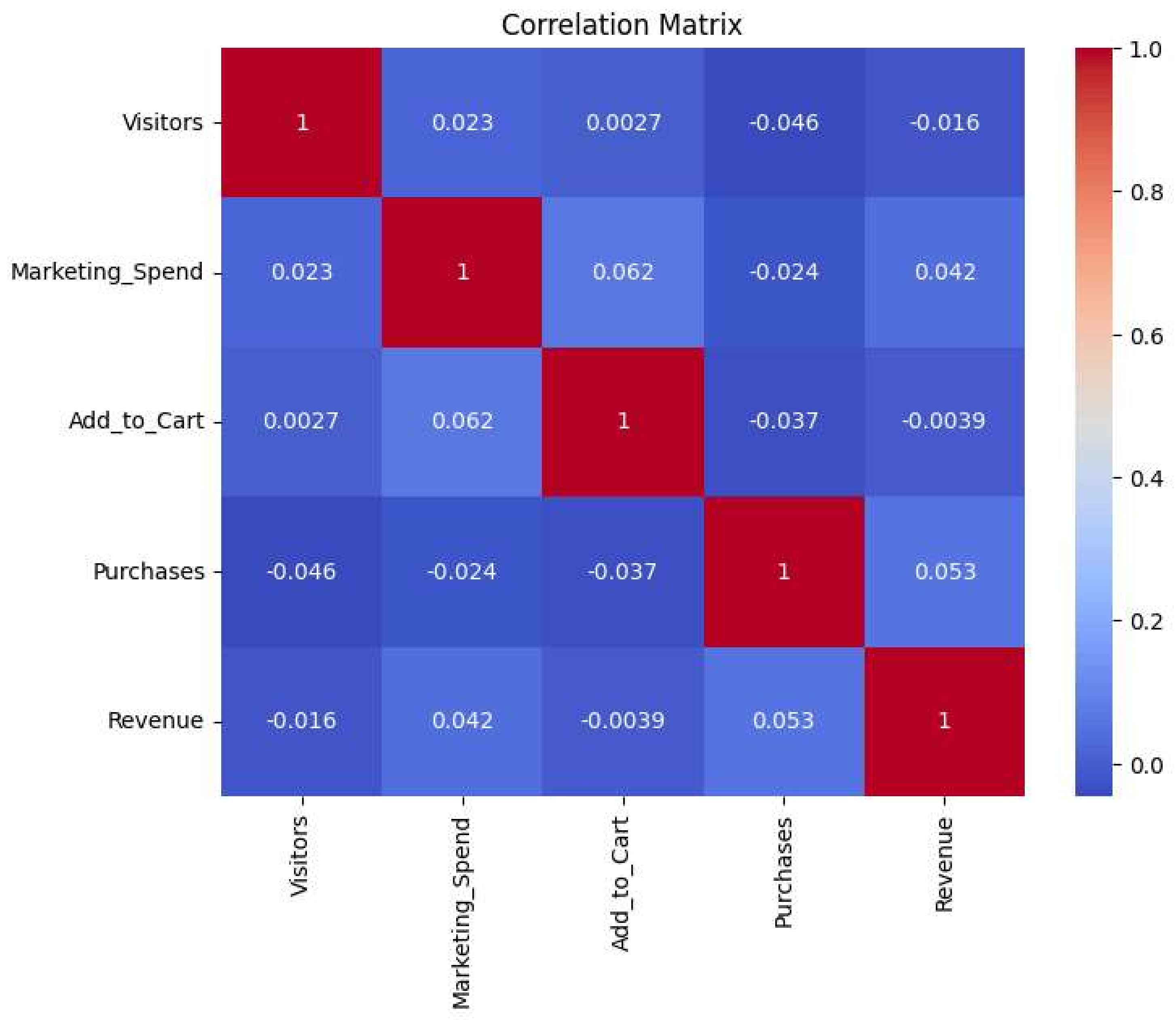
```
Avg. Cart Abandonment by Source:
 Traffic_Source
Organic Search    31.063418
Email Campaign    30.889425
Direct            29.666214
Social Media      29.202442
Google Ads        27.153368
Name: Cart_Abandonment_Rate, dtype: float64
```



4. Advanced Insights (Correlation & Trends)

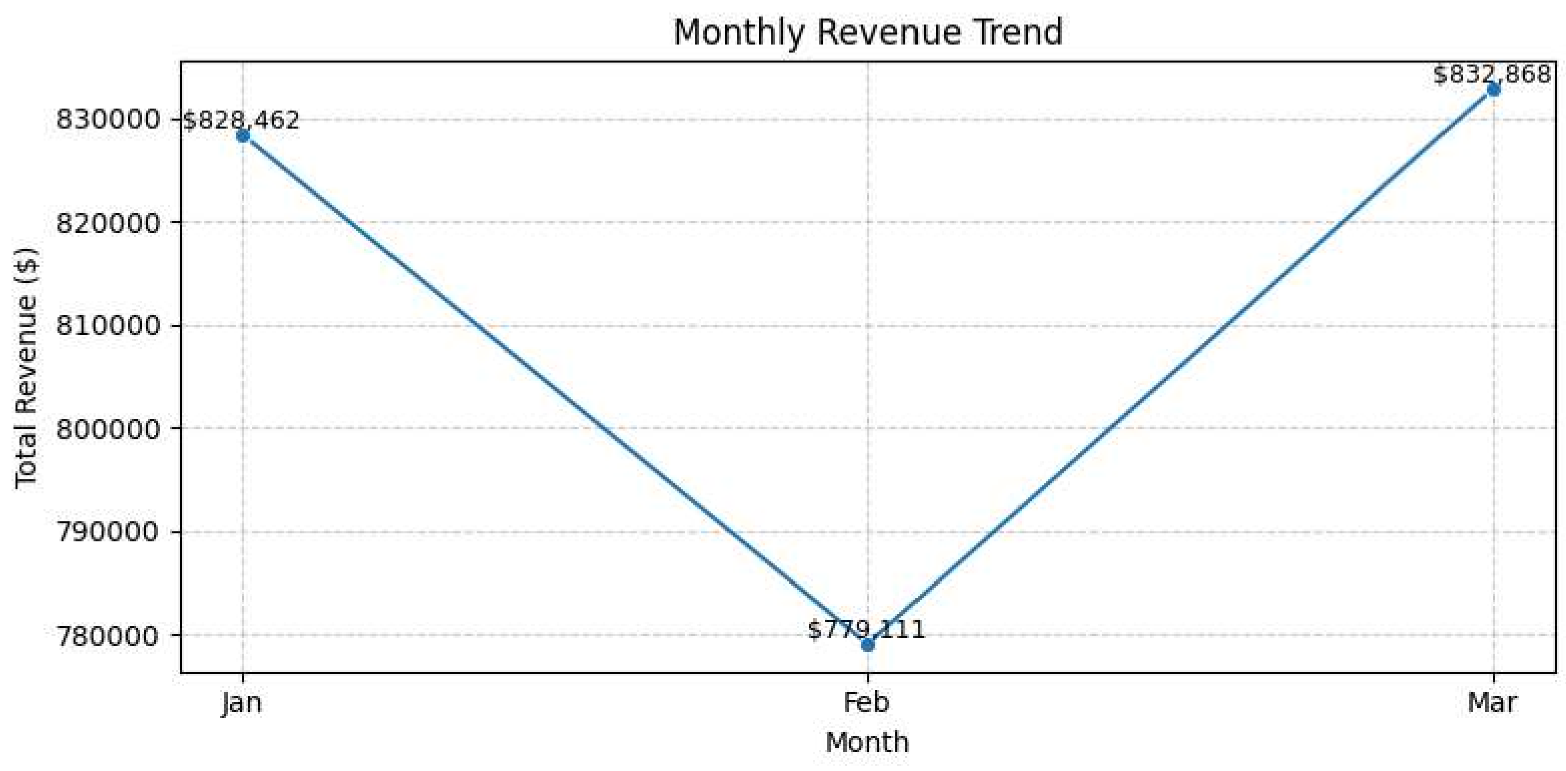
A. Correlation Matrix

```
In [17]: corr_matrix = df[['Visitors', 'Marketing_Spend', 'Add_to_Cart', 'Purchases', 'Revenue']].corr()
         plt.figure(figsize=(8, 6))
         sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
         plt.title("Correlation Matrix")
         plt.show()
```



B. Monthly Revenue Trend

In [19]:
```
df['Month'] = df['Date'].dt.month
monthly_revenue = df.groupby('Month')['Revenue'].sum()

plt.figure(figsize=(8, 4))
sns.lineplot(x=monthly_revenue.index, y=monthly_revenue.values, marker='o')
plt.grid(True, linestyle='--', alpha=0.6)

for x, y in zip(monthly_revenue.index, monthly_revenue.values):
    plt.text(x, y, f"${y:,.0f}", ha='center', va='bottom', fontsize=9)

plt.title("Monthly Revenue Trend")
plt.xlabel("Month")
plt.ylabel("Total Revenue ($)")
plt.xticks([1, 2, 3], ['Jan', 'Feb', 'Mar'])
plt.tight_layout()
plt.show()
```



In [20]:
```
# Group by traffic source
traffic_stats = df.groupby('Traffic_Source').agg({
    'Visitors': 'sum',
    'Marketing_Spend': 'sum',
    'Revenue': 'sum',
    'Conversion_Rate': 'mean',
    'Cart_Abandonment_Rate': 'mean'
}).sort_values('Revenue', ascending=False)

print(traffic_stats)
```

```
                Visitors  Marketing_Spend    Revenue  Conversion_Rate  \
Traffic_Source
Direct            114651        244688.35  580494.91         5.306699
Google Ads        106263        253435.67  481910.54         5.838947
Social Media       98681        215975.03  476650.92         5.661512
Email Campaign     89993        230107.16  473327.23         5.994138
Organic Search     83300        199307.96  428056.97         5.425316

                Cart_Abandonment_Rate
Traffic_Source
Direct                      29.666214
Google Ads                  27.153368
Social Media                29.202442
Email Campaign              30.889425
Organic Search              31.063418
```

In [ ]: