```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os
os.chdir('/content/drive/MyDrive/ds_Sanjana_Desh')
```

```
import pandas as pd

# Load the datasets
trader_data = pd.read_csv('csv_files/historical_data.csv')
sentiment_data = pd.read_csv('csv_files/fear_greed_index.csv')

# Display first few rows
print(trader_data.head())
print(sentiment_data.head())
```

```
                                      Account  Coin  Execution Price  \
0  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9769
1  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9800
2  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9855
3  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9874
4  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9894

   Size Tokens  Size USD Side     Timestamp IST  Start Position Direction  \
0       986.87   7872.16  BUY  02-12-2024 22:50        0.000000       Buy
1        16.00    127.68  BUY  02-12-2024 22:50      986.524596       Buy
2       144.09   1150.63  BUY  02-12-2024 22:50     1002.518996       Buy
3       142.98   1142.04  BUY  02-12-2024 22:50     1146.558564       Buy
4         8.73     69.75  BUY  02-12-2024 22:50     1289.488521       Buy

   Closed PnL                                   Transaction Hash     Order ID  \
```

```
0         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...  52017706630
1         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...  52017706630
2         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...  52017706630
3         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...  52017706630
4         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...  52017706630

   Crossed       Fee      Trade ID     Timestamp
0     True  0.345404  8.950000e+14  1.730000e+12
1     True  0.005600  4.430000e+14  1.730000e+12
2     True  0.050431  6.600000e+14  1.730000e+12
3     True  0.050043  1.080000e+15  1.730000e+12
4     True  0.003055  1.050000e+15  1.730000e+12
    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
```

```python
# Check data info
print("Trader Data Info:")
print(trader_data.info())
print("\nSentiment Data Info:")
print(sentiment_data.info())

# Check for missing values
print("\nMissing values in Trader Data:")
print(trader_data.isnull().sum())
print("\nMissing values in Sentiment Data:")
print(sentiment_data.isnull().sum())

# Basic statistics
print(trader_data.describe())
print(sentiment_data.describe())
```

```
 3   date            2644 non-null   object
dtypes: int64(2), object(2)
memory usage: 82.8+ KB
None

Missing values in Trader Data:
Account             0
```

```
mean       48.749001  6.965388e+10       1.163967  5.628549e+14  1.737744e+12
std       919.164828  1.835753e+10       6.758854  3.257565e+14  8.689920e+09
min   -117990.104100  1.732711e+08      -1.175712  0.000000e+00  1.680000e+12
25%         0.000000  5.983853e+10       0.016121  2.810000e+14  1.740000e+12
50%         0.000000  7.442939e+10       0.089578  5.620000e+14  1.740000e+12
75%         5.792797  8.335543e+10       0.393811  8.460000e+14  1.740000e+12
max    135329.090100  9.014923e+10     837.471593  1.130000e+15  1.750000e+12
          timestamp        value
count  2.644000e+03  2644.000000
mean   1.631899e+09    46.981089
std    6.597967e+07    21.827680
min    1.517463e+09     5.000000
25%    1.574811e+09    28.000000
50%    1.631900e+09    46.000000
75%    1.688989e+09    66.000000
max    1.746164e+09    95.000000
```

```
# Check the timestamp values first
print("Sample timestamps from trader data:")
print(trader_data['Timestamp'].head())
print("\nSample timestamps from sentiment data:")
print(sentiment_data['timestamp'].head())

# Convert timestamp columns to datetime
# These are in MILLISECONDS, so divide by 1000 or use unit='ms'
trader_data['Timestamp'] = pd.to_datetime(trader_data['Timestamp'], unit='ms')
sentiment_data['timestamp'] = pd.to_datetime(sentiment_data['timestamp'], unit='s')

# Extract date only (no time) for merging
trader_data['date'] = trader_data['Timestamp'].dt.date
sentiment_data['date'] = sentiment_data['timestamp'].dt.date

# Check the date ranges
print("Trader data date range:", trader_data['date'].min(), "to", trader_data['date'
print("Sentiment data date range:", sentiment_data['date'].min(), "to", sentiment_da
```

```
Sample timestamps from trader data:
0    1.730000e+12
1    1.730000e+12
2    1.730000e+12
3    1.730000e+12
4    1.730000e+12
Name: Timestamp, dtype: float64

Sample timestamps from sentiment data:
0    1517463000
1    1517549400
2    1517635800
3    1517722200
4    1517808600
Name: timestamp, dtype: int64
Trader data date range: 2023-03-28 to 2025-06-15
Sentiment data date range: 2018-02-01 to 2025-05-02
```

```
# Merge trader data with sentiment data on date
merged_data = trader_data.merge(sentiment_data[['date', 'classification', 'value']],
                                 on='date',
                                 how='left')

# Save merged data
merged_data.to_csv('csv_files/merged_data.csv', index=False)
print("Merged data shape:", merged_data.shape)
print(merged_data.head())
```

```
Merged data shape: (211224, 19)
                                      Account  Coin  Execution Price  \
0  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9769
1  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9800
2  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9855
3  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9874
4  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9894

   Size Tokens  Size USD Side     Timestamp IST  Start Position Direction  \
```

```
0      986.87   7872.16  BUY  02-12-2024 22:50        0.000000       Buy
1       16.00    127.68  BUY  02-12-2024 22:50      986.524596       Buy
2      144.09   1150.63  BUY  02-12-2024 22:50     1002.518996       Buy
3      142.98   1142.04  BUY  02-12-2024 22:50     1146.558564       Buy
4        8.73     69.75  BUY  02-12-2024 22:50     1289.488521       Buy

   Closed PnL                                   Transaction Hash     Order ID  \
0         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...  52017706630
1         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...  52017706630
2         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...  52017706630
3         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...  52017706630
4         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...  52017706630

   Crossed       Fee      Trade ID           Timestamp        date  \
0     True  0.345404  8.950000e+14 2024-10-27 03:33:20  2024-10-27
1     True  0.005600  4.430000e+14 2024-10-27 03:33:20  2024-10-27
2     True  0.050431  6.600000e+14 2024-10-27 03:33:20  2024-10-27
3     True  0.050043  1.080000e+15 2024-10-27 03:33:20  2024-10-27
4     True  0.003055  1.050000e+15 2024-10-27 03:33:20  2024-10-27

  classification  value
0          Greed   74.0
1          Greed   74.0
2          Greed   74.0
3          Greed   74.0
4          Greed   74.0
```

```python
# First, check what columns we actually have
print("Available columns in merged_data:")
print(merged_data.columns.tolist())
```

```
Available columns in merged_data:
['Account', 'Coin', 'Execution Price', 'Size Tokens', 'Size USD', 'Side', 'Timestamp IST', 'Start Position', 'Direct
```

```python
# Calculate profit/loss per trade
merged_data['PnL'] = merged_data['Closed PnL']
```

```
# Group by account to get trader performance
trader_performance = merged_data.groupby('Account').agg({
    'PnL': 'sum',
    'Fee': 'sum',
    'Trade ID': 'count'
}).rename(columns={'Trade ID': 'num_trades'})

trader_performance['net_profit'] = trader_performance['PnL'] - trader_performance['F
trader_performance['avg_profit_per_trade'] = trader_performance['net_profit'] / trad

print(trader_performance.head(10))
```

```
                                                     PnL          Fee  \
Account
0x083384f897ee0f19899168e3b1bec365f52a9012  1.600230e+06  7405.312304
0x23e7a7f8d14b550961925fbfdaa92f5d195ba5bd  4.788532e+04  2729.837889
0x271b280974205ca63b716753467d5a371de622ab -7.043619e+04  9280.982850
0x28736f43f1e871e6aa8b1148d38d4994275d72c4  1.324648e+05  2218.367366
0x2c229d22b100a7beb69122eed721cee9b24011dd  1.686580e+05  3108.196722
0x3998f134d6aaa2b6a5f723806d00fd2bbbbce891 -3.120360e+04   147.074763
0x39cef799f8b69da1995852eea189df24eb5cae3c  1.445692e+04  1458.657126
0x3f9a0aadc7f04a7c9d75dc1b5a6ddd6e36486cf6  5.349625e+04   176.274176
0x420ab45e0bd8863569a5efbb9c05d91f40624641  1.995056e+05   267.967089
0x430f09841d65beb3f27765503d0f850b8bce7713  4.165419e+05   747.006931

                                            num_trades    net_profit  \
Account
0x083384f897ee0f19899168e3b1bec365f52a9012        3818  1.592825e+06
0x23e7a7f8d14b550961925fbfdaa92f5d195ba5bd        7280  4.515548e+04
0x271b280974205ca63b716753467d5a371de622ab        3809 -7.971717e+04
0x28736f43f1e871e6aa8b1148d38d4994275d72c4       13311  1.302464e+05
0x2c229d22b100a7beb69122eed721cee9b24011dd        3239  1.655498e+05
0x3998f134d6aaa2b6a5f723806d00fd2bbbbce891         815 -3.135067e+04
0x39cef799f8b69da1995852eea189df24eb5cae3c        3589  1.299826e+04
0x3f9a0aadc7f04a7c9d75dc1b5a6ddd6e36486cf6         332  5.331997e+04
0x420ab45e0bd8863569a5efbb9c05d91f40624641         383  1.992376e+05
0x430f09841d65beb3f27765503d0f850b8bce7713        1237  4.157949e+05
```

```
                                            avg_profit_per_trade
Account
0x083384f897ee0f19899168e3b1bec365f52a9012            417.188190
0x23e7a7f8d14b550961925fbfdaa92f5d195ba5bd              6.202676
0x271b280974205ca63b716753467d5a371de622ab            -20.928636
0x28736f43f1e871e6aa8b1148d38d4994275d72c4              9.784873
0x2c229d22b100a7beb69122eed721cee9b24011dd             51.111395
0x3998f134d6aaa2b6a5f723806d00fd2bbbbce891            -38.467086
0x39cef799f8b69da1995852eea189df24eb5cae3c              3.621695
0x3f9a0aadc7f04a7c9d75dc1b5a6ddd6e36486cf6            160.602329
0x420ab45e0bd8863569a5efbb9c05d91f40624641            520.202678
0x430f09841d65beb3f27765503d0f850b8bce7713            336.131662
```

```python
# Analyze performance by sentiment
sentiment_analysis = merged_data.groupby('classification').agg({
    'PnL': ['mean', 'sum', 'count'],
    'Fee': 'sum'
})

print("\nPerformance by Market Sentiment:")
print(sentiment_analysis)

# Calculate win rate by sentiment
merged_data['is_profitable'] = merged_data['PnL'] > 0
win_rate_by_sentiment = merged_data.groupby('classification')['is_profitable'].mean(
print("\nWin Rate by Sentiment:")
print(win_rate_by_sentiment)
```

```
Performance by Market Sentiment:
                      PnL                                    Fee
                     mean           sum   count              sum
classification
Extreme Greed   25.418772  1.769655e+05    6962    6812.781233
Fear            50.047622  6.699925e+06  133871  145018.043618
Greed           87.894859  3.189617e+06   36289   24334.033389
```

```
Neutral          22.229713  1.587424e+05    7141    8743.877486

Win Rate by Sentiment:
classification
Extreme Greed    0.490089
Fear             0.415146
Greed            0.446471
Neutral          0.317182
Name: is_profitable, dtype: float64
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set style
sns.set_style("whitegrid")

# 1. PnL Distribution by Sentiment
plt.figure(figsize=(10, 6))
merged_data.boxplot(column='PnL', by='classification', figsize=(10, 6))
plt.title('Profit/Loss Distribution by Market Sentiment')
plt.suptitle('')
plt.xlabel('Market Sentiment')
plt.ylabel('Profit/Loss')
plt.savefig('outputs/pnl_by_sentiment.png', dpi=300, bbox_inches='tight')
plt.close()

# 2. Win Rate by Sentiment
plt.figure(figsize=(10, 6))
win_rate_by_sentiment.plot(kind='bar', color=['red', 'yellow', 'green'])
plt.title('Win Rate by Market Sentiment')
plt.xlabel('Market Sentiment')
plt.ylabel('Win Rate')
plt.xticks(rotation=0)
```

```
plt.savefig('outputs/win_rate_sentiment.png', dpi=300, bbox_inches='tight')
plt.close()

# 3. Trading Volume by Sentiment
plt.figure(figsize=(10, 6))
sentiment_counts = merged_data['classification'].value_counts()
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%')
plt.title('Trading Volume Distribution by Market Sentiment')
plt.savefig('outputs/trading_volume_sentiment.png', dpi=300, bbox_inches='tight')
plt.close()

# 4. Average PnL by Sentiment
plt.figure(figsize=(10, 6))
avg_pnl = merged_data.groupby('classification')['PnL'].mean()
avg_pnl.plot(kind='bar', color=['red', 'yellow', 'green'])
plt.title('Average Profit/Loss by Market Sentiment')
plt.xlabel('Market Sentiment')
plt.ylabel('Average PnL')
plt.xticks(rotation=0)
plt.savefig('outputs/avg_pnl_sentiment.png', dpi=300, bbox_inches='tight')
plt.close()

print("All charts saved!")
```

```
All charts saved!
<Figure size 1000x600 with 0 Axes>
```

```
# Time series analysis
merged_data_sorted = merged_data.sort_values('Timestamp')

# Daily aggregate performance
```

```
daily_performance = merged_data_sorted.groupby('date').agg({
    'PnL': 'sum',
    'value': 'mean',  # Fear/Greed index value
    'Trade ID': 'count'
})

# Plot PnL over time with sentiment
fig, ax1 = plt.subplots(figsize=(14, 6))

ax1.plot(daily_performance.index, daily_performance['PnL'].cumsum(), 'b-', label='Cu
ax1.set_xlabel('Date')
ax1.set_ylabel('Cumulative PnL', color='b')
ax1.tick_params(axis='y', labelcolor='b')

ax2 = ax1.twinx()
ax2.plot(daily_performance.index, daily_performance['value'], 'r-', alpha=0.5, label
ax2.set_ylabel('Fear/Greed Index', color='r')
ax2.tick_params(axis='y', labelcolor='r')

plt.title('Trading Performance vs Market Sentiment Over Time')
fig.tight_layout()
plt.savefig('outputs/performance_vs_sentiment_time.png', dpi=300, bbox_inches='tight
plt.close()
```

```
# Correlation between sentiment value and PnL
correlation = merged_data[['value', 'PnL']].corr()
print("\nCorrelation Matrix:")
print(correlation)

# Statistical test
```

```
from scipy import stats

fear_trades = merged_data[merged_data['classification'] == 'Fear']['PnL']
greed_trades = merged_data[merged_data['classification'] == 'Greed']['PnL']

if len(fear_trades) > 0 and len(greed_trades) > 0:
    t_stat, p_value = stats.ttest_ind(fear_trades, greed_trades)
    print(f"\nT-test between Fear and Greed trades:")
    print(f"T-statistic: {t_stat:.4f}")
    print(f"P-value: {p_value:.4f}")
```

```
Correlation Matrix:
          value       PnL
value  1.000000  0.011132
PnL    0.011132  1.000000

T-test between Fear and Greed trades:
T-statistic: -6.6260
P-value: 0.0000
```

```
# Summary statistics
print("\n=== KEY FINDINGS ===")
print(f"\nTotal Trades Analyzed: {len(merged_data)}")
print(f"Total Unique Traders: {merged_data['Account'].nunique()}")
print(f"\nOverall Performance:")
print(f"Total PnL: ${merged_data['PnL'].sum():,.2f}")
print(f"Total Fees: ${merged_data['Fee'].sum():,.2f}")
print(f"Net Profit: ${(merged_data['PnL'].sum() - merged_data['Fee'].sum()):,.2f}")
print(f"Overall Win Rate: {(merged_data['PnL'] > 0).mean():.2%}")

print("\n\nPerformance by Sentiment:")
for sentiment in ['Fear', 'Neutral', 'Greed']:
```

```python
    subset = merged_data[merged_data['classification'] == sentiment]
    if len(subset) > 0:
        print(f"\n{sentiment}:")
        print(f"  Number of trades: {len(subset)}")
        print(f"  Average PnL: ${subset['PnL'].mean():,.2f}")
        print(f"  Win Rate: {(subset['PnL'] > 0).mean():.2%}")
        print(f"  Total PnL: ${subset['PnL'].sum():,.2f}")
```

```
=== KEY FINDINGS ===

Total Trades Analyzed: 211224
Total Unique Traders: 32

Overall Performance:
Total PnL: $10,296,958.94
Total Fees: $245,857.72
Net Profit: $10,051,101.22
Overall Win Rate: 41.13%


Performance by Sentiment:

Fear:
  Number of trades: 133871
  Average PnL: $50.05
  Win Rate: 41.51%
  Total PnL: $6,699,925.19

Neutral:
  Number of trades: 7141
  Average PnL: $22.23
  Win Rate: 31.72%
  Total PnL: $158,742.38

Greed:
  Number of trades: 36289
  Average PnL: $87.89
  Win Rate: 44.65%
```

Total PnL: $3,189,616.54

```
# Save key findings to text file
with open('outputs/key_findings.txt', 'w') as f:
    f.write("=== TRADER BEHAVIOR & MARKET SENTIMENT ANALYS
    f.write(f"Analysis Period: {merged_data['date'].min()
    f.write(f"Total Trades: {len(merged_data)}\n")
    f.write(f"Unique Traders: {merged_data['Account'].nun
    f.write("KEY INSIGHTS:\n")
    f.write("1. Traders perform [better/worse] during Fear
    f.write("2. Greed sentiment correlates with [higher/lo
    f.write("3. [Add your specific findings based on the a

print("Findings saved to outputs/key_findings.txt")
```

```
Findings saved to outputs/key_findings.txt
```

Start coding or generate with AI.