Assignment-Trader Behavior Insights

Introduction: "To analyze the relationship between trader performance (from historical data) and market sentiment (from the Fear & Greed Index) to uncover patterns and propose data-driven trading strategies.

Step 1: Load the Datasets

In [ ]:
```
#  Load the Datasets
import pandas as pd

# Load Fear & Greed Index data
fear_greed_df = pd.read_csv('/Users/rakeshpathlavath/Desktop/internshala

# Load Historical Trader data
historical_trades_df = pd.read_csv('/Users/rakeshpathlavath/Desktop/inter

# Display first few rows for verification
print("\n--- Fear & Greed Index Data (First 5 Rows) ---")
display(fear_greed_df.head())

print("\n--- Historical Trader Data (First 5 Rows) ---")
display(historical_trades_df.head())
```

```
--- Fear & Greed Index Data (First 5 Rows) ---
```

	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

```
--- Historical Trader Data (First 5 Rows) ---
```

	Account	Coin	Execution Price	Size Tokens	Size USD
0	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9769	986.87	7872.16
1	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9800	16.00	127.68
2	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9855	144.09	1150.63
3	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9874	142.98	1142.04
4	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9894	8.73	69.75

Step 2: Data Cleaning and Preprocessing

In [ ]:
```
#  Data Cleaning and Preprocessing

# Convert 'date' column in Fear & Greed data to datetime
fear_greed_df['date'] = pd.to_datetime(fear_greed_df['date'])

# Convert 'Timestamp IST' in Historical Trades to datetime and create a n
historical_trades_df['Timestamp IST'] = pd.to_datetime(historical_trades_
historical_trades_df['date'] = historical_trades_df['Timestamp IST'].dt.n

# Verify transformations
print("\n--- Verification of Data Types ---")
fear_greed_df.info()
historical_trades_df.info()

print("\n--- Preview of Cleaned Date Columns ---")
display(historical_trades_df[['Timestamp IST', 'date']].head())
```

```
--- Verification of Data Types ---
<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   datetime64[ns]
dtypes: datetime64[ns](1), int64(2), object(1)
memory usage: 82.8+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211224 entries, 0 to 211223
Data columns (total 17 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]
dtypes: bool(1), datetime64[ns](2), float64(8), int64(1), object(5)
memory usage: 26.0+ MB

--- Preview of Cleaned Date Columns ---
```

	Timestamp IST	date
0	2024-12-02 22:50:00	2024-12-02
1	2024-12-02 22:50:00	2024-12-02
2	2024-12-02 22:50:00	2024-12-02
3	2024-12-02 22:50:00	2024-12-02
4	2024-12-02 22:50:00	2024-12-02

In [ ]:
```
#  Merge the Datasets

# Merge historical trades with sentiment data using the common 'date' col
merged_df = pd.merge(historical_trades_df, fear_greed_df, on='date', how=

# Verify merge success
print("\n--- Merged DataFrame Info ---")
merged_df.info()

print("\n--- Sample of Merged Columns ---")
display(merged_df[['Timestamp IST', 'Side', 'Closed PnL', 'date', 'classi

# Check for trades with missing sentiment data
missing_sentiment_count = merged_df['classification'].isnull().sum()
print(f"\nTrades without matching sentiment data: {missing_sentiment_coun

# Drop rows missing sentiment classification (if any)
if missing_sentiment_count > 0:
    merged_df.dropna(subset=['classification'], inplace=True)
    print(f"After dropping missing sentiment rows → Remaining trades: {le
```

```
--- Merged DataFrame 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

--- Sample of Merged Columns ---
```

	Timestamp IST	Side	Closed PnL	date	classification	value
0	2024-12-02 22:50:00	BUY	0.0	2024-12-02	Extreme Greed	80.0
1	2024-12-02 22:50:00	BUY	0.0	2024-12-02	Extreme Greed	80.0
2	2024-12-02 22:50:00	BUY	0.0	2024-12-02	Extreme Greed	80.0
3	2024-12-02 22:50:00	BUY	0.0	2024-12-02	Extreme Greed	80.0
4	2024-12-02 22:50:00	BUY	0.0	2024-12-02	Extreme Greed	80.0

```
Trades without matching sentiment data: 6
After dropping missing sentiment rows → Remaining trades: 211218
```

step 3: Exploratory Data Analysis (EDA)

In [ ]:
```
#  Basic Exploratory Data Analysis (EDA) ===

# Check for missing values
print("\n=== Missing Values per Column ===")
print(merged_df.isnull().sum())

# Inspect unique categories for key categorical columns
print("\n=== Unique Values in Categorical Columns ===")
for col in ['Side', 'Direction', 'classification']:
    if col in merged_df.columns:
        print(f"{col}: {merged_df[col].unique()}")

# Generate descriptive statistics for numeric features
print("\n=== Summary Statistics (Numerical Columns) ===")
```

```
print(merged_df.describe())

# Check sentiment group distribution
print("\n=== Sentiment Classification Distribution ===")
print(merged_df['classification'].value_counts())
```

```
=== Missing Values per Column ===
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
date                0
timestamp           0
value               0
classification      0
dtype: int64

=== Unique Values in Categorical Columns ===
Side: ['BUY' 'SELL']
Direction: ['Buy' 'Sell' 'Open Long' 'Close Long' 'Spot Dust Conversion'
'Open Short'
 'Close Short' 'Long > Short' 'Short > Long' 'Auto-Deleveraging'
 'Liquidated Isolated Short' 'Settlement']
classification: ['Extreme Greed' 'Extreme Fear' 'Fear' 'Greed' 'Neutral']

=== Summary Statistics (Numerical Columns) ===
       Execution Price   Size Tokens      Size USD  \
count    211218.000000  2.112180e+05  2.112180e+05
mean      11415.047529  4.623341e+03  5.639192e+03
min           0.000005  8.740000e-07  0.000000e+00
25%           4.858550  2.940000e+00  1.937900e+02
50%          18.280000  3.200000e+01  5.970200e+02
75%         101.895000  1.878900e+02  2.058878e+03
max      109004.000000  1.582244e+07  3.921431e+06
std       29448.010305  1.042744e+05  3.657557e+04

                      Timestamp IST  Start Position     Closed PnL  \
count                        211218    2.112180e+05  211218.000000
mean   2025-01-31 12:08:21.724569344   -2.994671e+04      48.549304
min              2023-05-01 01:06:00   -1.433463e+07 -117990.104100
25%              2024-12-31 21:53:45   -3.760725e+02       0.000000
50%              2025-02-24 18:55:00    8.477051e+01       0.000000
75%              2025-04-02 18:22:00    9.337697e+03       5.790132
max              2025-05-01 12:13:00    3.050948e+07  135329.090100
std                             NaN    6.738170e+05     917.989791

           Order ID            Fee      Trade ID     Timestamp  \
count  2.112180e+05  211218.000000  2.112180e+05  2.112180e+05
mean   6.965470e+10       1.163960  5.628506e+14  1.737745e+12
min    1.732711e+08      -1.175712  0.000000e+00  1.680000e+12
25%    5.984223e+10       0.016121  2.810000e+14  1.740000e+12
50%    7.442939e+10       0.089572  5.620000e+14  1.740000e+12
75%    8.335543e+10       0.393774  8.460000e+14  1.740000e+12
max    9.014923e+10     837.471593  1.130000e+15  1.750000e+12
```

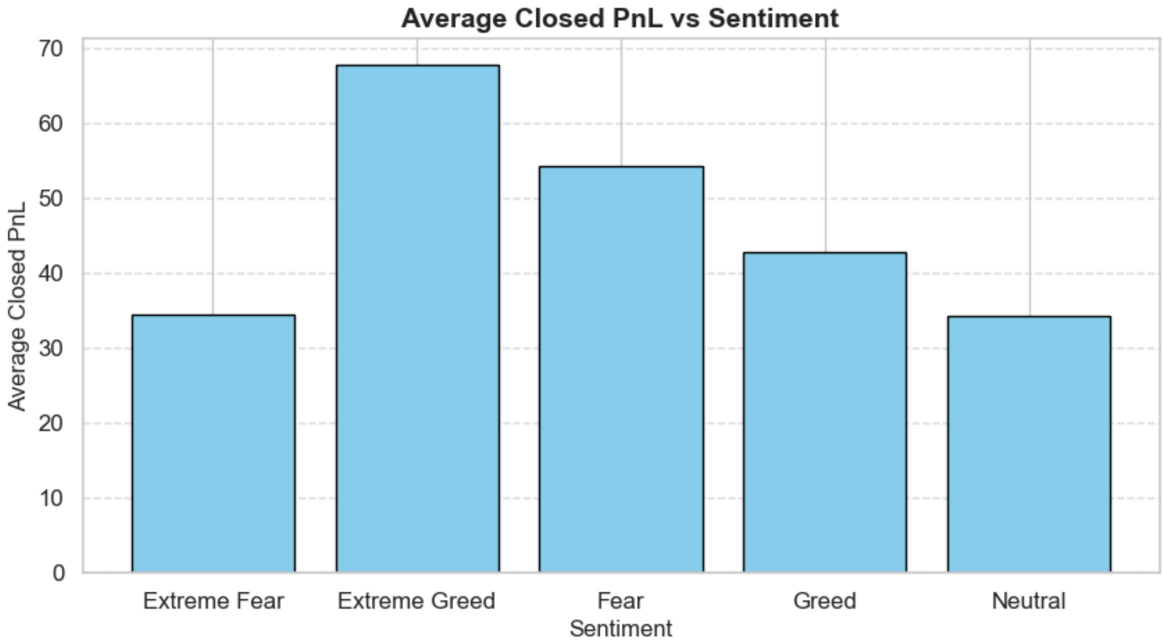
```
std    1.835714e+10       6.758948  3.257541e+14  8.689946e+09

                                 date     timestamp          value
count                          211218  2.112180e+05  211218.000000
mean   2025-01-30 23:58:26.735221248  1.738301e+09      51.649656
min              2023-05-01 00:00:00  1.682919e+09      10.000000
25%              2024-12-31 00:00:00  1.735623e+09      33.000000
50%              2025-02-24 00:00:00  1.740375e+09      49.000000
75%              2025-04-02 00:00:00  1.743572e+09      72.000000
max              2025-05-01 00:00:00  1.746077e+09      94.000000
std                               NaN  8.029302e+06      21.012784

=== Sentiment Classification Distribution ===
classification
Fear             61837
Greed            50303
Extreme Greed    39992
Neutral          37686
Extreme Fear     21400
Name: count, dtype: int64
```

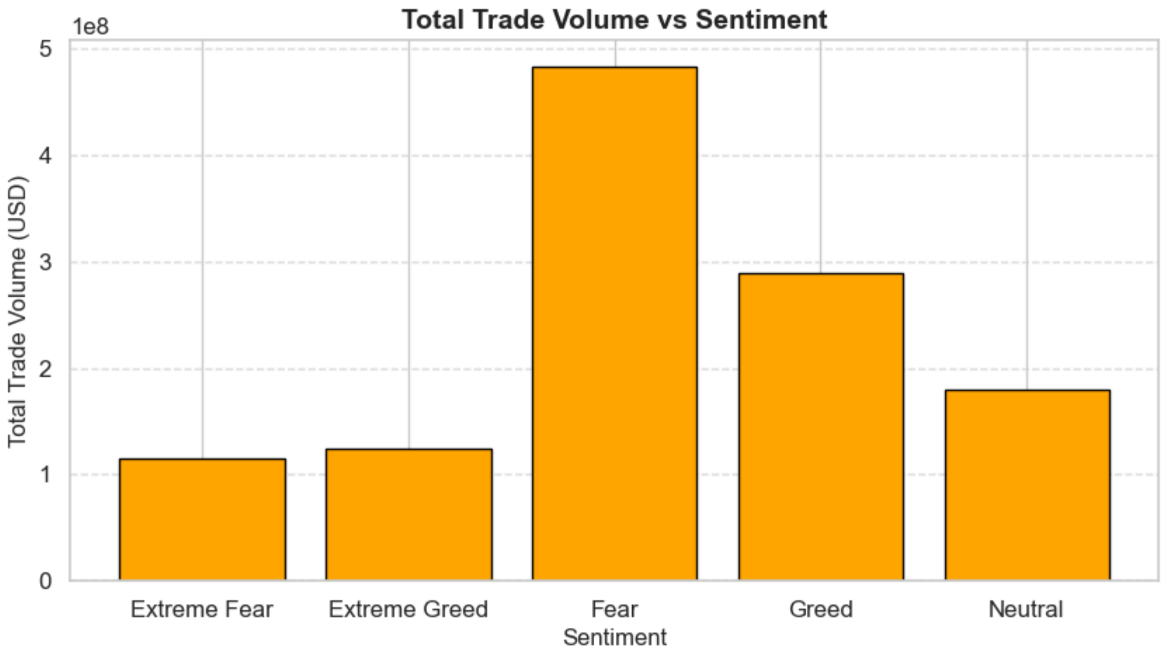
In [ ]:

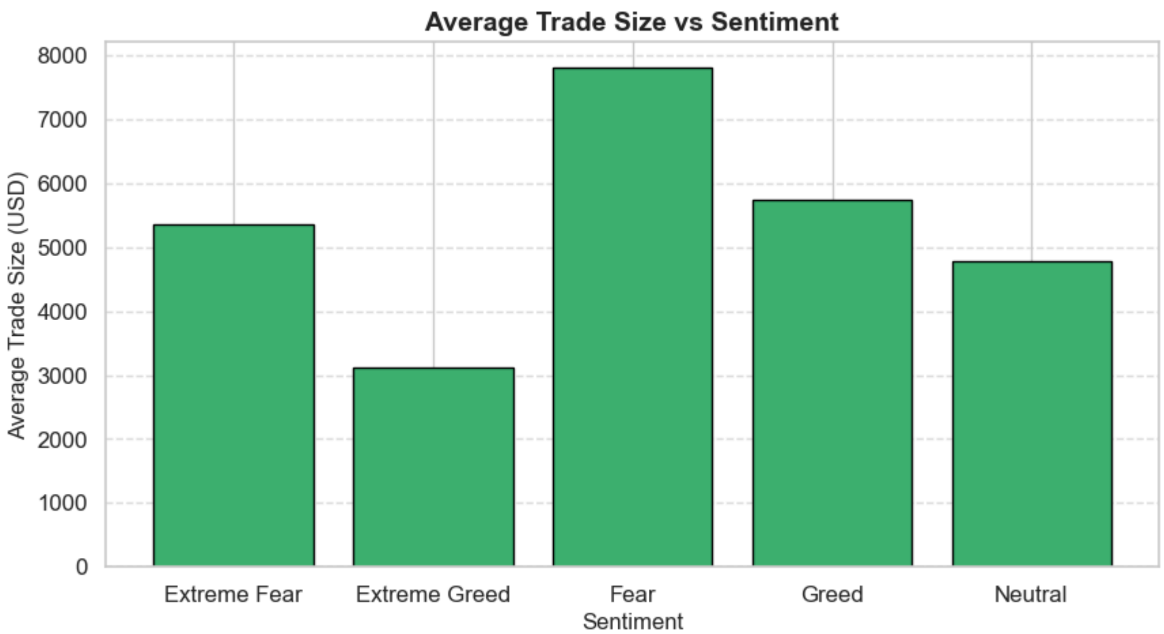
```python
#  Sentiment vs Trader Performance ===

# Aggregate key performance metrics by sentiment classification
sentiment_stats = (
    merged_df.groupby('classification')
    .agg({
        'Closed PnL': ['mean', 'sum'],
        'Size USD': ['mean', 'sum'],
        'Fee': ['mean'],
        'Side': lambda x: x.value_counts(normalize=True).to_dict()
    })
    .reset_index()
)

# Rename columns for readability
sentiment_stats.columns = [
    'Sentiment',
    'Avg PnL',
    'Total PnL',
    'Avg Trade Size USD',
    'Total Trade Volume USD',
    'Avg Fee',
    'Side Distribution'
]

# Display summarized sentiment performance
print("\n=== Sentiment Performance Summary ===")
print(sentiment_stats)
```

```
=== Sentiment Performance Summary ===
        Sentiment    Avg PnL     Total PnL  Avg Trade Size USD  \
0    Extreme Fear  34.537862  7.391102e+05         5349.731843
1   Extreme Greed  67.892861  2.715171e+06         3112.251565
2            Fear  54.290400  3.357155e+06         7816.109931
3           Greed  42.743559  2.150129e+06         5736.884375
4         Neutral  34.307718  1.292921e+06         4782.732661

   Total Trade Volume USD   Avg Fee  \
0            1.144843e+08  1.116291
1            1.244652e+08  0.675902
2            4.833248e+08  1.495172
3            2.885825e+08  1.254372
4            1.802421e+08  1.044798

                                   Side Distribution
0  {'BUY': 0.5109813084112149, 'SELL': 0.48901869...
1  {'SELL': 0.5514102820564113, 'BUY': 0.44858971...
2  {'SELL': 0.5104872487345764, 'BUY': 0.48951275...
3  {'SELL': 0.5114406695425721, 'BUY': 0.48855933...
4  {'BUY': 0.5033434166533991, 'SELL': 0.49665658...
```

In [ ]:
```python
#  Visualization of Key Insights ===
import matplotlib.pyplot as plt

#  1 .Average Closed PnL vs Sentiment
plt.figure(figsize=(10, 5))
plt.bar(sentiment_stats['Sentiment'], sentiment_stats['Avg PnL'], color='
plt.title('Average Closed PnL vs Sentiment', fontsize=14, fontweight='bol
plt.xlabel('Sentiment', fontsize=12)
plt.ylabel('Average Closed PnL', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.show()

#  2 . Total Trade Volume vs Sentiment
plt.figure(figsize=(10, 5))
plt.bar(sentiment_stats['Sentiment'], sentiment_stats['Total Trade Volume
plt.title('Total Trade Volume vs Sentiment', fontsize=14, fontweight='bol
plt.xlabel('Sentiment', fontsize=12)
plt.ylabel('Total Trade Volume (USD)', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.show()

#  3 . Average Trade Size vs Sentiment
plt.figure(figsize=(10, 5))
plt.bar(sentiment_stats['Sentiment'], sentiment_stats['Avg Trade Size USD
plt.title('Average Trade Size vs Sentiment', fontsize=14, fontweight='bol
plt.xlabel('Sentiment', fontsize=12)
plt.ylabel('Average Trade Size (USD)', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.show()
```


Average Closed PnL vs Sentiment
Average Closed PnL
70
60
50
40
30
20
10
0
Extreme Fear
Extreme Greed
Fear
Greed
Neutral
Sentiment


Total Trade Volume vs Sentiment
1e8
Total Trade Volume (USD)
5
4
3
2
1
0
Extreme Fear
Extreme Greed
Fear
Greed
Neutral
Sentiment


Average Trade Size vs Sentiment
Average Trade Size (USD)
8000
7000
6000
5000
4000
3000
2000
1000
0
Extreme Fear
Extreme Greed
Fear
Greed
Neutral
Sentiment

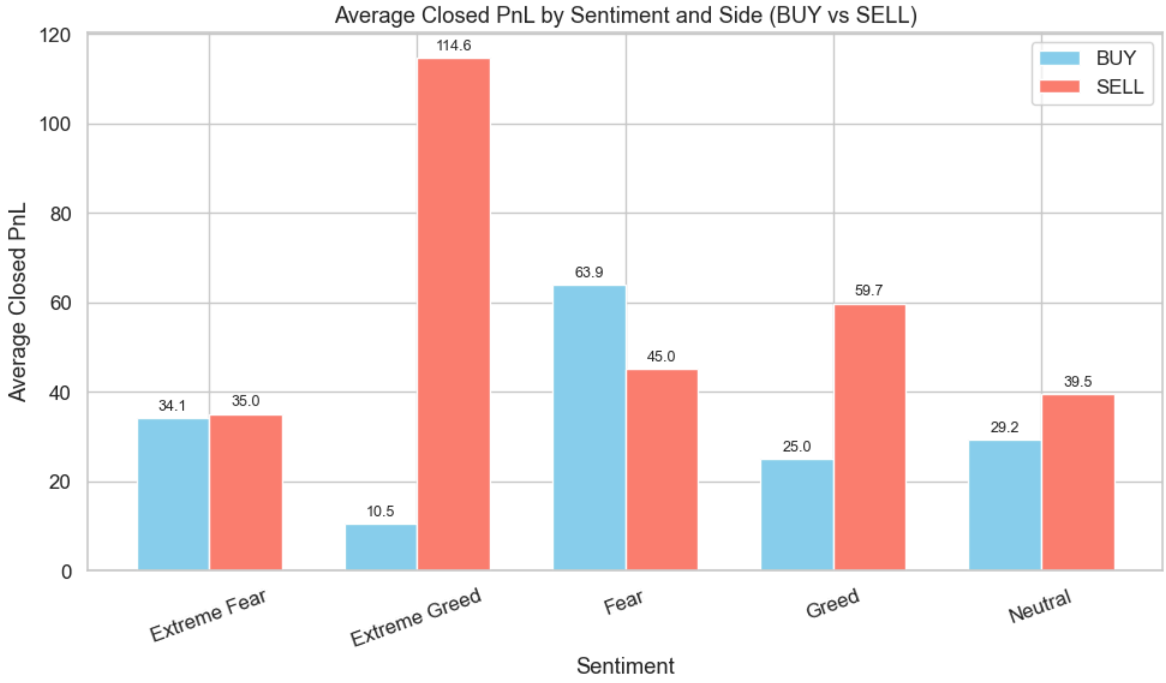
In [ ]:
```
#  Buy/Sell Behavior vs Sentiment

print("Analyzing Buy/Sell performance under different sentiment condition

# Group by sentiment classification and trade side
buy_sell_perf = (
    merged_df
    .groupby(['classification', 'Side'])
    .agg({
        'Closed PnL': ['mean', 'sum'],
        'Size USD': ['mean', 'sum']
    })
    .reset_index()
)

# Clean up column names for readability
buy_sell_perf.columns = [
    'Sentiment', 'Side',
    'Avg PnL', 'Total PnL',
    'Avg Trade Size USD', 'Total Trade Volume USD'
]

# Display results
print("=== Buy/Sell Performance by Sentiment ===")
print(buy_sell_perf)
```

```
Analyzing Buy/Sell performance under different sentiment conditions...

=== Buy/Sell Performance by Sentiment ===
        Sentiment  Side     Avg PnL     Total PnL  Avg Trade Size USD  \
0    Extreme Fear   BUY   34.114627  3.730434e+05         5161.502485
1    Extreme Fear  SELL   34.980106  3.660668e+05         5546.414885
2   Extreme Greed   BUY   10.498927  1.883508e+05         3363.034672
3   Extreme Greed  SELL  114.584643  2.526821e+06         2908.231569
4            Fear   BUY   63.927104  1.935073e+06         8154.666208
5            Fear  SELL   45.049641  1.422082e+06         7491.463987
6           Greed   BUY   25.002302  6.144566e+05         6306.490894
7           Greed  SELL   59.691091  1.535673e+06         5192.761477
8         Neutral   BUY   29.227429  5.544151e+05         3881.410441
9         Neutral  SELL   39.456408  7.385056e+05         5696.190011

   Total Trade Volume USD
0            5.644103e+07
1            5.804323e+07
2            6.033284e+07
3            6.413232e+07
4            2.468417e+08
5            2.364830e+08
6            1.549883e+08
7            1.335942e+08
8            7.362647e+07
9            1.066156e+08
```

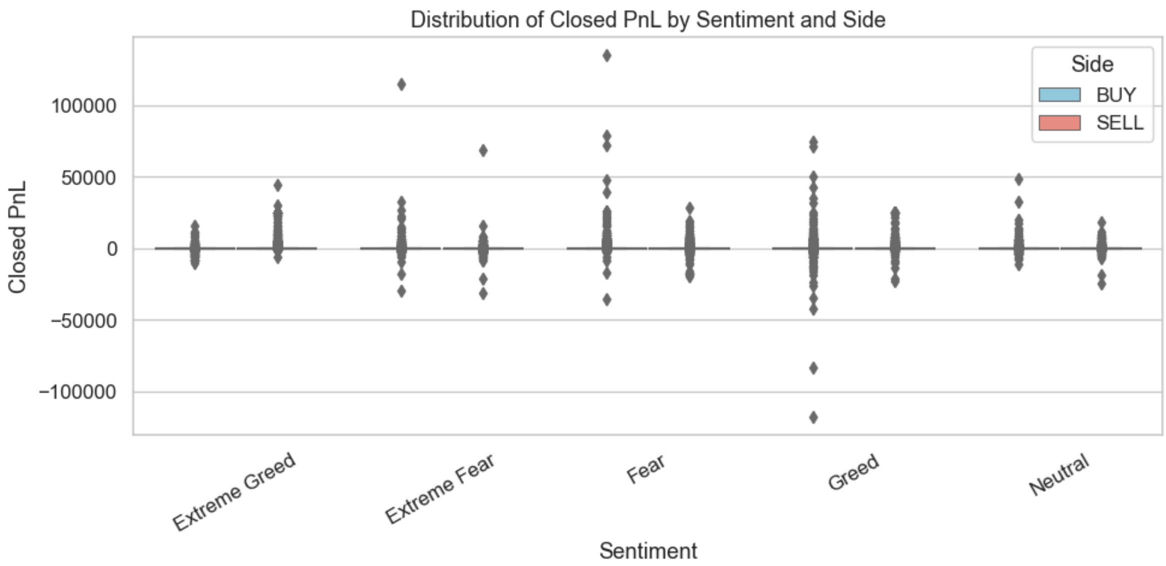
In [ ]:
```
#  Average Closed PnL by Sentiment and Side

import numpy as np
import matplotlib.pyplot as plt

# Compute average Closed PnL by sentiment and side
avg_pnl_by_side = (
```

```
    merged_df.groupby(['classification', 'Side'])['Closed PnL']
    .mean()
    .unstack()[['BUY', 'SELL']]
)

# Display summary table
print("=== Avg Closed PnL (by Sentiment and Side) ===")
print(avg_pnl_by_side.round(4))

# --- Visualization ---
labels = avg_pnl_by_side.index.tolist()
x = np.arange(len(labels))
width = 0.35

plt.figure(figsize=(10, 6))
bars_buy = plt.bar(x - width / 2, avg_pnl_by_side['BUY'], width, label='B
bars_sell = plt.bar(x + width / 2, avg_pnl_by_side['SELL'], width, label=

plt.title('Average Closed PnL by Sentiment and Side (BUY vs SELL)')
plt.xlabel('Sentiment')
plt.ylabel('Average Closed PnL')
plt.xticks(x, labels, rotation=20)
plt.legend()

# Annotate bars with values
for bars in [bars_buy, bars_sell]:
    for bar in bars:
        height = bar.get_height()
        plt.text(
            bar.get_x() + bar.get_width() / 2,
            height + 1,
            f'{height:.1f}',
            ha='center',
            va='bottom',
            fontsize=9
        )

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

```
=== Avg Closed PnL (by Sentiment and Side) ===
Side                BUY      SELL
classification
Extreme Fear    34.1146   34.9801
Extreme Greed   10.4989  114.5846
Fear            63.9271   45.0496
Greed           25.0023   59.6911
Neutral         29.2274   39.4564
```



In [ ]:
```
#  Distribution of Closed PnL by Sentiment and Side

import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
sns.boxplot(
    x="classification",
    y="Closed PnL",
    hue="Side",
    data=merged_df,
    palette={"BUY": "skyblue", "SELL": "salmon"}
)

plt.title("Distribution of Closed PnL by Sentiment and Side", fontsize=13
plt.xlabel("Sentiment")
plt.ylabel("Closed PnL")
plt.xticks(rotation=30)
plt.legend(title="Side")
plt.tight_layout()
plt.show()
```



In [ ]:
```
#  ANOVA Test – Sentiment Impact on Closed PnL

from scipy import stats

# Prepare Closed PnL data grouped by sentiment classification
groups = [
    merged_df.loc[merged_df['classification'] == sentiment, 'Closed PnL']
    for sentiment in merged_df['classification'].unique()
]

# Perform one-way ANOVA
f_stat, p_val = stats.f_oneway(*groups)

# Display results
print(f"ANOVA F-statistic: {f_stat:.4f}")
print(f"P-value: {p_val:.6f}")

# Interpretation
if p_val < 0.05:
    print("\n Significant difference found between at least two sentiment
else:
    print("\n No significant difference found – PnL means across sentimen
```

```
ANOVA F-statistic: 9.0622
P-value: 0.000000

 Significant difference found between at least two sentiment groups (p <
0.05).
```

In [ ]:
```
#  T-Test – BUY vs SELL Mean Closed PnL Comparison

from scipy.stats import ttest_ind

# Separate BUY and SELL Closed PnL values
buy_pnl = merged_df.loc[merged_df['Side'] == 'BUY', 'Closed PnL']
sell_pnl = merged_df.loc[merged_df['Side'] == 'SELL', 'Closed PnL']

# Perform independent t-test (Welch's t-test assumes unequal variance)
t_stat, p_val = ttest_ind(buy_pnl, sell_pnl, equal_var=False)

# Display results
print(f"T-statistic: {t_stat:.4f}")
```

```
print(f"P-value: {p_val:.6f}")

# Interpretation
if p_val < 0.05:
    print("\n Significant difference between BUY and SELL mean PnL (p < 0
else:
    print("\n No significant difference between BUY and SELL mean PnL.")
```

```
T-statistic: -6.1881
P-value: 0.000000

 Significant difference between BUY and SELL mean PnL (p < 0.05).
```

In [ ]:

```
#  Trading Behavior Summary by Sentiment

# Aggregate key trade metrics grouped by sentiment
behavior_summary = merged_df.groupby('classification').agg({
    'Size USD': ['mean', 'sum'],
    'Fee': 'mean',
    'Account': 'count'
}).reset_index()

# Rename columns for readability
behavior_summary.columns = [
    'Sentiment',
    'Avg Trade Size USD',
    'Total Trade Volume USD',
    'Avg Fee',
    'Trade Count'
]

# Display neatly sorted summary
print("\n=== Trading Behavior Summary by Sentiment ===")
print(behavior_summary.sort_values('Sentiment'))
```

```
=== Trading Behavior Summary by Sentiment ===
        Sentiment  Avg Trade Size USD  Total Trade Volume USD   Avg Fee  \
0    Extreme Fear         5349.731843            1.144843e+08  1.116291
1   Extreme Greed         3112.251565            1.244652e+08  0.675902
2            Fear         7816.109931            4.833248e+08  1.495172
3           Greed         5736.884375            2.885825e+08  1.254372
4         Neutral         4782.732661            1.802421e+08  1.044798

   Trade Count
0        21400
1        39992
2        61837
3        50303
4        37686
```

In [ ]:

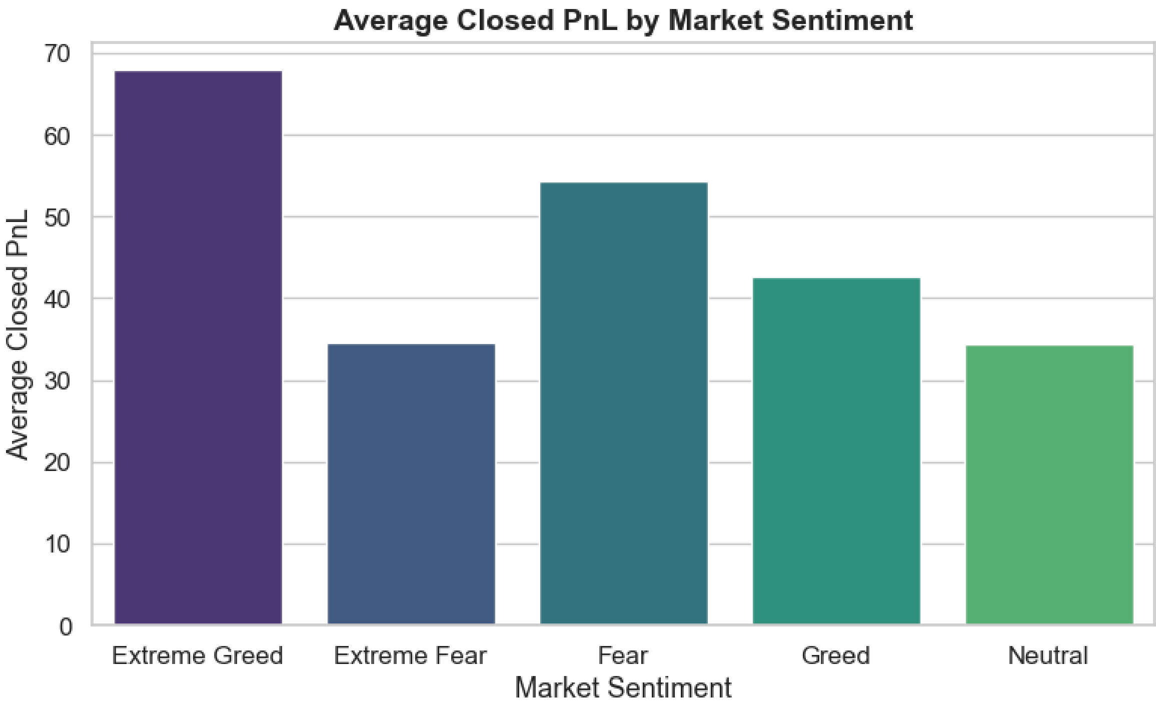
```
#  Trade Frequency and PnL Volatility by Sentiment

# Extract only the date from Timestamp for grouping
merged_df['Trade Date'] = merged_df['Timestamp IST'].dt.date

# Count trades and measure volatility (std of PnL) per day, grouped by se
daily_activity = merged_df.groupby(['classification', 'Trade Date']).agg(
    'Account': 'count',
    'Closed PnL': 'std'
}).reset_index()
```

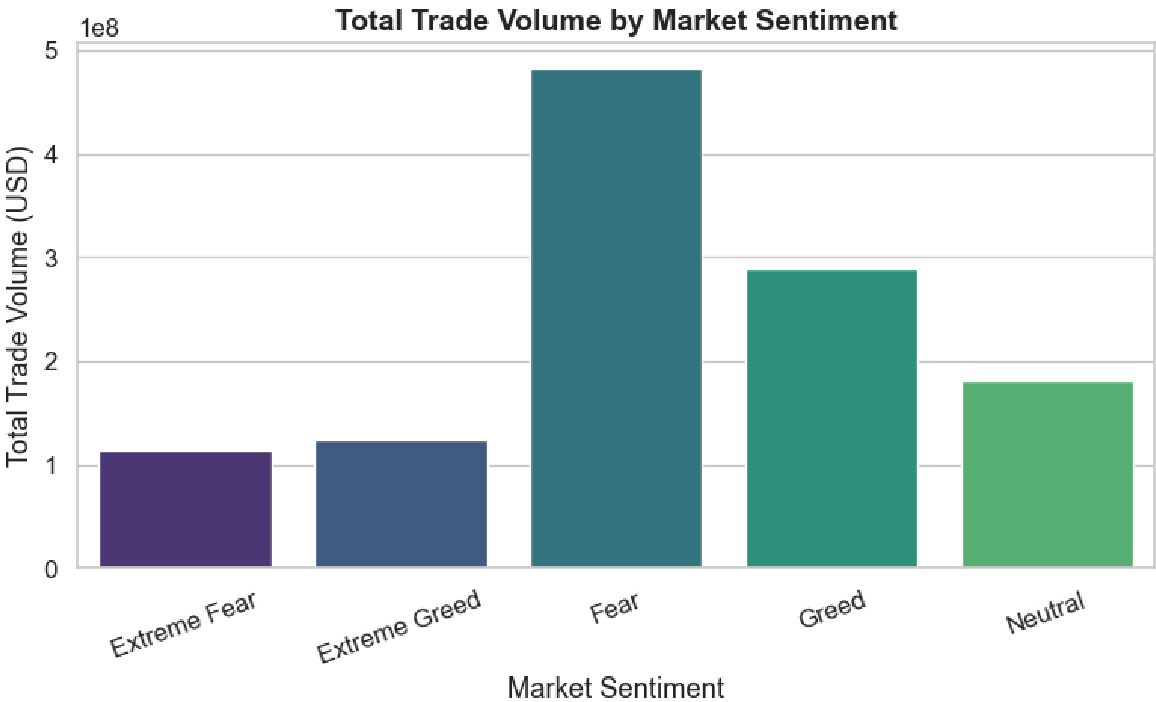
```
# Rename columns for clarity
daily_activity.rename(columns={
    'Account': 'Trade Count',
    'Closed PnL': 'PnL Volatility'
}, inplace=True)

# Compute average daily activity and volatility by sentiment
freq_vol_summary = daily_activity.groupby('classification').agg({
    'Trade Count': 'mean',
    'PnL Volatility': 'mean'
}).reset_index()

# Rename and display
freq_vol_summary.columns = ['Sentiment', 'Avg Daily Trades', 'Avg PnL Vol

print("\n=== Average Daily Trade Activity and PnL Volatility by Sentiment
print(freq_vol_summary.sort_values('Sentiment'))
```

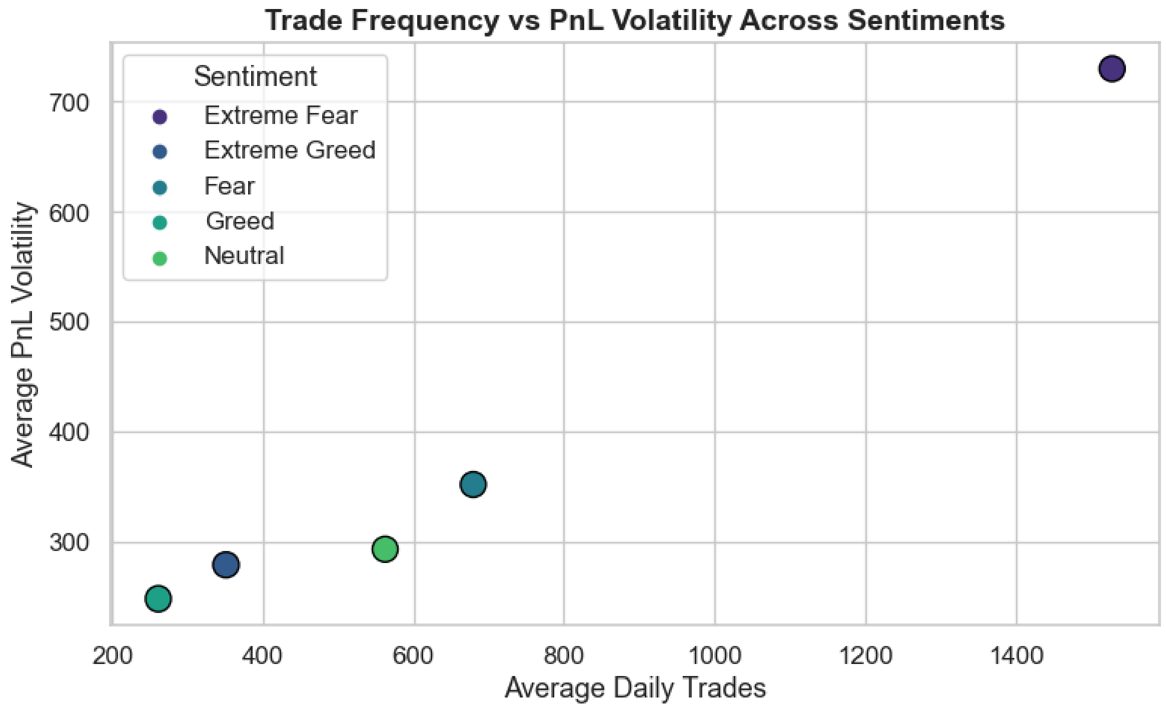
```
=== Average Daily Trade Activity and PnL Volatility by Sentiment ===
       Sentiment  Avg Daily Trades  Avg PnL Volatility
0   Extreme Fear       1528.571429          729.867572
1  Extreme Greed        350.807018          278.656985
2           Fear        679.527473          351.688431
3          Greed        260.637306          247.820567
4        Neutral        562.477612          292.949624
```

In [ ]:
```
#  Visualization of Sentiment-Based Trading Behavior
import matplotlib.pyplot as plt
import seaborn as sns

# Set consistent visual theme
sns.set(style="whitegrid", palette="viridis", font_scale=1.1)

# 1. Average Closed PnL by Sentiment
plt.figure(figsize=(8, 5))
sns.barplot(x='classification', y='Closed PnL', data=merged_df, estimator
plt.title('Average Closed PnL by Market Sentiment', fontsize=14, fontweig
plt.xlabel('Market Sentiment')
plt.ylabel('Average Closed PnL')
plt.tight_layout()
plt.show()

#  2 . Total Trade Volume vs Sentiment
plt.figure(figsize=(8, 5))
sns.barplot(x='Sentiment', y='Total Trade Volume USD',
            data=behavior_summary.sort_values('Sentiment'))
plt.title('Total Trade Volume by Market Sentiment', fontsize=14, fontweig
plt.xlabel('Market Sentiment')
plt.ylabel('Total Trade Volume (USD)')
plt.xticks(rotation=20)
plt.tight_layout()
plt.show()

#  3 . Trade Frequency vs PnL Volatility
plt.figure(figsize=(8, 5))
sns.scatterplot(x='Avg Daily Trades', y='Avg PnL Volatility',
                hue='Sentiment', data=freq_vol_summary, s=150, edgecolor=
plt.title('Trade Frequency vs PnL Volatility Across Sentiments', fontsize
plt.xlabel('Average Daily Trades')
plt.ylabel('Average PnL Volatility')
```

```
plt.legend(title='Sentiment')
plt.tight_layout()
plt.show()
```

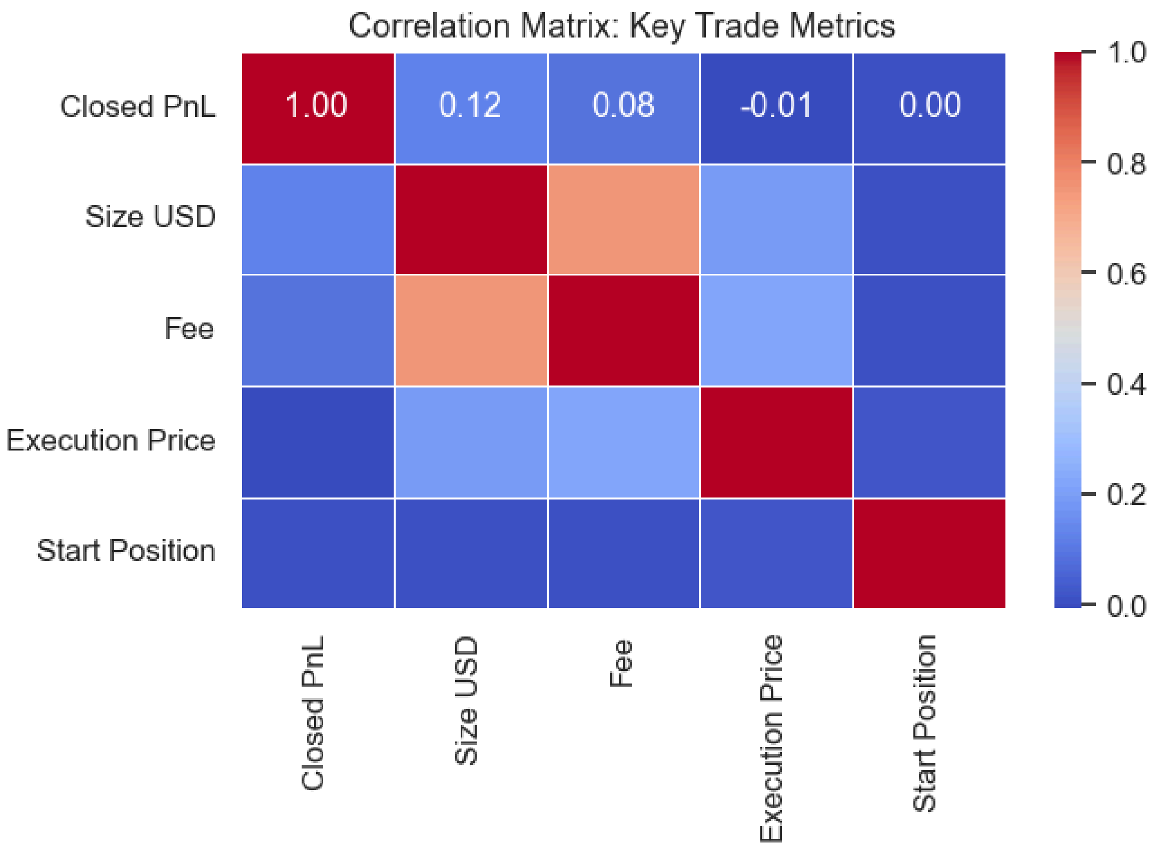






In [ ]:
```
# === New Section 1: Correlation Analysis of Key Trade Metrics ===
import matplotlib.pyplot as plt
import seaborn as sns

# Select relevant numeric features (adjust if any column names differ)
corr_cols = ['Closed PnL', 'Size USD', 'Fee', 'Execution Price', 'Start P
corr_matrix = merged_df[corr_cols].corr()

# Print correlation matrix (rounded for readability)
print("\n=== Correlation Between Trade Metrics ===")
print(corr_matrix.round(3))

# Plot heatmap
plt.figure(figsize=(7,5))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidt
plt.title("Correlation Matrix: Key Trade Metrics")
plt.tight_layout()
plt.show()
```

```
=== Correlation Between Trade Metrics ===
                 Closed PnL  Size USD    Fee  Execution Price  Start Posit
ion
Closed PnL            1.000     0.124  0.084           -0.006           0.
004
Size USD              0.124     1.000  0.746            0.190           0.
008
Fee                   0.084     0.746  1.000            0.225           0.
011
Execution Price      -0.006     0.190  0.225            1.000           0.
017
Start Position        0.004     0.008  0.011            0.017           1.
000
```

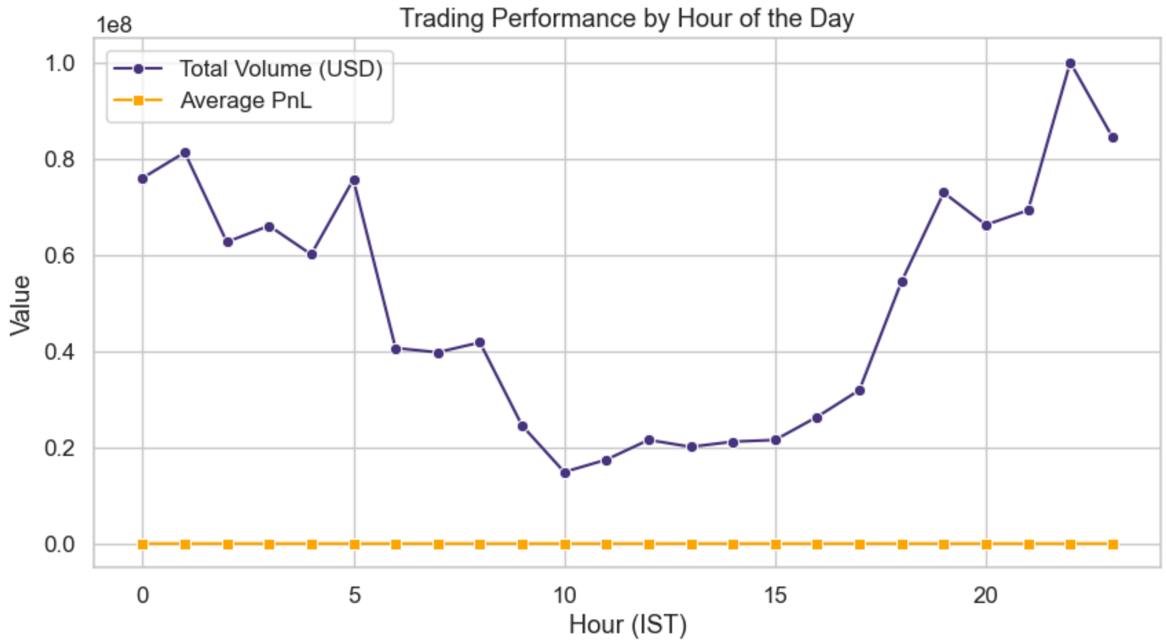


In [ ]:

```
# === New Section 2: Time-of-Day Performance Analysis ===
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Extract hour from Timestamp IST
merged_df['Trade Hour'] = merged_df['Timestamp IST'].dt.hour

# Group by trading hour to analyze PnL and volume
hourly_perf = merged_df.groupby('Trade Hour').agg({
    'Closed PnL': ['mean', 'sum'],
    'Size USD': 'sum',
    'Account': 'count'
}).reset_index()

# Rename columns
hourly_perf.columns = ['Hour', 'Avg PnL', 'Total PnL', 'Total Volume USD'

# Print summary table
print("\n=== Hourly Trading Performance Summary ===")
print(hourly_perf.round(2))

# Plot trade volume and PnL by hour
plt.figure(figsize=(10,5))
sns.lineplot(x='Hour', y='Total Volume USD', data=hourly_perf, marker='o'
sns.lineplot(x='Hour', y='Avg PnL', data=hourly_perf, marker='s', label='
plt.title("Trading Performance by Hour of the Day")
plt.xlabel("Hour (IST)")
plt.ylabel("Value")
plt.legend()
plt.grid(True)
plt.show()
```

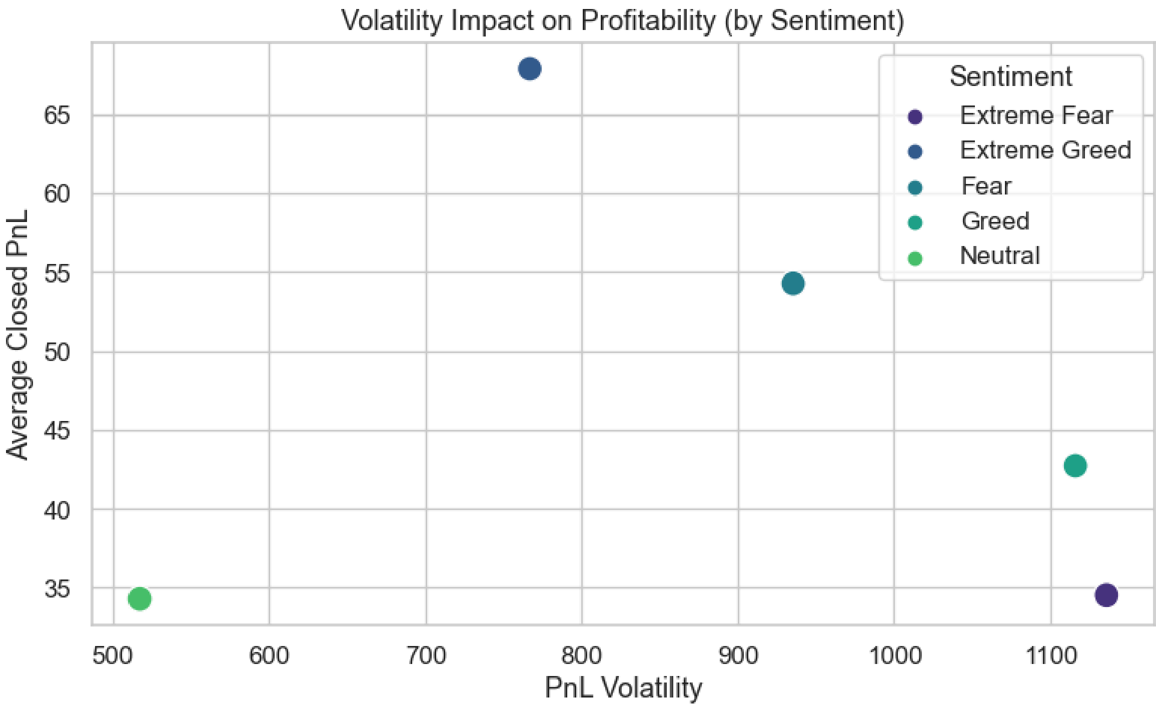
```
=== Hourly Trading Performance Summary ===
    Hour  Avg PnL  Total PnL  Total Volume USD  Trade Count
0      0    43.13  425052.51      7.599172e+07         9856
1      1    49.92  523198.63      8.131113e+07        10481
2      2    34.21  279834.37      6.269316e+07         8181
3      3    43.71  460020.57      6.607090e+07        10524
4      4    44.45  444895.60      6.011737e+07        10009
5      5    40.46  385913.30      7.567222e+07         9538
6      6    32.98  298730.76      4.064307e+07         9057
7      7    83.03  712784.33      3.975941e+07         8585
8      8    58.89  462140.62      4.180750e+07         7848
9      9    44.98  294525.28      2.454014e+07         6548
10    10    61.35  413788.81      1.487741e+07         6745
11    11    76.86  472707.47      1.743643e+07         6150
12    12   131.17  911657.26      2.154751e+07         6950
13    13    52.38  434825.88      2.010831e+07         8301
14    14    23.12  158171.11      2.118058e+07         6840
15    15    58.56  427562.29      2.153527e+07         7301
16    16    41.98  246188.87      2.634997e+07         5865
17    17    34.48  217196.37      3.191403e+07         6299
18    18    44.99  433781.63      5.448169e+07         9641
19    19    55.86  705423.69      7.296070e+07        12628
20    20    49.71  632875.06      6.625425e+07        12731
21    21    31.17  343553.58      6.929642e+07        11022
22    22    37.81  381715.76      1.000332e+08        10096
23    23    18.75  187943.20      8.451636e+07        10022
```

```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [ ]:
```
#  Volatility Impact on Profitability
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Group by sentiment to measure volatility and performance
volatility_impact = merged_df.groupby('classification').agg({
    'Closed PnL': ['mean', 'std'],
    'Size USD': 'mean',
    'Fee': 'mean'
}).reset_index()

volatility_impact.columns = ['Sentiment', 'Avg PnL', 'PnL Volatility', 'A

# Display table
print("\n=== Volatility Impact Summary ===")
print(volatility_impact.round(2))

# Plot Avg PnL vs PnL Volatility
plt.figure(figsize=(8,5))
sns.scatterplot(x='PnL Volatility', y='Avg PnL', hue='Sentiment', s=150,
plt.title('Volatility Impact on Profitability (by Sentiment)')
plt.xlabel('PnL Volatility')
plt.ylabel('Average Closed PnL')
plt.legend(title='Sentiment')
plt.tight_layout()
plt.show()
```

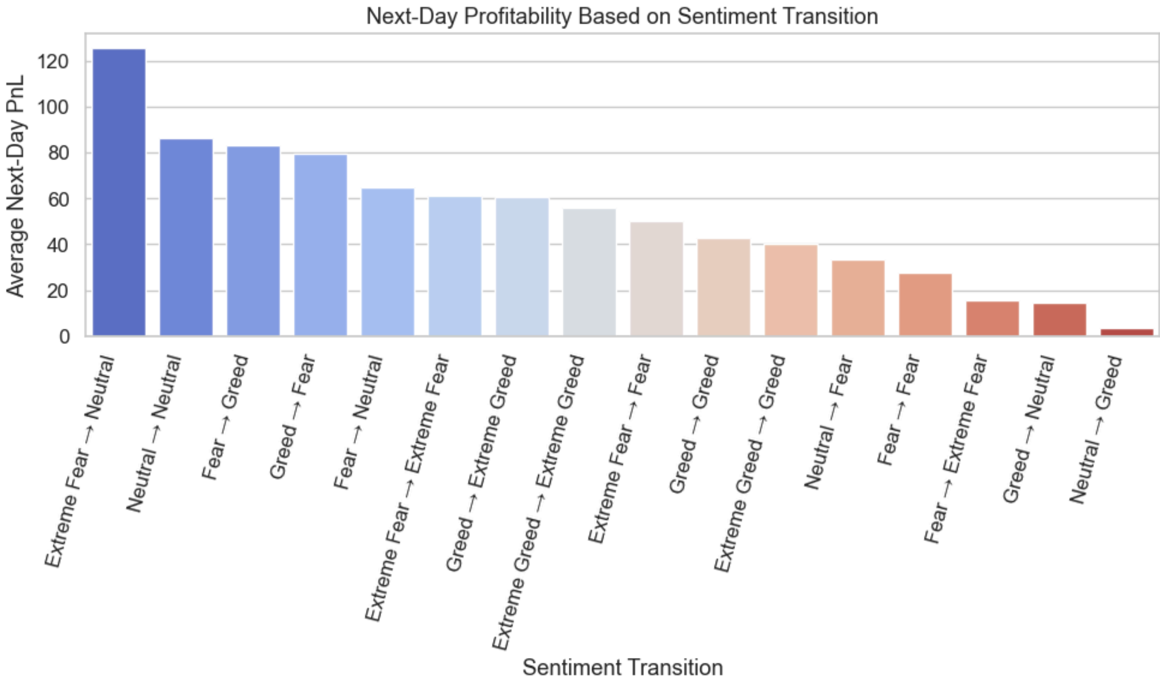
```
=== Volatility Impact Summary ===
        Sentiment  Avg PnL  PnL Volatility  Avg Trade Size USD  Avg Fee
0    Extreme Fear    34.54         1136.06             5349.73     1.12
1   Extreme Greed    67.89          766.83             3112.25     0.68
2            Fear    54.29          935.36             7816.11     1.50
3           Greed    42.74         1116.03             5736.88     1.25
4         Neutral    34.31          517.12             4782.73     1.04
```



In [ ]:
```
#  Sentiment Transition & Momentum Effect
import pandas as pd

# Extract daily average PnL by sentiment
daily_sentiment = merged_df.groupby(['date', 'classification'])['Closed P

# Sort by date for sequential analysis
daily_sentiment = daily_sentiment.sort_values('date')

# Assign next-day sentiment and next-day PnL for transition analysis
daily_sentiment['Next Sentiment'] = daily_sentiment['classification'].shi
daily_sentiment['Next Day PnL'] = daily_sentiment['Closed PnL'].shift(-1)

# Remove last day (no next-day data)
daily_sentiment.dropna(inplace=True)

# Create transition column
daily_sentiment['Transition'] = daily_sentiment['classification'] + " → "

# Group by sentiment transitions
transition_summary = daily_sentiment.groupby('Transition').agg({
    'Next Day PnL': ['mean', 'count']
}).reset_index()

transition_summary.columns = ['Transition', 'Avg Next-Day PnL', 'Occurren

# Display summary
print("\n=== Sentiment Transition & Momentum Summary ===")
print(transition_summary.sort_values('Avg Next-Day PnL', ascending=False)
```

```
=== Sentiment Transition & Momentum Summary ===
                    Transition  Avg Next-Day PnL  Occurrences
2        Extreme Fear → Neutral            125.57            1
15            Neutral → Neutral             86.09           34
7                  Fear → Greed             82.98            2
10                 Greed → Fear             79.57            2
8                Fear → Neutral             64.80           15
0     Extreme Fear → Extreme Fear           61.14            7
9          Greed → Extreme Greed            60.42           24
3   Extreme Greed → Extreme Greed           55.75           90
1           Extreme Fear → Fear             49.95            6
11                Greed → Greed             42.86          150
4         Extreme Greed → Greed             39.94           24
13              Neutral → Fear              33.14           16
6                  Fear → Fear              27.72           67
5           Fear → Extreme Fear             15.72            7
12              Greed → Neutral             14.79           17
14              Neutral → Greed              3.34           16
```

In [ ]:
```python
# Visualization: Sentiment Transition vs Next-Day PnL
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,6))
sns.barplot(x='Transition', y='Avg Next-Day PnL', data=transition_summary

plt.title('Next-Day Profitability Based on Sentiment Transition')
plt.xlabel('Sentiment Transition')
plt.ylabel('Average Next-Day PnL')
plt.xticks(rotation=75, ha='right')
plt.tight_layout()
plt.show()
```



In [ ]: