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

```
df = pd.read_csv("/content/archive (5).zip")
df.head()
```

	Unnamed: 0	price	discount	promotion_intensity	footfall	ad_spend	competitor_price	s
0	0	45.197454	5.514259	4.062653	277.017484	2559.073870	44.255411	1
1	1	49.327512	6.572035	4.964657	250.760714	2536.417155	50.331704	1
2	2	47.328457	6.972713	4.363191	263.130478	2552.952356	49.285996	1
3	3	50.964538	4.808234	3.577988	297.603918	2605.398826	46.839936	1
4	4	44.530213	8.180216	4.966638	208.931691	2432.485683	45.336500	1

Next steps: Generate code with df New interactive sheet

```
df = df.drop("Unnamed: 0",axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   price                15000 non-null  float64
 1   discount             15000 non-null  float64
 2   promotion_intensity  15000 non-null  float64
 3   footfall             15000 non-null  float64
 4   ad_spend             15000 non-null  float64
 5   competitor_price     15000 non-null  float64
 6   stock_level          15000 non-null  float64
 7   weather_index        15000 non-null  float64
 8   customer_sentiment   15000 non-null  float64
 9   return_rate          15000 non-null  float64
dtypes: float64(10)
memory usage: 1.1 MB
```

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

```
np.int64(0)
```

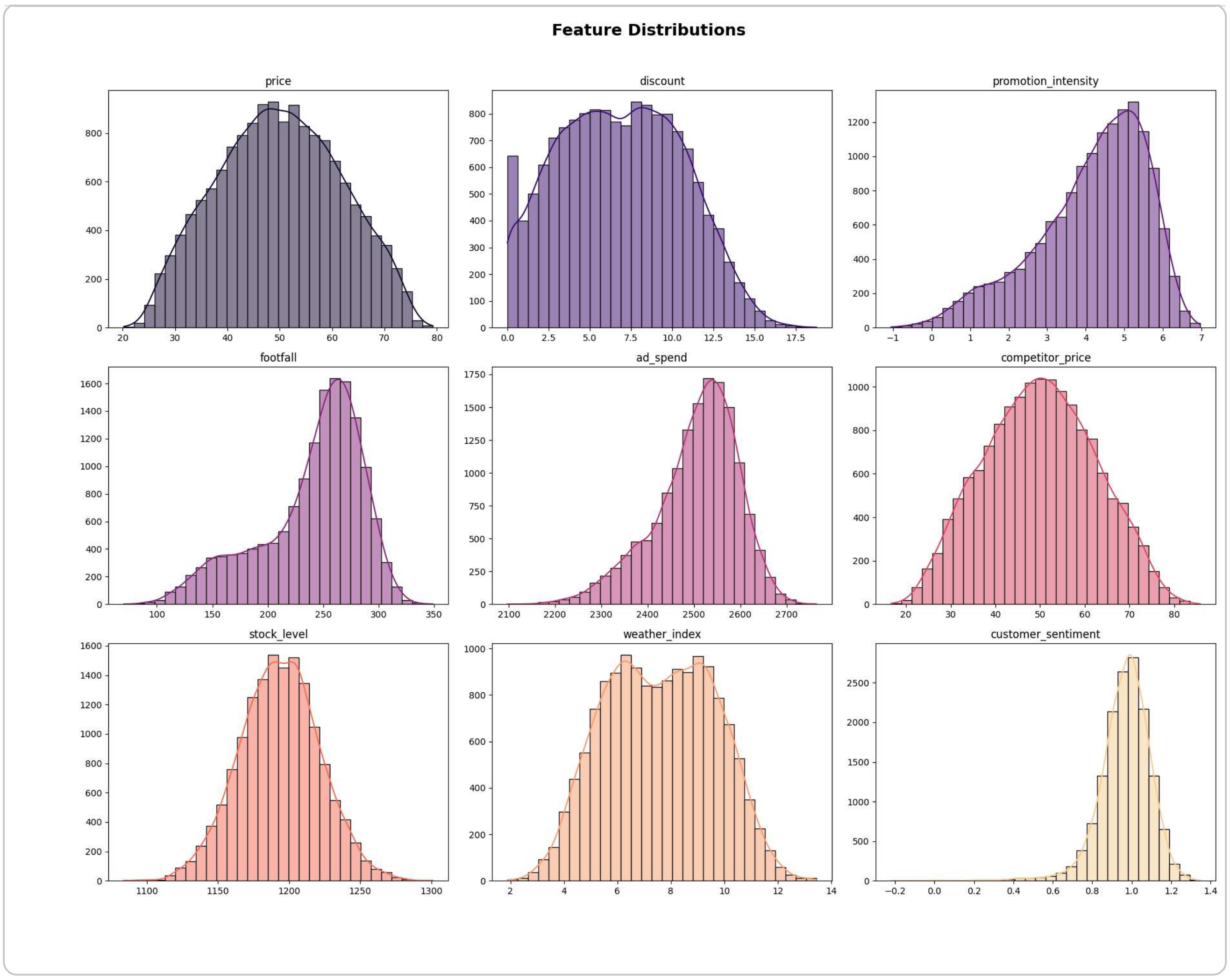
```
df.duplicated().sum()
```

```
np.int64(0)
```

```
print("🔎 UNIQUE VALUE CHECK\n")

for col in df.columns:
    unique_count = df[col].nunique()

    print(f"📌 Column: {col}")
    print(f"   ➤ Unique Count: {unique_count}")
    if unique_count < 10:
        uniques = df[col].unique()
        print(f"    ➤ Unique Values: {list(uniques)}")

    print("-" * 50)
```

```
🔎 UNIQUE VALUE CHECK

📌 Column: price
   ➤ Unique Count: 15000
--------------------------------------------------
📌 Column: discount
   ➤ Unique Count: 14625
--------------------------------------------------
📌 Column: promotion_intensity
   ➤ Unique Count: 15000
--------------------------------------------------
📌 Column: footfall
   ➤ Unique Count: 15000
--------------------------------------------------
📌 Column: ad_spend
   ➤ Unique Count: 15000
--------------------------------------------------
📌 Column: competitor_price
   ➤ Unique Count: 15000
--------------------------------------------------
📌 Column: stock_level
   ➤ Unique Count: 15000
--------------------------------------------------
📌 Column: weather_index
   ➤ Unique Count: 15000
--------------------------------------------------
📌 Column: customer_sentiment
   ➤ Unique Count: 15000
--------------------------------------------------
📌 Column: return_rate
   ➤ Unique Count: 14999
--------------------------------------------------
```

```
features = [
    "price", "discount", "promotion_intensity", "footfall",
    "ad_spend", "competitor_price", "stock_level",
    "weather_index", "customer_sentiment"]

palette = sns.color_palette("magma", n_colors=len(features))

plt.figure(figsize=(18, 14))

for i, col in enumerate(features, 1):
    plt.subplot(3, 3, i)
    sns.histplot(
        data=df,
        x=col,
        bins=30,
        kde=True,
        color=palette[i-1]
    )
    plt.title(col, fontsize=12)
    plt.xlabel("")
    plt.ylabel("")

plt.suptitle("Feature Distributions", fontsize=18, fontweight="bold")
plt.tight_layout(rect=[0, 0,  1, 0.96])
plt.show()
```



Key EDA Observations

price, competitor_price, and stock_level show very similar, near-symmetric distributions, suggesting stable and well-controlled value ranges.

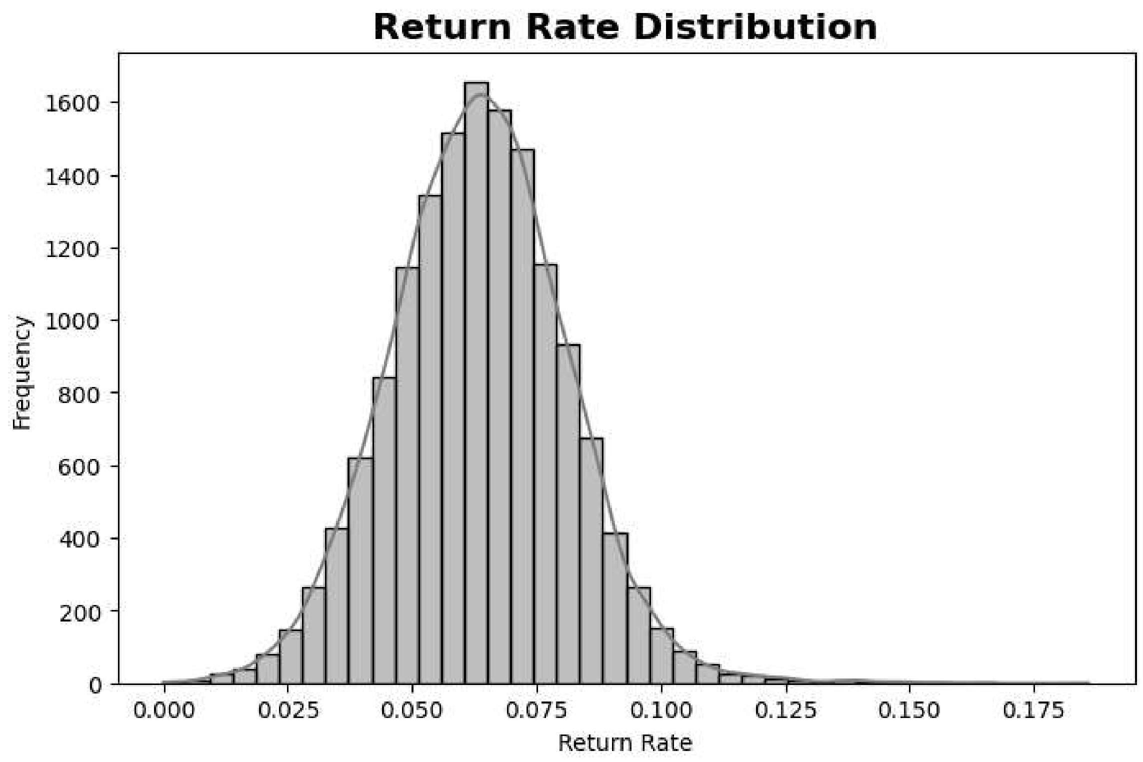
promotion_intensity appears approximately symmetric and smoothly distributed, indicating balanced promotional activity rather than extreme campaigns.

•# discount shows slight right skewness, meaning higher discount values are present but less frequent.

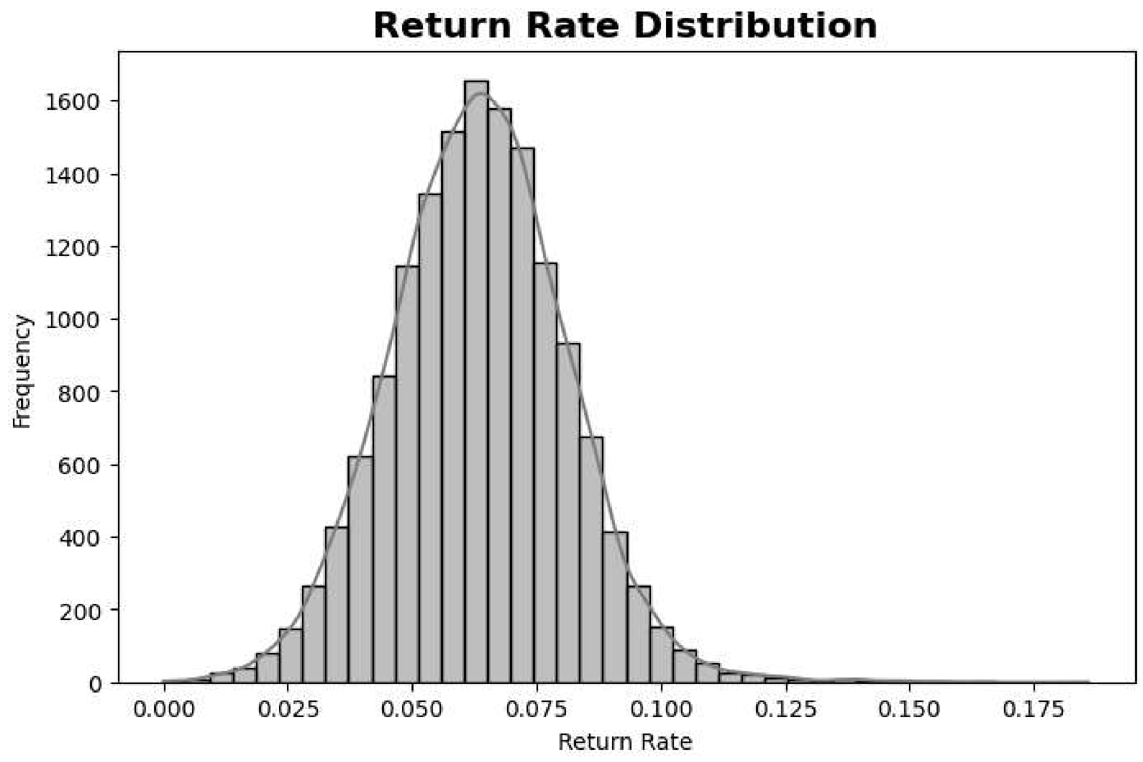
• customer_sentiment is strongly concentrated at higher values, indicating generally positive customer experiences with limited negative cases.

```
plt.figure(figsize=(8, 5))

sns.histplot(
    df["return_rate"],
    bins=40,
    kde=True,
    color="gray"
)
```

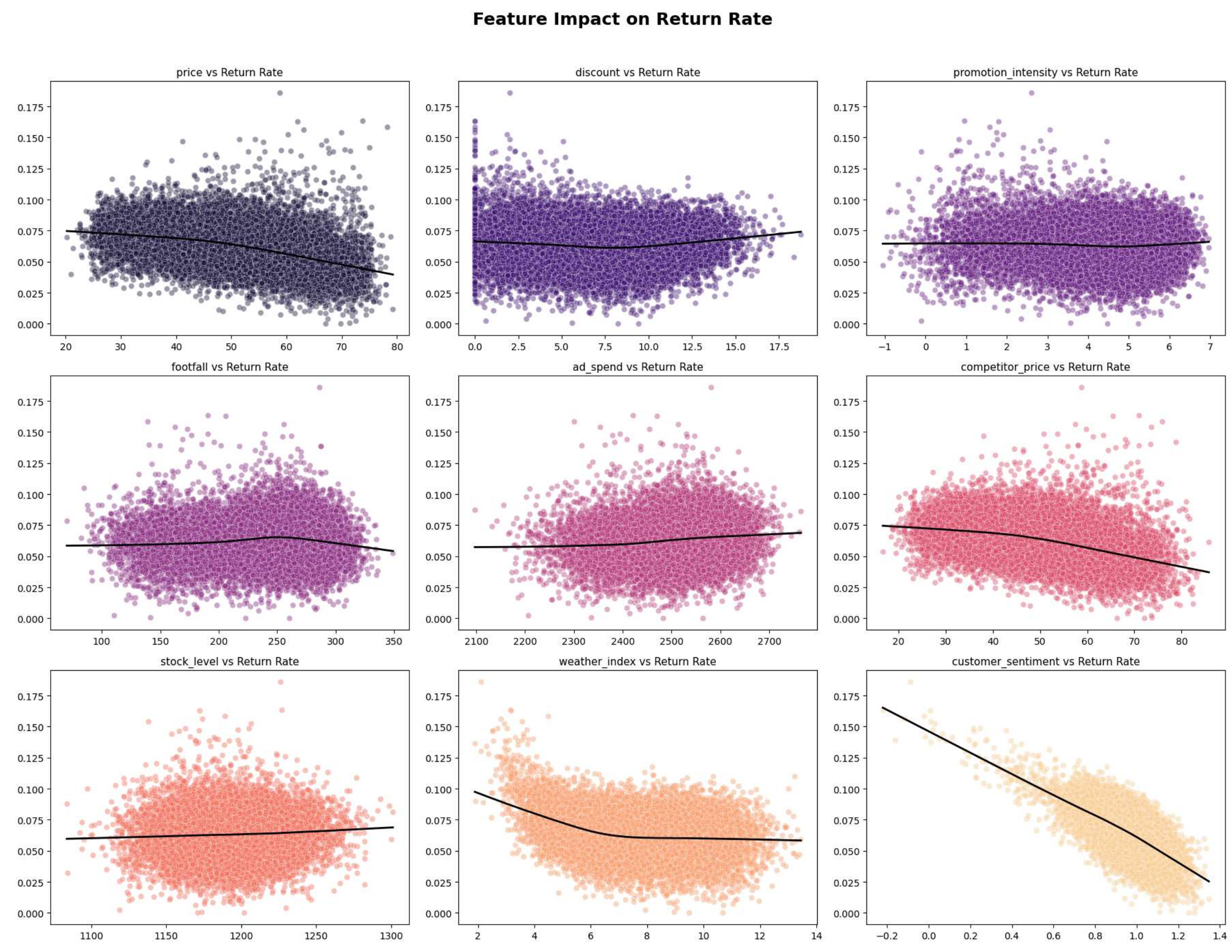
```
plt.title("Return Rate Distribution", fontsize=16, fontweight="bold")
plt.xlabel("Return Rate")
plt.ylabel("Frequency")
plt.show()
```



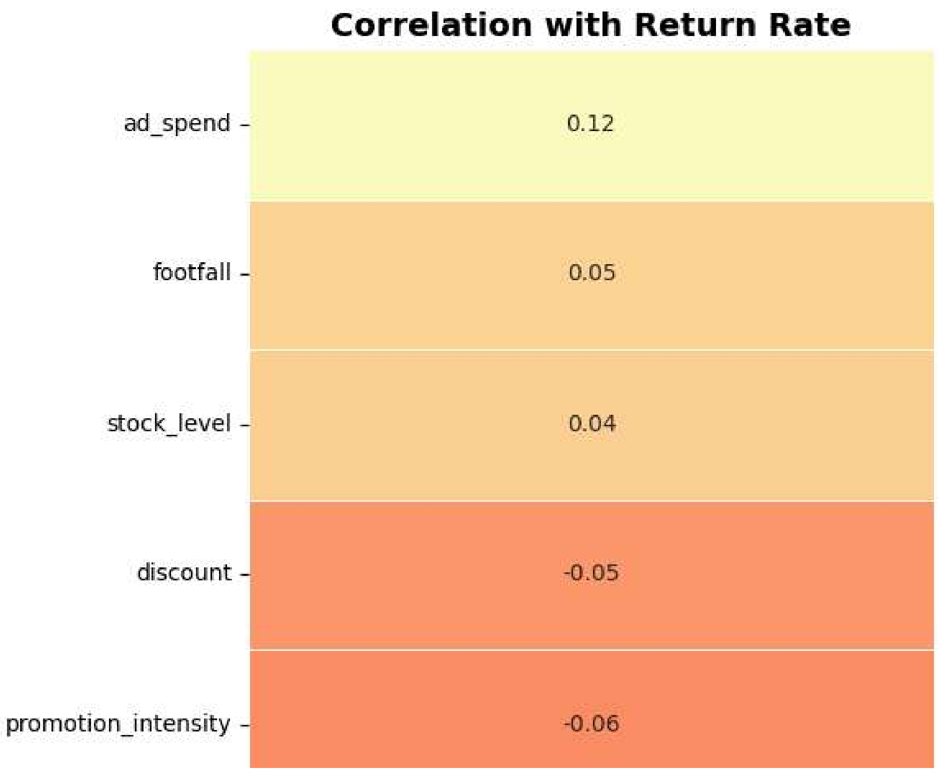
```
plt.figure(figsize=(8, 5))

sns.histplot(
    df["return_rate"],
    bins=40,
    kde=True,
    color="gray"
)

plt.title("Return Rate Distribution", fontsize=16, fontweight="bold")
plt.xlabel("Return Rate")
plt.ylabel("Frequency")
plt.show()
```



```
palette = sns.color_palette("magma", n_colors=len(features))

plt.figure(figsize=(18, 14))

for i, col in enumerate(features, 1):
    plt.subplot(3, 3, i)
    sns.scatterplot(
        data=df,
        x=col,
        y=df["return_rate"],
        alpha=0.4,
        color=palette[i-1])

    sns.regplot(
        data=df,
        x=col,
        y=df["return_rate"],
        scatter=False,
        lowess=True,
        color="black",
        line_kws={"linewidth": 2})

    plt.title(f"{col} vs Return Rate", fontsize=11)
    plt.xlabel("")
    plt.ylabel("")

plt.suptitle("Feature Impact on Return Rate", fontsize=18, fontweight="bold")
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



```
df_corr = df.copy()

corr_matrix = df_corr.corr()

target_corr = (
    corr_matrix["return_rate"]
    .drop("return_rate")
    .sort_values(ascending=False))

plt.figure(figsize=(6, 9))

sns.heatmap(
    target_corr.to_frame(),
    annot=True,
    fmt=".2f",
    cmap="magma",
    linewidths=0.5,
    cbar=False)

plt.title("Correlation with Return Rate", fontsize=14, fontweight="bold")
plt.ylabel("")
plt.xlabel("")

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


Correlation with Return Rate
ad_spend
0.12
footfall
0.05
stock_level
0.04
discount
-0.05
promotion_intensity
-0.06