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

pd.set_option('display.max_columns', None)
sns.set_style('whitegrid')

TRADER_DATA_PATH = 'historical_data.csv'
SENTIMENT_DATA_PATH = 'fear_greed_index.csv'

try:
    df_trader = pd.read_csv(TRADER_DATA_PATH)
    df_sentiment = pd.read_csv(SENTIMENT_DATA_PATH)
    print("Data loaded successfully.")
except FileNotFoundError:
    print("ERROR: Please ensure the files are downloaded and paths are correct.")

print("\n--- Trader Data Info ---")
df_trader.info()
print("\n--- Sentiment Data Info ---")
df_sentiment.info()
```

```
Data loaded successfully.

--- Trader Data 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

--- Sentiment Data 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
```

```
#Data cleaning and preprocessing

df_trader['Trade_Time'] = pd.to_datetime(df_trader['Timestamp'], unit='ms')
df_trader['Trade_Date'] = df_trader['Trade_Time'].dt.date


financial_cols = ['Execution Price', 'Size USD', 'Closed PnL', 'Fee']
for col in financial_cols:
    df_trader[col] = pd.to_numeric(df_trader[col], errors='coerce')

df_trader.dropna(subset=['Closed PnL'], inplace=True)


df_sentiment['Sentiment_Date'] = pd.to_datetime(df_sentiment['date'])
df_sentiment['Sentiment_Date'] = df_sentiment['Sentiment_Date'].dt.date

df_sentiment['value'] = pd.to_numeric(df_sentiment['value'], errors='coerce')
```

```
df_sentiment['classification'] = df_sentiment['classification'].str.title()

df_sentiment_clean = df_sentiment[['Sentiment_Date', 'value', 'classification']].copy()
df_sentiment_clean.drop_duplicates(subset=['Sentiment_Date'], inplace=True)

print("\nData cleaning complete.")
```

```
Data cleaning complete.
```

```
#Data integration

df_merged = pd.merge(
    df_trader,
    df_sentiment_clean,
    left_on='Trade_Date',
    right_on='Sentiment_Date',
    how='left'
)

print(f"\nTotal Trades: {len(df_trader)}")
print(f"Trades matched to Sentiment: {len(df_merged.dropna(subset=['classification']))}")

df_analysis = df_merged.dropna(subset=['classification']).copy()

print("\nData merging complete. Ready for analysis.")
df_analysis.head()
```

```
Total Trades: 211224
Trades matched to Sentiment: 184263

Data merging complete. Ready for analysis.
```

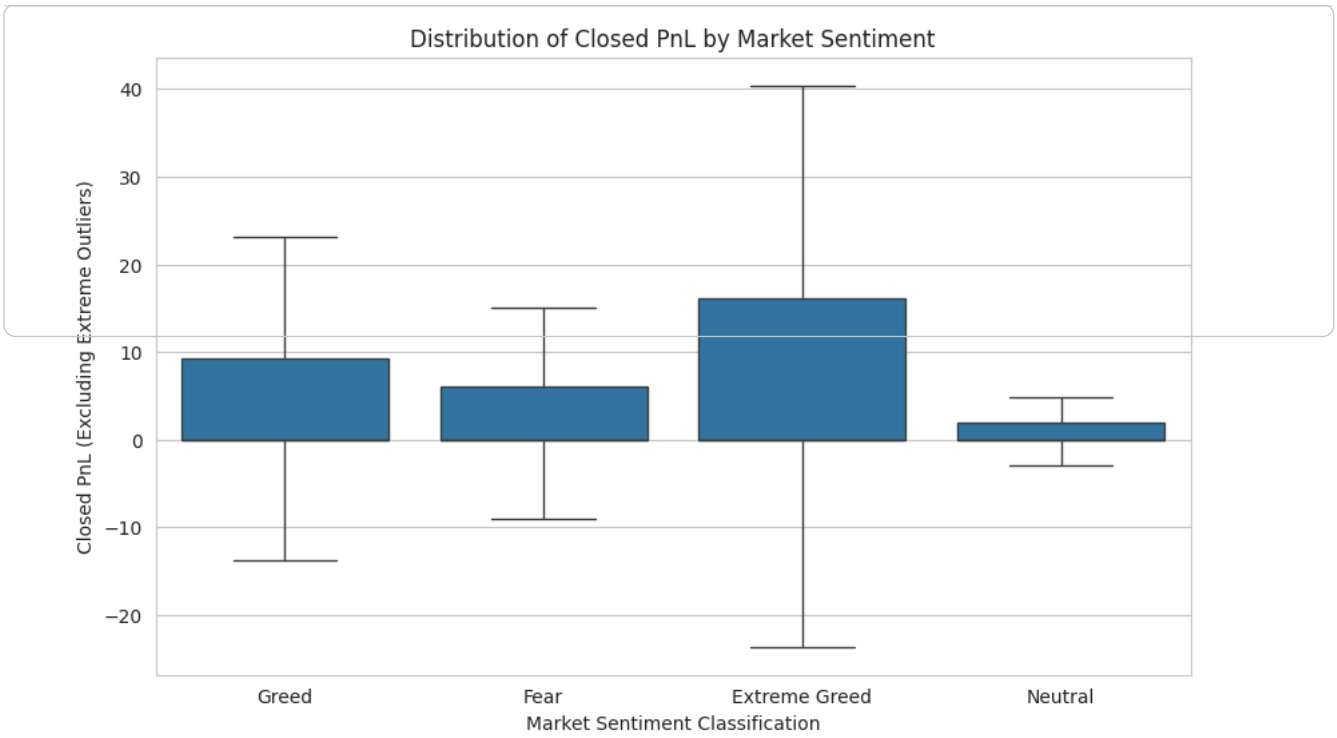
	Account	Coin	Execution Price	Size Tokens	Size USD	Side	Timestamp IST	Start Position	Direction	Clos
0	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9769	986.87	7872.16	BUY	02-12-2024 22:50	0.000000	Buy	
1	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9800	16.00	127.68	BUY	02-12-2024 22:50	986.524596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

```
#Analysis: performance by sentiment
sentiment_performance = df_analysis.groupby('classification').agg(
    Total_PnL=('Closed PnL', 'sum'),
    Mean_PnL=('Closed PnL', 'mean'),
    Median_PnL=('Closed PnL', 'median'),
    Total_Trades=('Closed PnL', 'count'),
    Mean_Trade_Size_USD=('Size USD', 'mean')
).reset_index()

def calculate_win_rate(group):
    winning_trades = (group['Closed PnL'] > 0).sum()
    total_trades = len(group)
    return winning_trades / total_trades if total_trades > 0 else 0

win_rate_series = df_analysis.groupby('classification').apply(calculate_win_rate).rename('Win_Rate')
sentiment_performance = sentiment_performance.merge(win_rate_series, on='classification').round(4)

print("\n--- Performance Metrics by Sentiment Classification ---")
print(sentiment_performance)


pnl_fear = df_analysis[df_analysis['classification'] == 'Fear']['Closed PnL']
pnl_greed = df_analysis[df_analysis['classification'] == 'Greed']['Closed PnL']

t_stat, p_value = stats.ttest_ind(pnl_fear, pnl_greed, equal_var=False, nan_policy='omit')

print("\n--- T-Test: Mean PnL (Fear vs. Greed) ---")
```

```
print(f"T-Statistic: {t_stat:.4f}")
print(f"P-Value: {p_value:.4f}")
if p_value < 0.05:
    print("Conclusion: The difference in mean PnL between Fear and Greed days is **statistically significant** (p < 0.05).")
else:
    print("Conclusion: The difference in mean PnL is **not statistically significant** (p > 0.05).")
```

```
--- Performance Metrics by Sentiment Classification ---
  classification     Total_PnL  Mean_PnL  Median_PnL  Total_Trades  \
0  Extreme Greed  1.769655e+05   25.4188         0.0          6962
1           Fear  6.699925e+06   50.0476         0.0        133871
2          Greed  3.189617e+06   87.8949         0.0         36289
3        Neutral  1.587424e+05   22.2297         0.0          7141

   Mean_Trade_Size_USD  Win_Rate
0            5660.2658    0.4901
1            5259.9778    0.4151
2            3182.8838    0.4465
3            3058.8481    0.3172

--- T-Test: Mean PnL (Fear vs. Greed) ---
T-Statistic: -5.8046
P-Value: 0.0000
Conclusion: The difference in mean PnL between Fear and Greed days is **statistically significant** (p < 0.05).
/tmp/ipython-input-2249439710.py:15: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behav
  win_rate_series = df_analysis.groupby('classification').apply(calculate_win_rate).rename('Win_Rate')
```

```
#Visualization
plt.figure(figsize=(10, 6))
sns.boxplot(x='classification', y='Closed PnL', data=df_analysis, showfliers=False) # showfliers=False removes extreme outli
plt.title('Distribution of Closed PnL by Market Sentiment')
plt.xlabel('Market Sentiment Classification')
plt.ylabel('Closed PnL (Excluding Extreme Outliers)')
plt.show()


trader_total_pnl = df_analysis.groupby('Account')['Closed PnL'].sum().sort_values(ascending=False)
TOP_N_PERCENT = 0.10
top_n = int(len(trader_total_pnl) * TOP_N_PERCENT)

successful_traders = trader_total_pnl.head(top_n).index

df_successful = df_analysis[df_analysis['Account'].isin(successful_traders)]

successful_sentiment_side = df_successful.groupby(['classification', 'Side']).agg(
    Count=('Trade ID', 'count')
).reset_index()

successful_behavior = successful_sentiment_side.pivot(index='classification', columns='Side', values='Count')
successful_behavior['Total'] = successful_behavior.sum(axis=1)
successful_behavior['Buy_Ratio'] = successful_behavior['BUY'] / successful_behavior['Total']
successful_behavior['Sell_Ratio'] = successful_behavior['SELL'] / successful_behavior['Total']

print("\n--- Successful Traders' (Top 10%) Behavior by Sentiment ---")
print("Ratio of BUY vs. SELL trades:")
print(successful_behavior[['Buy_Ratio', 'Sell_Ratio']].round(4))
```




```
--- Successful Traders' (Top 10%) Behavior by Sentiment ---
Ratio of BUY vs. SELL trades:
Side            Buy_Ratio  Sell_Ratio
classification
Extreme Greed      0.4919      0.5081
```