```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

load dataset

```
data_df = pd.read_csv("/historical_data.csv")
sentiment_df = pd.read_csv("/fear_greed_index.csv")
```

Clean and Prepare data_df

```
data_df.head()
```

	Account	Coin	Execution Price	Size Tokens	Size USD	Side	Timestamp IST	Start Position	Direction	Closed PnL	
0	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9769	986.87	7872.16	BUY	02-12-2024 22:50	0.000000	Buy	0.0	0xec
1	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9800	16.00	127.68	BUY	02-12-2024 22:50	986.524596	Buy	0.0	0xec
2	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9855	144.09	1150.63	BUY	02-12-2024 22:50	1002.518996	Buy	0.0	0xec
3	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9874	142.98	1142.04	BUY	02-12-2024 22:50	1146.558564	Buy	0.0	0xec
4	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9894	8.73	69.75	BUY	02-12-2024 22:50	1289.488521	Buy	0.0	0xec

```
data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211224 entries, 0 to 211223
Data columns (total 16 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Account          211224 non-null  object
 1   Coin             211224 non-null  object
 2   Execution Price  211224 non-null  float64
 3   Size Tokens      211224 non-null  float64
 4   Size USD         211224 non-null  float64
 5   Side             211224 non-null  object
 6   Timestamp IST    211224 non-null  object
 7   Start Position   211224 non-null  float64
 8   Direction        211224 non-null  object
 9   Closed PnL       211224 non-null  float64
 10  Transaction Hash 211224 non-null  object
 11  Order ID         211224 non-null  int64
 12  Crossed          211224 non-null  bool
 13  Fee              211224 non-null  float64
 14  Trade ID         211224 non-null  float64
 15  Timestamp        211224 non-null  float64
dtypes: bool(1), float64(8), int64(1), object(6)
memory usage: 24.4+ MB
```

```
data_df.describe()
```

	Execution Price	Size Tokens	Size USD	Start Position	Closed PnL	Order ID	Fee	Trade ID	Timestamp
count	211224.000000	2.112240e+05	2.112240e+05	2.112240e+05	211224.000000	2.112240e+05	211224.000000	2.112240e+05	2.112240e+05
mean	11414.723350	4.623365e+03	5.639451e+03	-2.994625e+04	48.749001	6.965388e+10	1.163967	5.628549e+14	1.737744e+12
std	29447.654868	1.042729e+05	3.657514e+04	6.738074e+05	919.164828	1.835753e+10	6.758854	3.257565e+14	8.689920e+09
min	0.000005	8.740000e-07	0.000000e+00	-1.433463e+07	-117990.104100	1.732711e+08	-1.175712	0.000000e+00	1.680000e+12
25%	4.854700	2.940000e+00	1.937900e+02	-3.762311e+02	0.000000	5.983853e+10	0.016121	2.810000e+14	1.740000e+12
50%	18.280000	3.200000e+01	5.970450e+02	8.472793e+01	0.000000	7.442939e+10	0.089578	5.620000e+14	1.740000e+12
75%	101.580000	1.879025e+02	2.058960e+03	9.337278e+03	5.792797	8.335543e+10	0.393811	8.460000e+14	1.740000e+12

```
data_df.isnull().sum()
```

	0
Account	0
Coin	0
Execution Price	0
Size Tokens	0
Size USD	0
Side	0
Timestamp IST	0
Start Position	0
Direction	0
Closed PnL	0
Transaction Hash	0
Order ID	0
Crossed	0
Fee	0
Trade ID	0
Timestamp	0

dtype: int64

```
data_df["Timestamp IST"] = pd.to_datetime(data_df["Timestamp IST"], format="%d-%m-%Y %H:%M")
```

```
data_df["Timestamp IST"]
```

	Timestamp IST
0	2024-12-02 22:50:00
1	2024-12-02 22:50:00
2	2024-12-02 22:50:00
3	2024-12-02 22:50:00
4	2024-12-02 22:50:00
...	...
211219	2025-04-25 15:35:00
211220	2025-04-25 15:35:00
211221	2025-04-25 15:35:00
211222	2025-04-25 15:35:00
211223	2025-04-25 15:35:00

211224 rows × 1 columns

dtype: datetime64[ns]

```
data_df["date"] = data_df["Timestamp IST"].dt.date
data_df["date"] = pd.to_datetime(data_df["date"])
```

```
numeric_cols = [
    "Execution Price", "Size Tokens", "Size USD", "Fee",
    "Start Position", "Closed PnL", "Trade ID", "Timestamp"
]
data_df[numeric_cols] = data_df[numeric_cols].apply(pd.to_numeric, errors='coerce')
```

Clean and Prepare sentiment_df

```
sentiment_df.head()
```

	timestamp	value	classification	date
0	1517463000	30	Fear	2018-02-01
1	1517549400	15	Extreme Fear	2018-02-02
2	1517635800	40	Fear	2018-02-03
3	1517722200	24	Extreme Fear	2018-02-04
4	1517808600	11	Extreme Fear	2018-02-05

Next steps: Generate code with sentiment_df | View recommended plots | New interactive sheet

```
sentiment_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2644 entries, 0 to 2643
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   timestamp       2644 non-null   int64
 1   value           2644 non-null   int64
 2   classification  2644 non-null   object
 3   date            2644 non-null   object
dtypes: int64(2), object(2)
memory usage: 82.8+ KB
```

```
sentiment_df.isnull().sum()
```

	0
timestamp	0
value	0
classification	0
date	0

dtype: int64

```
sentiment_df["date"] = pd.to_datetime(sentiment_df["date"], errors="coerce")
```

Merge DataFrames

```
merged_df = pd.merge(data_df, sentiment_df, on="date", how="left")
```

Analysis & Visualization

```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211224 entries, 0 to 211223
Data columns (total 20 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Account           211224 non-null  object
 1   Coin              211224 non-null  object
 2   Execution Price   211224 non-null  float64
 3   Size Tokens       211224 non-null  float64
 4   Size USD          211224 non-null  float64
 5   Side              211224 non-null  object
 6   Timestamp IST     211224 non-null  datetime64[ns]
 7   Start Position    211224 non-null  float64
 8   Direction         211224 non-null  object
```

```
 9   Closed PnL        211224 non-null  float64
 10  Transaction Hash  211224 non-null  object
 11  Order ID          211224 non-null  int64
 12  Crossed           211224 non-null  bool
 13  Fee               211224 non-null  float64
 14  Trade ID          211224 non-null  float64
 15  Timestamp         211224 non-null  float64
 16  date              211224 non-null  datetime64[ns]
 17  timestamp         211218 non-null  float64
 18  value             211218 non-null  float64
 19  classification    211218 non-null  object
dtypes: bool(1), datetime64[ns](2), float64(10), int64(1), object(6)
memory usage: 30.8+ MB
```

```
merged_df.describe(include='all')
```

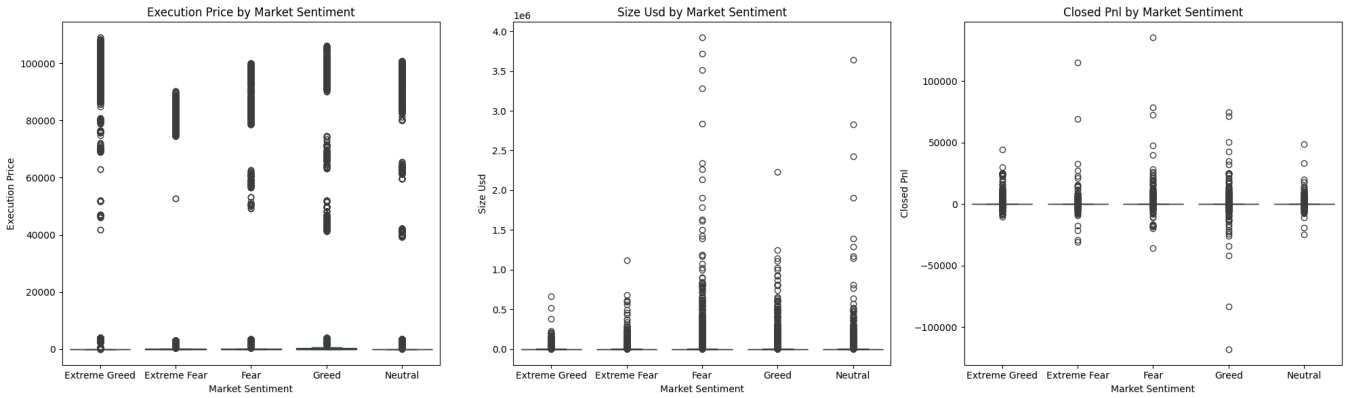
	Account	Coin	Execution Price	Size Tokens	Size USD	Side	Timestamp IST	
count	211224	211224	211224.000000	2.112240e+05	2.112240e+05	211224	211224	2.11
unique	32	246	NaN	NaN	NaN	2	NaN	
top	0xbee1707d6b44d4d52bfe19e41f8a828645437aab	HYPE	NaN	NaN	NaN	SELL	NaN	
freq	40184	68005	NaN	NaN	NaN	108528	NaN	
mean	NaN	NaN	11414.723350	4.623365e+03	5.639451e+03	NaN	2025-01-31 12:04:22.915009792	-2.99
min	NaN	NaN	0.000005	8.740000e-07	0.000000e+00	NaN	2023-05-01 01:06:00	-1.43
25%	NaN	NaN	4.854700	2.940000e+00	1.937900e+02	NaN	2024-12-31 21:00:45	-3.76
50%	NaN	NaN	18.280000	3.200000e+01	5.970450e+02	NaN	2025-02-24 18:55:00	8.47
75%	NaN	NaN	101.580000	1.879025e+02	2.058960e+03	NaN	2025-04-02 18:22:00	9.33
max	NaN	NaN	109004.000000	1.582244e+07	3.921431e+06	NaN	2025-05-01 12:13:00	3.05
std	NaN	NaN	29447.654868	1.042729e+05	3.657514e+04	NaN	NaN	6.73

```
summary_stats = merged_df.groupby('classification').agg({
    'Execution Price': 'mean',
    'Size USD': 'mean',
    'Closed PnL': ['mean', 'sum'],
})
print(summary_stats)
```

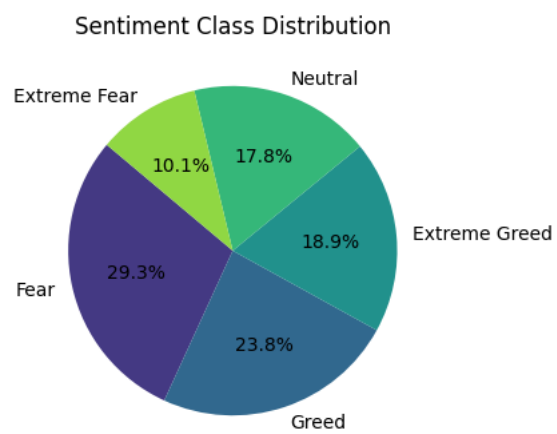
```
                Execution Price     Size USD Closed PnL
                           mean         mean       mean           sum
classification
Extreme Fear       7054.795108  5349.731843  34.537862  7.391102e+05
Extreme Greed      6082.195865  3112.251565  67.892861  2.715171e+06
Fear              14152.620222  7816.109931  54.290400  3.357155e+06
Greed             13411.276344  5736.884375  42.743559  2.150129e+06
Neutral           12393.692779  4782.732661  34.307718  1.292921e+06
```

```
import os
os.makedirs('outputs',exist_ok=True)
```

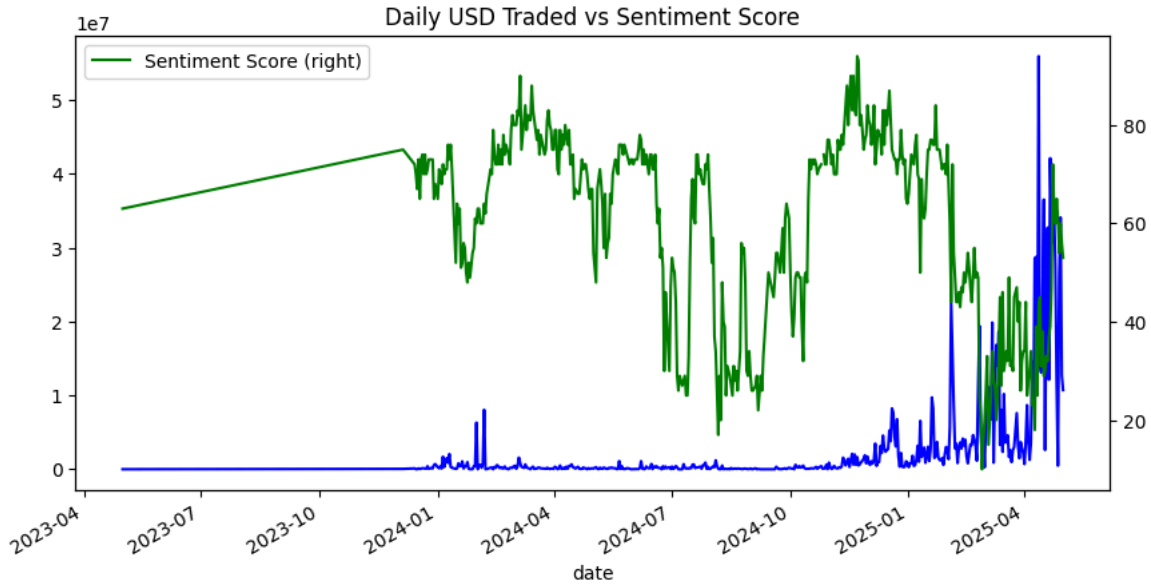
```
# Box plots of key metrics by Sentiment Classification

cols = ['Execution Price', 'Size USD', 'Closed PnL']
plt.figure(figsize=(20, 6))
for i, col in enumerate(cols, 1):
    plt.subplot(1, len(cols), i)
    sns.boxplot(data=merged_df, x='classification', y=col)
    plt.title(f'{col.replace("_", " ").title()} by Market Sentimen
    plt.xlabel('Market Sentiment')
    plt.ylabel(col.replace("_", " ").title())

plt.tight_layout()
plt.savefig("outputs/boxplots_summary_stats.png")
plt.show()
```



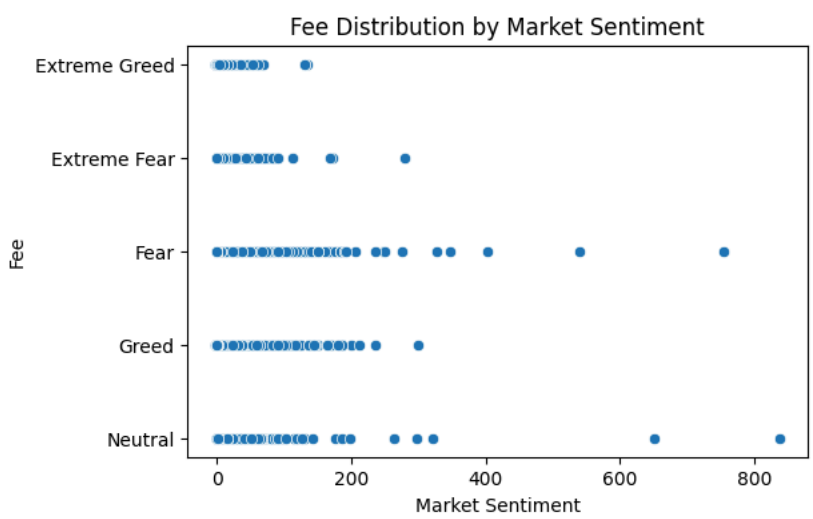

```
# Sentiment Class Distribution
sentiment_counts = merged_df['classification'].value_counts()

plt.figure(figsize=(6, 4))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%', startangle=140, colors=sns.color_palette('viridis', len(sent
plt.title("Sentiment Class Distribution")
plt.savefig("outputs/sentiment_distribution.png")
plt.show()
```



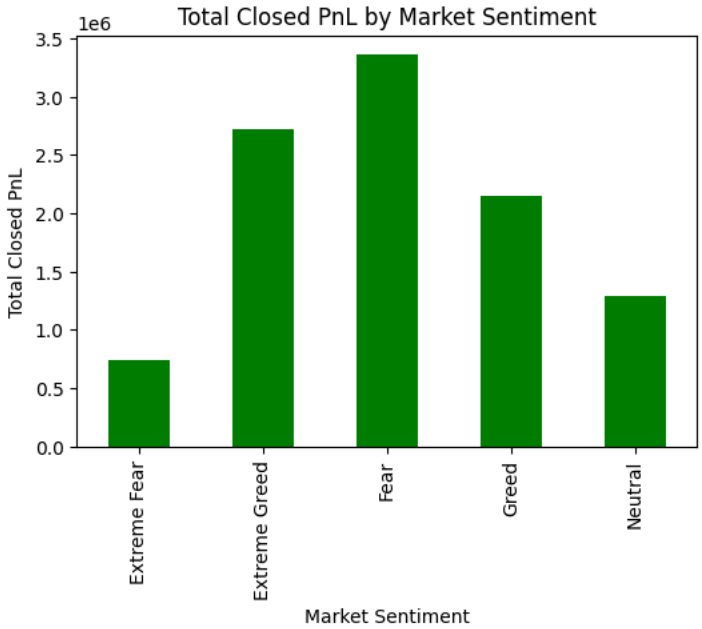

```
# Daily USD Traded vs Sentiment Score
daily_volume = merged_df.groupby("date")["Size USD"].sum()
daily_sentiment = merged_df.groupby("date")["value"].first()

plt.figure(figsize=(10, 5))
daily_volume.plot(label="USD Volume", color='blue')
daily_sentiment.plot(secondary_y=True, label="Sentiment Score", color='green')
plt.title("Daily USD Traded vs Sentiment Score")
plt.xlabel("Date")
plt.legend()
plt.savefig("outputs/daily_volume_vs_sentiment.png")
plt.show()
```



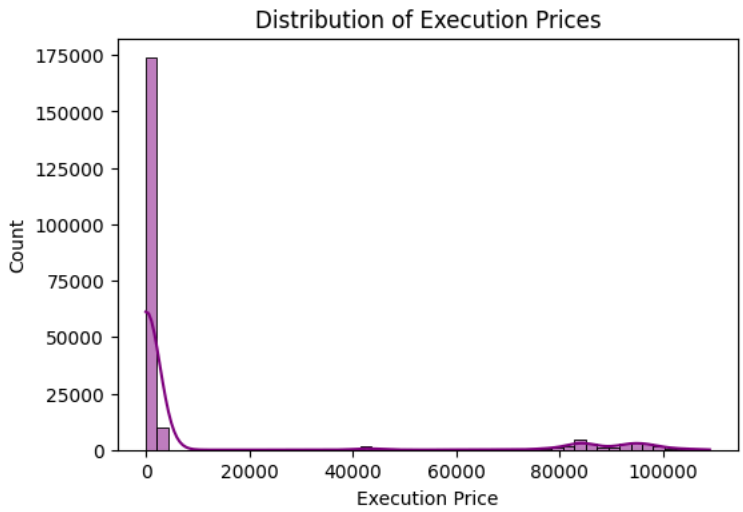

```
#Fee Distribution by Sentiment
plt.figure(figsize=(6,4))
sns.scatterplot(data=merged_df, x="Fee", y="classification")
plt.title("Fee Distribution by Market Sentiment")
plt.xlabel("Market Sentiment")
plt.ylabel("Fee")
plt.savefig("outputs/fee_by_sentiment.png")
plt.show()
```



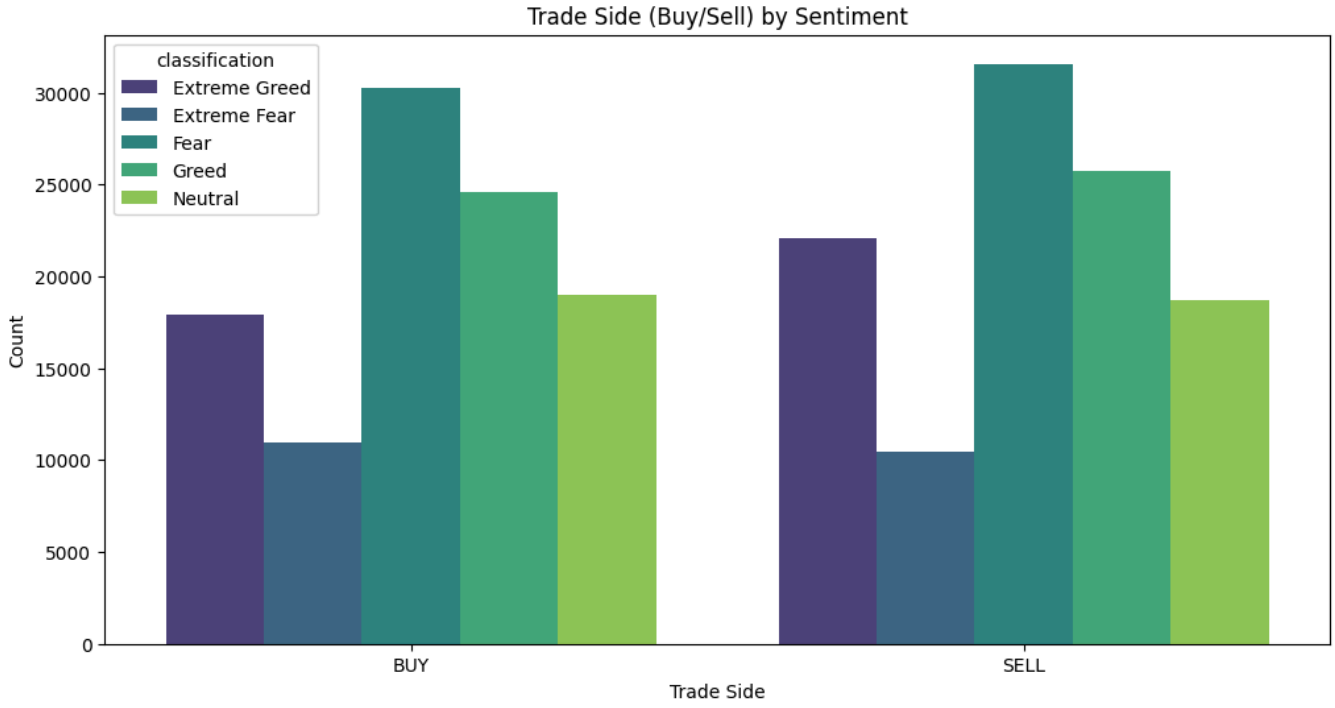

```
# Total Closed PnL by Sentiment
plt.figure(figsize=(6,4))
merged_df.groupby("classification")["Closed PnL"].sum().plot(kind='bar', color='green')
plt.title("Total Closed PnL by Market Sentiment")
plt.xlabel("Market Sentiment")
plt.ylabel("Total Closed PnL")
plt.savefig("outputs/pnl_by_sentiment.png")
plt.show()
```



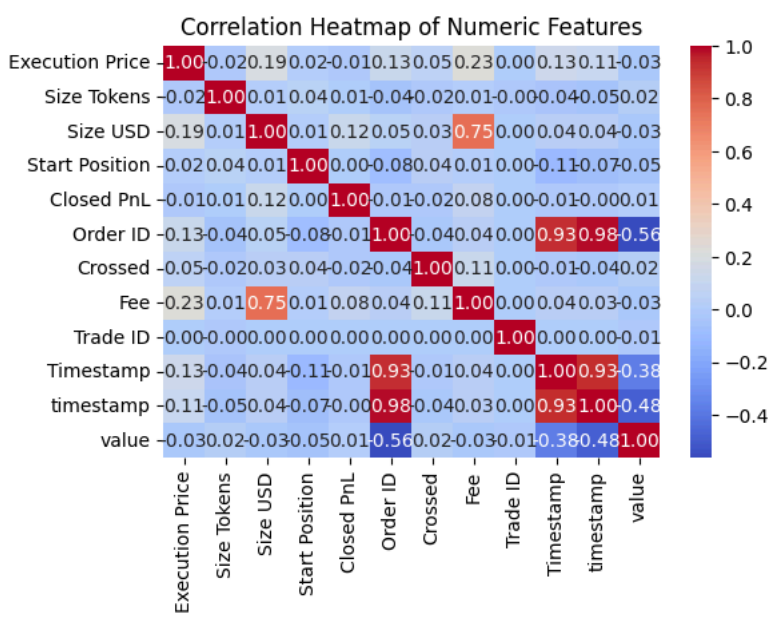

```
#Execution Price Distribution
plt.figure(figsize=(6,4))
sns.histplot(merged_df["Execution Price"],  bins=50,kde='True',color="purple")
plt.title("Distribution of Execution Prices")
plt.xlabel("Execution Price")
plt.savefig("outputs/execution_price_distribution.png")
plt.show()
```



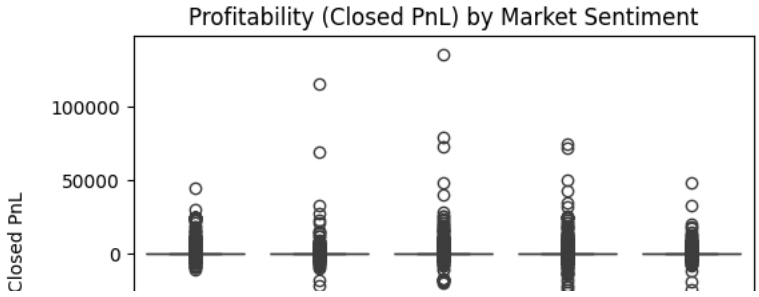

```
# Trade Side Counts by Sentiment
plt.figure(figsize=(12,6))
sns.countplot(data=merged_df, x="Side", hue="classification",palette='viridis')
plt.title("Trade Side (Buy/Sell) by Sentiment")
plt.xlabel("Trade Side")
plt.ylabel("Count")
plt.savefig("outputs/side_by_sentiment.png")
plt.show()
```




```
#Correlation Heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(merged_df.corr(numeric_only=True), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap of Numeric Features")
plt.savefig("outputs/correlation_heatmap.png")
plt.show()
```




```
# Profitability (Closed PnL) by Market Sentiment
plt.figure(figsize=(6, 4))
sns.boxplot(data=merged_df, x='classification', y='Closed PnL')
plt.title("Profitability (Closed PnL) by Market Sentiment")
plt.xlabel("Market Sentiment")
plt.ylabel("Closed PnL")
plt.savefig("outputs/profitability_by_sentiment_from_combined.png")
plt.show()
```




```
#pairplot
sampled_df = merged_df.sample(n=1000, random_state=42)  # Sample 1000 rows from the entire DataFrame
sns.pairplot(sampled_df[numeric_cols + ['classification']], hue='classification', palette='Set2', diag_kind='kde', plot_kws={'alpha':0.!
plt.suptitle("Pairplot of Numeric Features by Market Sentiment (Sampled Data)", y=1.02)
plt.savefig("outputs/pairplot_by_sentiment.png")
plt.show()
```