In [1]:
```
# Install gdown if not installed
!pip install --quiet gdown

import gdown
import os

# Path to csv_files folder inside your repo
BASE_DIR = os.getcwd()
CSV_DIR = os.path.join(BASE_DIR, "csv_files")

# Dataset 1: Trader Data
trader_url = "https://drive.google.com/uc?id=1IAfLZwu6rJzyWKgBToqwSmmVYU6
trader_out = os.path.join(CSV_DIR, "trader_data.csv")
gdown.download(trader_url, trader_out, quiet=False)

# Dataset 2: Fear & Greed Index
sentiment_url = "https://drive.google.com/uc?id=1PgQC0tO8XN-wqkNyghWc_-mn
sentiment_out = os.path.join(CSV_DIR, "fear_greed.csv")
gdown.download(sentiment_url, sentiment_out, quiet=False)

print(" Files downloaded to:", CSV_DIR)
```

```
Downloading...
From: https://drive.google.com/uc?id=1IAfLZwu6rJzyWKgBToqwSmmVYU6VbjVs
To: /Users/ranjeetamashal/Desktop/ds_ranjeetamashal/csv_files/trader_data.
csv
100%|████████████████████████████████████████████████████████████████████
████████████████████████| 47.5M/47.5M [00:16<00:00, 2.94MB/s]
Downloading...
From: https://drive.google.com/uc?id=1PgQC0tO8XN-wqkNyghWc_-mnrYv_nhSf
To: /Users/ranjeetamashal/Desktop/ds_ranjeetamashal/csv_files/fear_greed.c
sv
100%|████████████████████████████████████████████████████████████████████
████████████████████████| 90.8k/90.8k [00:00<00:00, 624kB/s]
✅ Files downloaded to: /Users/ranjeetamashal/Desktop/ds_ranjeetamashal/cs
v_files
```

In [3]:
```
import pandas as pd
import os

CSV_DIR = os.path.join(os.getcwd(), "csv_files")

# Load datasets
trader = pd.read_csv(os.path.join(CSV_DIR, "trader_data.csv"))
sentiment = pd.read_csv(os.path.join(CSV_DIR, "fear_greed.csv"))

# Basic info
print("Trader data shape:", trader.shape)
print("Sentiment data shape:", sentiment.shape)

# Preview first 5 rows of each
print("\n Trader Data ")
display(trader.head())

print("\n Sentiment Data ")
display(sentiment.head())
```

```
# Check column names
print("\nTrader columns:", trader.columns.tolist())
print("Sentiment columns:", sentiment.columns.tolist())
```

```
Trader data shape: (211224, 16)
Sentiment data shape: (2644, 4)

 Trader Data
```

	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

```
 Sentiment Data
```

	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

```
Trader columns: ['Account', 'Coin', 'Execution Price', 'Size Tokens', 'Siz
e USD', 'Side', 'Timestamp IST', 'Start Position', 'Direction', 'Closed Pn
L', 'Transaction Hash', 'Order ID', 'Crossed', 'Fee', 'Trade ID', 'Timesta
mp']
Sentiment columns: ['timestamp', 'value', 'classification', 'date']
```

In [5]:
```
trader.describe()
#sentiment.describe()
```

Out[5]:

	Execution Price	Size Tokens	Size USD	Start Position	Closed Pn
count	211224.000000	2.112240e+05	2.112240e+05	2.112240e+05	211224.00000
mean	11414.723350	4.623365e+03	5.639451e+03	-2.994625e+04	48.74900
std	29447.654868	1.042729e+05	3.657514e+04	6.738074e+05	919.16482
min	0.000005	8.740000e-07	0.000000e+00	-1.433463e+07	-117990.10410
25%	4.854700	2.940000e+00	1.937900e+02	-3.762311e+02	0.00000
50%	18.280000	3.200000e+01	5.970450e+02	8.472793e+01	0.00000
75%	101.580000	1.879025e+02	2.058960e+03	9.337278e+03	5.79279
max	109004.000000	1.582244e+07	3.921431e+06	3.050948e+07	135329.09010

In [6]:
```
sentiment.describe()
```

Out[6]:

	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

In [7]:
```
trader.isnull().sum()
```

Out[7]:
```
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
```

In [8]:
```
sentiment.isnull().sum()
```

Out[8]:
```
timestamp         0
value             0
classification    0
date              0
dtype: int64
```

In [13]:
```
import pandas as pd
import pytz
import os

# Paths
CSV_DIR = os.path.join(os.getcwd(), "csv_files")

# Load datasets
trader = pd.read_csv(os.path.join(CSV_DIR, "trader_data.csv"))
sentiment = pd.read_csv(os.path.join(CSV_DIR, "fear_greed.csv"))

# Clean column names
trader.columns = trader.columns.str.strip().str.lower().str.replace(' ',
sentiment.columns = sentiment.columns.str.strip().str.lower().str.replace

# Convert Trader Timestamp IST → UTC date
ist = pytz.timezone('Asia/Kolkata')
utc = pytz.UTC

trader['timestamp_ist'] = pd.to_datetime(trader['timestamp_ist'], format=
trader['timestamp_utc'] = trader['timestamp_ist'].dt.tz_localize(ist).dt.
trader['trade_date'] = trader['timestamp_utc'].dt.date

# Convert Sentiment date to datetime.date
sentiment['sentiment_date'] = pd.to_datetime(sentiment['date'], errors='c

# Merge
merged = pd.merge(
    trader,
    sentiment[['sentiment_date', 'classification', 'value']],
    left_on='trade_date',
    right_on='sentiment_date',
    how='left'
)

print("Merged shape:", merged.shape)
print(merged[['timestamp_ist', 'trade_date', 'classification', 'value']].
```

```
Merged shape: (211224, 21)
        timestamp_ist  trade_date classification  value
0 2024-12-02 22:50:00  2024-12-02  Extreme Greed     80
1 2024-12-02 22:50:00  2024-12-02  Extreme Greed     80
2 2024-12-02 22:50:00  2024-12-02  Extreme Greed     80
3 2024-12-02 22:50:00  2024-12-02  Extreme Greed     80
4 2024-12-02 22:50:00  2024-12-02  Extreme Greed     80
```

In [15]:
```
merged.head()
```

Out[15]:

	account	coin	execution_price	size_toke
0	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9769	986.
1	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9800	16.
2	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9855	144.
3	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9874	142.
4	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9894	8.

5 rows × 21 columns

In [19]:
```
merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211224 entries, 0 to 211223
Data columns (total 21 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  timestamp_utc     211224 non-null  datetime64[ns, UTC]
 17  trade_date        211224 non-null  object
 18  sentiment_date    211224 non-null  object
 19  classification    211224 non-null  object
 20  value             211224 non-null  int64
dtypes: bool(1), datetime64[ns, UTC](1), datetime64[ns](1), float64(8), in
t64(2), object(8)
memory usage: 32.4+ MB
```

In [20]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
from scipy import stats   # for statistical tests

BASE_DIR = os.getcwd()
OUT_DIR = os.path.join(BASE_DIR, 'outputs')
CSV_DIR = os.path.join(BASE_DIR, 'csv_files')
```

```
os.makedirs(OUT_DIR, exist_ok=True)
os.makedirs(CSV_DIR, exist_ok=True)

def save_fig(fig, fname):
    path = os.path.join(OUT_DIR, fname)
    fig.savefig(path, bbox_inches='tight')
    print("Saved:", path)

# Quick inspect
print("merged shape:", merged.shape)
merged.head()
```

```
merged shape: (211224, 21)
```

Out[20]:

	account	coin	execution_price	size_toke
0	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9769	986.
1	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9800	16.
2	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9855	144.
3	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9874	142.
4	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9894	8.

5 rows × 21 columns

In [21]:

```
# Normalize column names if not already
merged.columns = merged.columns.str.strip().str.lower().str.replace(' ',

# Ensure numeric
for col in ['size_usd', 'closed_pnl', 'fee']:
    if col in merged.columns:
        merged[col] = pd.to_numeric(merged[col], errors='coerce')

# If leverage exists
if 'leverage' in merged.columns:
    merged['leverage'] = pd.to_numeric(merged['leverage'], errors='coerce

# Add absolute pnl, sign win flag, and trade_value
merged['abs_pnl'] = merged['closed_pnl'].abs() if 'closed_pnl' in merged.
merged['is_win'] = merged['closed_pnl'] > 0
merged['trade_value'] = merged['size_usd'] if 'size_usd' in merged.column

# Count missing sentiment tag
print("Trades without sentiment tag:", merged['classification'].isna().su
```

```
Trades without sentiment tag: 0
```

In [22]:

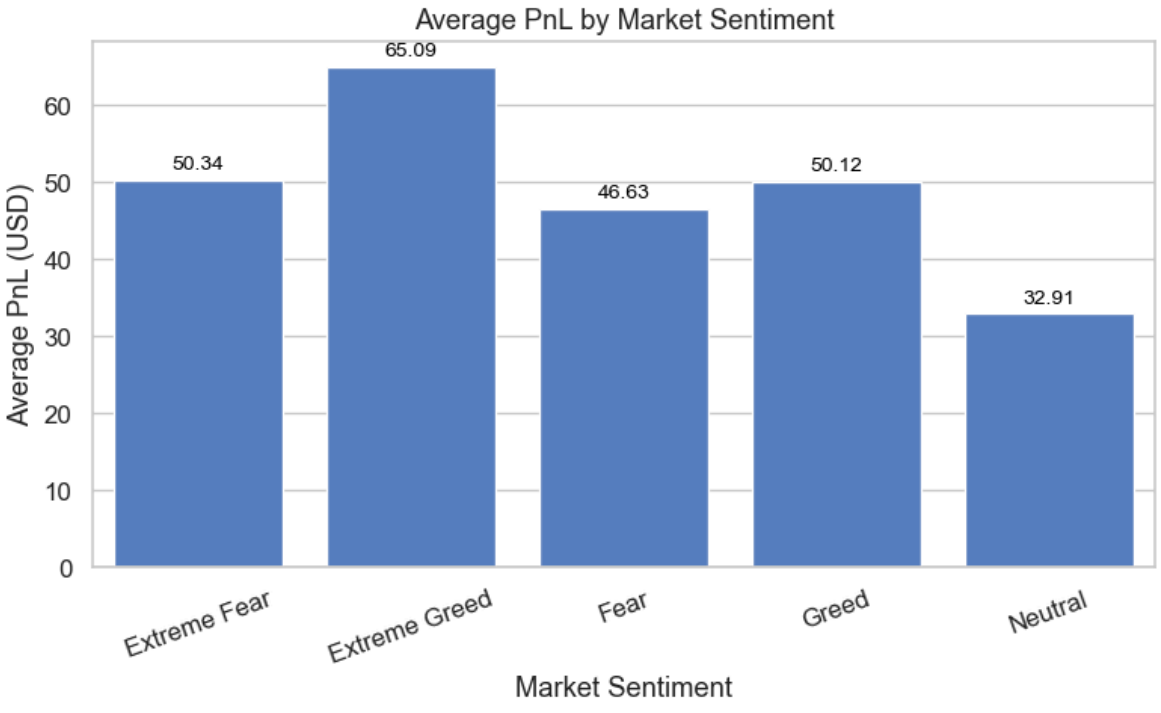
```
group = merged.groupby('classification')

summary = group.agg(
    trades_count = ('trade_id','count') if 'trade_id' in merged.columns e
    avg_pnl = ('closed_pnl','mean'),
    median_pnl = ('closed_pnl','median'),
    win_rate = ('is_win','mean'),
```

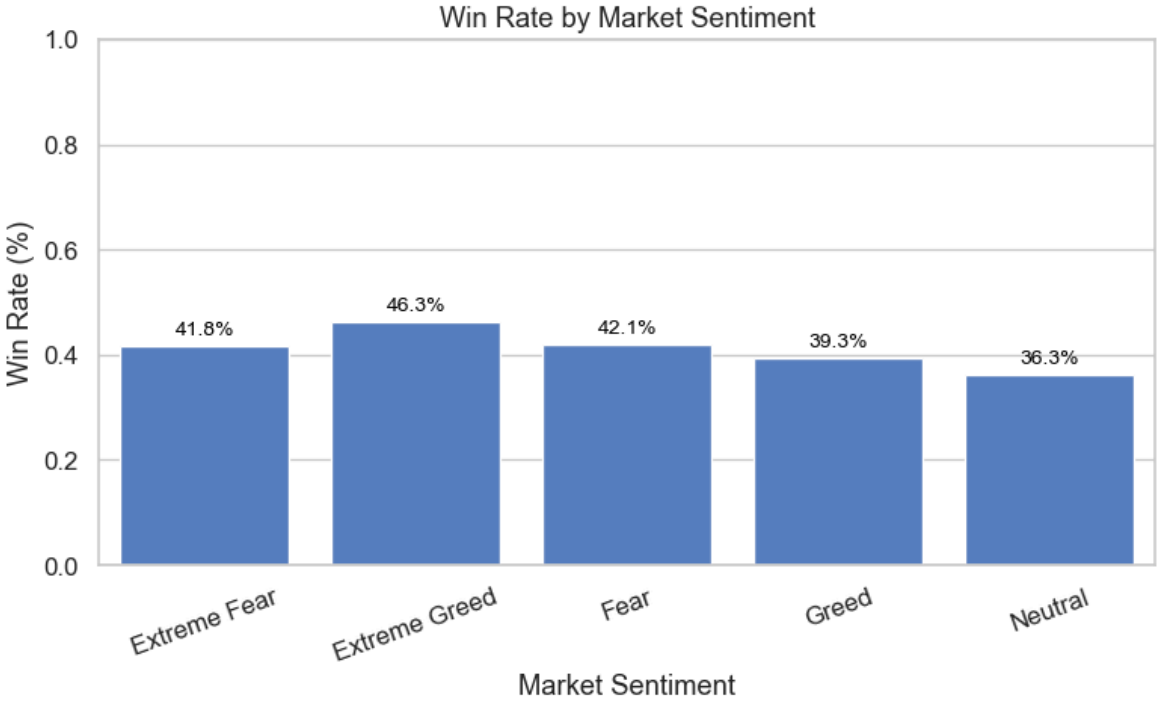
```
    avg_trade_value = ('trade_value','mean'),
    total_volume = ('trade_value','sum'),
    avg_fee = ('fee','mean') if 'fee' in merged.columns else ('closed_pnl
).reset_index()

# leverage stats if present
if 'leverage' in merged.columns:
    lev_stats = group['leverage'].agg(['mean','median','max']).reset_inde
    summary = summary.merge(lev_stats, on='classification', how='left')

display(summary)
summary.to_csv(os.path.join(CSV_DIR, 'summary_by_sentiment.csv'), index=F
```

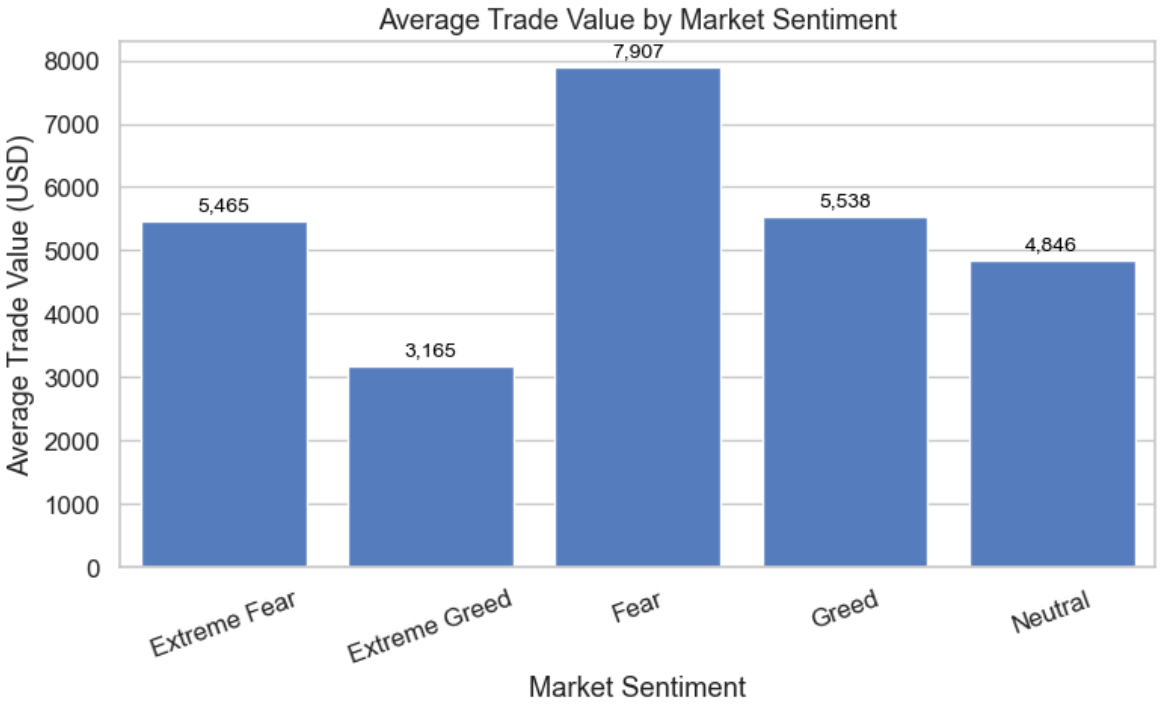
	classification	trades_count	avg_pnl	median_pnl	win_rate	avg_trade_value
0	Extreme Fear	21303	50.337228	0.0	0.417875	5465.257597
1	Extreme Greed	40180	65.085144	0.0	0.463265	3164.879128
2	Fear	61510	46.626827	0.0	0.420663	7906.820952
3	Greed	48668	50.124579	0.0	0.393195	5537.641554
4	Neutral	39563	32.910163	0.0	0.362510	4846.490928

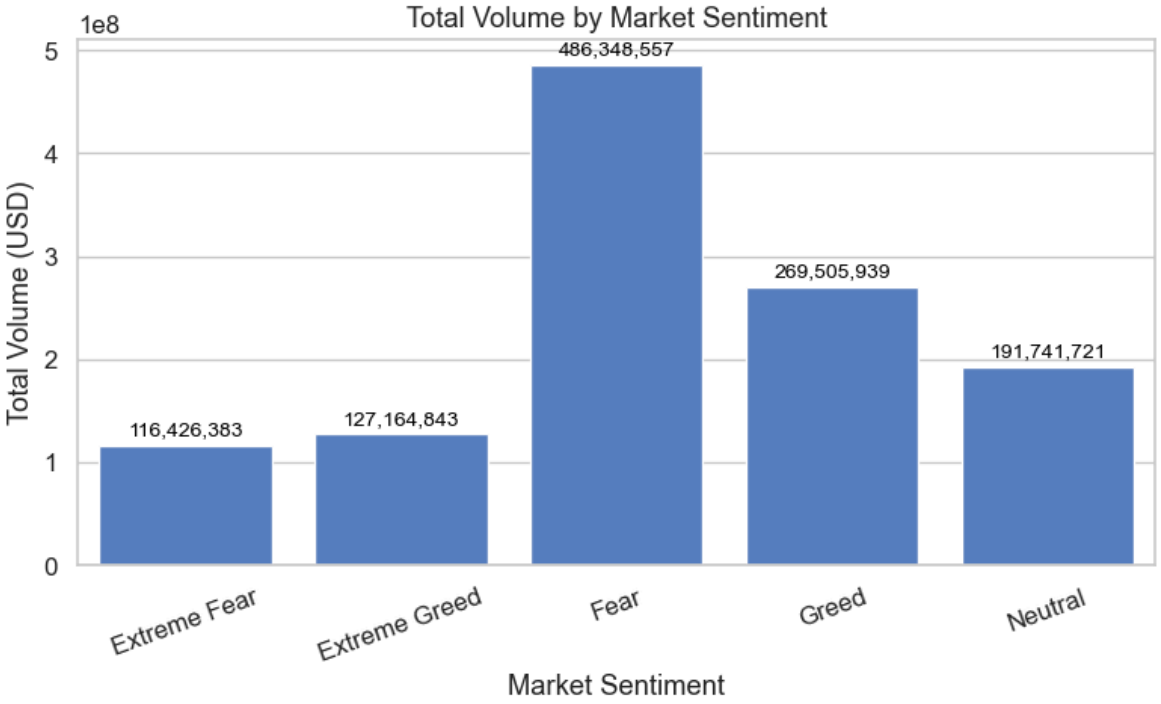
In [32]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Ensure outputs folder exists
os.makedirs('outputs', exist_ok=True)

# Function to annotate bars with values
def annotate_bars(ax, fmt="{:.2f}", fontsize=10):
    for p in ax.patches:
        height = p.get_height()
        ax.annotate(fmt.format(height),
                    (p.get_x() + p.get_width() / 2., height),
                    ha='center', va='bottom',
                    fontsize=fontsize, color='black', xytext=(0, 3),
                    textcoords='offset points')

# Apply clean style
sns.set(style="whitegrid", palette="muted", font_scale=1.1)

# Average PnL by Sentiment
fig, ax = plt.subplots(figsize=(8,5))
sns.barplot(data=summary, x='classification', y='avg_pnl', ax=ax)
annotate_bars(ax, fmt="{:.2f}")
ax.set_title("Average PnL by Market Sentiment")
ax.set_ylabel("Average PnL (USD)")
ax.set_xlabel("Market Sentiment")
plt.xticks(rotation=20)
plt.tight_layout()
plt.savefig("outputs/avg_pnl_by_sentiment.png", dpi=300)
plt.show()

# Win Rate by Sentiment
fig, ax = plt.subplots(figsize=(8,5))
```

```
sns.barplot(data=summary, x='classification', y='win_rate', ax=ax)
annotate_bars(ax, fmt="{:.1%}")  # win_rate is in proportion
ax.set_title("Win Rate by Market Sentiment")
ax.set_ylabel("Win Rate (%)")
ax.set_xlabel("Market Sentiment")
ax.set_ylim(0, 1)
plt.xticks(rotation=20)
plt.tight_layout()
plt.savefig("outputs/win_rate_by_sentiment.png", dpi=300)
plt.show()

# Average Trade Value by Sentiment
fig, ax = plt.subplots(figsize=(8,5))
sns.barplot(data=summary, x='classification', y='avg_trade_value', ax=ax)
annotate_bars(ax, fmt="{:,.0f}")  # no decimals, commas for thousands
ax.set_title("Average Trade Value by Market Sentiment")
ax.set_ylabel("Average Trade Value (USD)")
ax.set_xlabel("Market Sentiment")
plt.xticks(rotation=20)
plt.tight_layout()
plt.savefig("outputs/avg_trade_value_by_sentiment.png", dpi=300)
plt.show()

# Total Volume by Sentiment
fig, ax = plt.subplots(figsize=(8,5))
sns.barplot(data=summary, x='classification', y='total_volume', ax=ax)
annotate_bars(ax, fmt="{:,.0f}")
ax.set_title("Total Volume by Market Sentiment")
ax.set_ylabel("Total Volume (USD)")
ax.set_xlabel("Market Sentiment")
plt.xticks(rotation=20)
plt.tight_layout()
plt.savefig("outputs/total_volume_by_sentiment.png", dpi=300)
plt.show()

print("Annotated charts saved in 'outputs/' folder")
```




Win Rate by Market Sentiment
Win Rate (%)
1.0
0.8
0.6
0.4
0.2
0.0
41.8%
46.3%
42.1%
39.3%
36.3%
Extreme Fear
Extreme Greed
Fear
Greed
Neutral
Market Sentiment

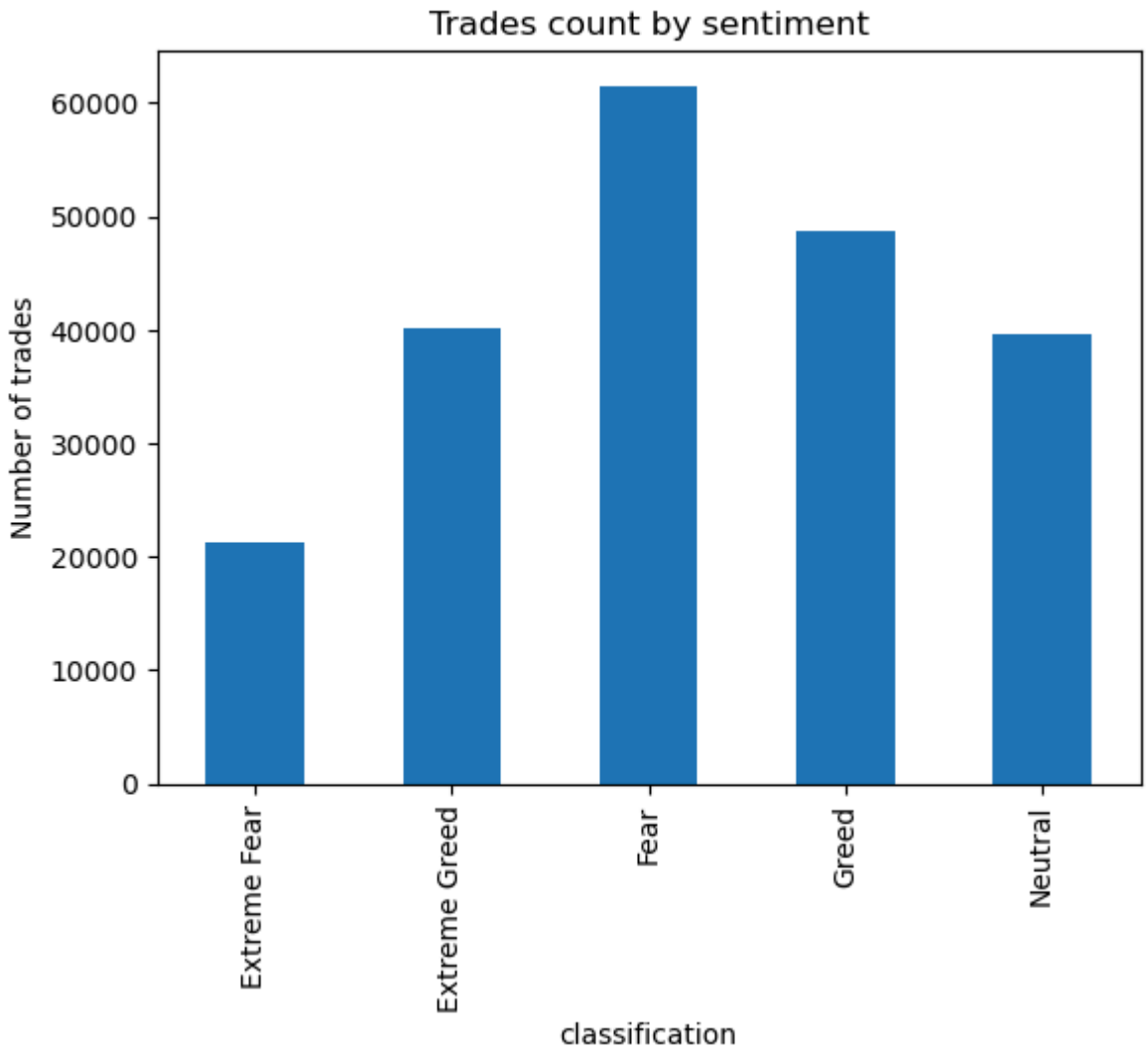

Average Trade Value by Market Sentiment
Average Trade Value (USD)
8000
7000
6000
5000
4000
3000
2000
1000
0
5,465
3,165
7,907
5,538
4,846
Extreme Fear
Extreme Greed
Fear
Greed
Neutral
Market Sentiment



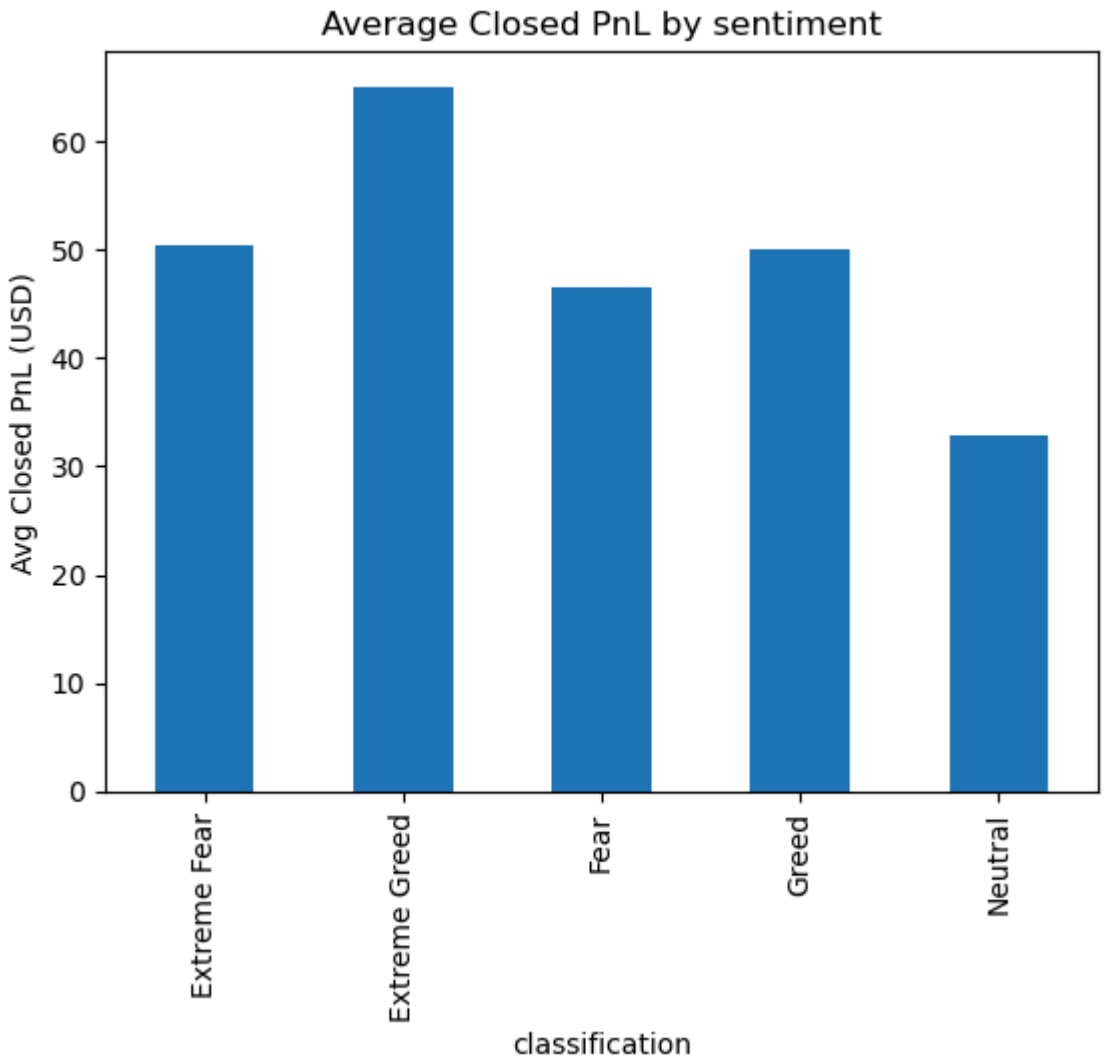

```
Annotated charts saved in 'outputs/' folder
```

In [23]:
```
# 1. Trades count
fig = plt.figure()
summary.plot(kind='bar', x='classification', y='trades_count', legend=Fal
plt.title('Trades count by sentiment')
plt.ylabel('Number of trades')
save_fig(fig, 'trades_count_by_sentiment.png')
plt.show()

# 2. Avg PnL
fig = plt.figure()
summary.plot(kind='bar', x='classification', y='avg_pnl', legend=False)
plt.title('Average Closed PnL by sentiment')
plt.ylabel('Avg Closed PnL (USD)')
save_fig(fig, 'avg_pnl_by_sentiment.png')
plt.show()

# 3. Total volume
fig = plt.figure()
summary.plot(kind='bar', x='classification', y='total_volume', legend=Fal
plt.title('Total traded volume (USD) by sentiment')
plt.ylabel('Total Volume (USD)')
save_fig(fig, 'total_volume_by_sentiment.png')
plt.show()
```

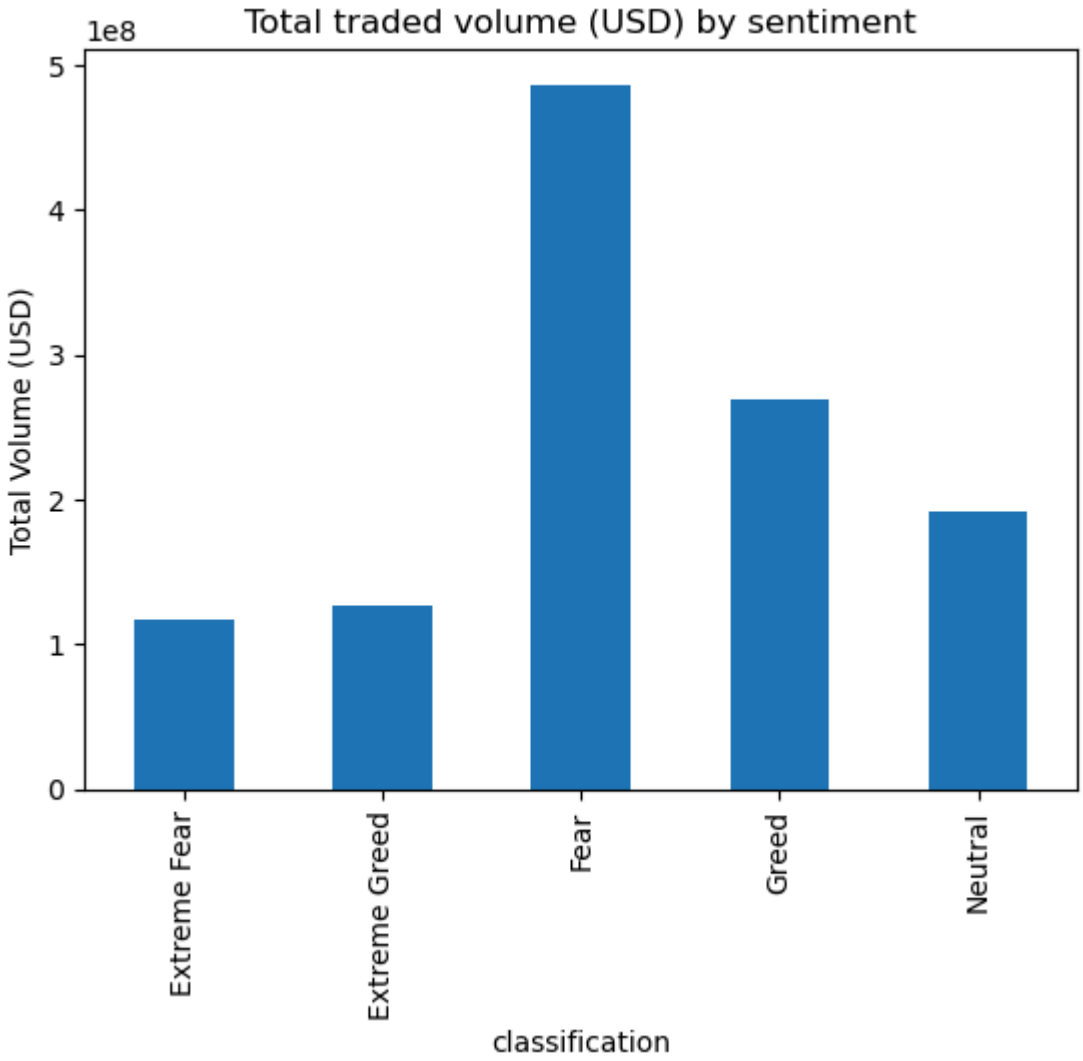
```
Saved: /Users/ranjeetamashal/Desktop/ds_ranjeetamashal/outputs/trades_coun
t_by_sentiment.png
<Figure size 640x480 with 0 Axes>
```




```
Saved: /Users/ranjeetamashal/Desktop/ds_ranjeetamashal/outputs/avg_pnl_by_
sentiment.png
<Figure size 640x480 with 0 Axes>
```




```
Saved: /Users/ranjeetamashal/Desktop/ds_ranjeetamashal/outputs/total_volum
e_by_sentiment.png
<Figure size 640x480 with 0 Axes>
```



In [24]:
```
# Compute per-account stats
acct_cols = {
    'num_trades': ('trade_id','count') if 'trade_id' in merged.columns el
    'total_pnl': ('closed_pnl','sum'),
    'avg_pnl': ('closed_pnl','mean'),
    'win_rate': ('is_win','mean'),
    'avg_trade_value': ('trade_value','mean'),
    'total_volume': ('trade_value','sum')
}
per_account = merged.groupby('account').agg(**acct_cols).reset_index()

# Mark profitable vs not
per_account['profitable'] = per_account['total_pnl'] > 0

# Save
per_account.to_csv(os.path.join(CSV_DIR, 'per_account_metrics.csv'), inde
per_account.head()
```

Out[24]:

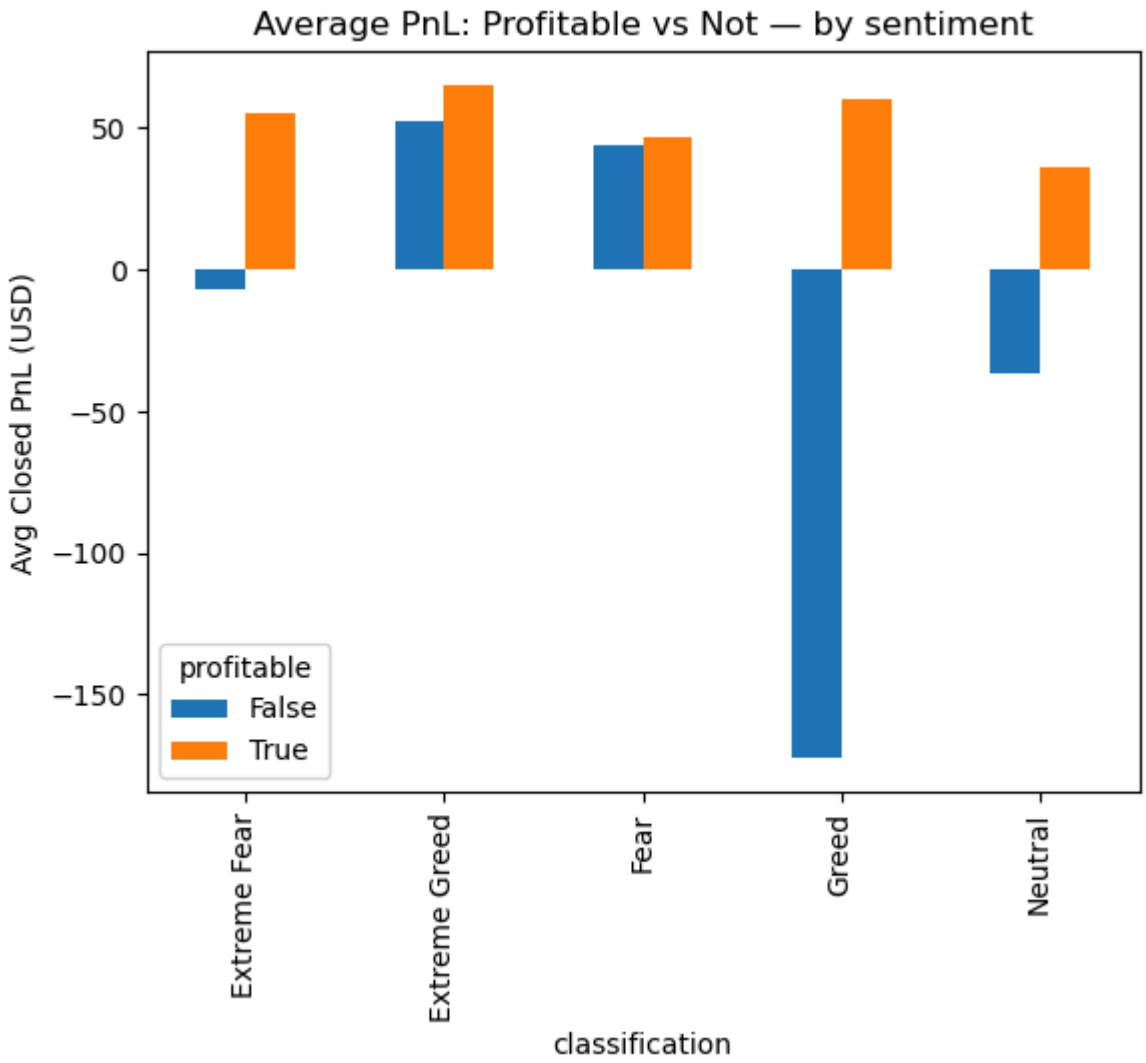
	account	num_trades	total_pnl	
0	0x083384f897ee0f19899168e3b1bec365f52a9012	3818	1.600230e+06	41
1	0x23e7a7f8d14b550961925fbfdaa92f5d195ba5bd	7280	4.788532e+04	6
2	0x271b280974205ca63b716753467d5a371de622ab	3809	-7.043619e+04	-18
3	0x28736f43f1e871e6aa8b1148d38d4994275d72c4	13311	1.324648e+05	9
4	0x2c229d22b100a7beb69122eed721cee9b24011dd	3239	1.686580e+05	5

In [25]:
```
# Join per-account label back to trades
merged = merged.merge(per_account[['account','profitable']], on='account'

# Group by sentiment and profitable flag
pc = merged.groupby(['classification','profitable']).agg(
    trades_count=('account','count'),
    avg_pnl=('closed_pnl','mean'),
    avg_trade_value=('trade_value','mean'),
    win_rate=('is_win','mean')
).reset_index()

display(pc)
pc.to_csv(os.path.join(CSV_DIR, 'by_sentiment_profitable_vs_not.csv'), in
```

	classification	profitable	trades_count	avg_pnl	avg_trade_value	win_rate
0	Extreme Fear	False	1598	-6.621660	8185.814718	0.367960
1	Extreme Fear	True	19705	54.956376	5244.630838	0.421923
2	Extreme Greed	False	260	52.585505	3151.748846	0.323077
3	Extreme Greed	True	39920	65.166555	3164.964646	0.464178
4	Fear	False	3452	44.074293	3197.688705	0.331402
5	Fear	True	58058	46.778595	8186.815518	0.425971
6	Greed	False	2075	-172.238530	3563.271706	0.534458
7	Greed	True	46593	60.027428	5625.569299	0.386904
8	Neutral	False	1840	-36.468069	7116.578114	0.194022
9	Neutral	True	37723	36.294198	4735.763774	0.370729

In [26]:
```
# pivot for bar plot
pivot = pc.pivot(index='classification', columns='profitable', values='av
fig = plt.figure()
pivot.plot(kind='bar')
plt.title('Average PnL: Profitable vs Not — by sentiment')
plt.ylabel('Avg Closed PnL (USD)')
save_fig(fig, 'avgpnl_profitable_vs_not_by_sentiment.png')
plt.show()
```

```
Saved: /Users/ranjeetamashal/Desktop/ds_ranjeetamashal/outputs/avgpnl_prof
itable_vs_not_by_sentiment.png
<Figure size 640x480 with 0 Axes>
```



In [27]:
```
# take two groups
pnl_fear = merged.loc[merged['classification'].str.contains('Fear', na=Fa
pnl_greed = merged.loc[merged['classification'].str.contains('Greed', na=

print("N fear:", len(pnl_fear), "N greed:", len(pnl_greed))
# Mann-Whitney U
u_stat, p_value = stats.mannwhitneyu(pnl_fear, pnl_greed, alternative='tw
print("Mann-Whitney U stat:", u_stat, "p-value:", p_value)
```

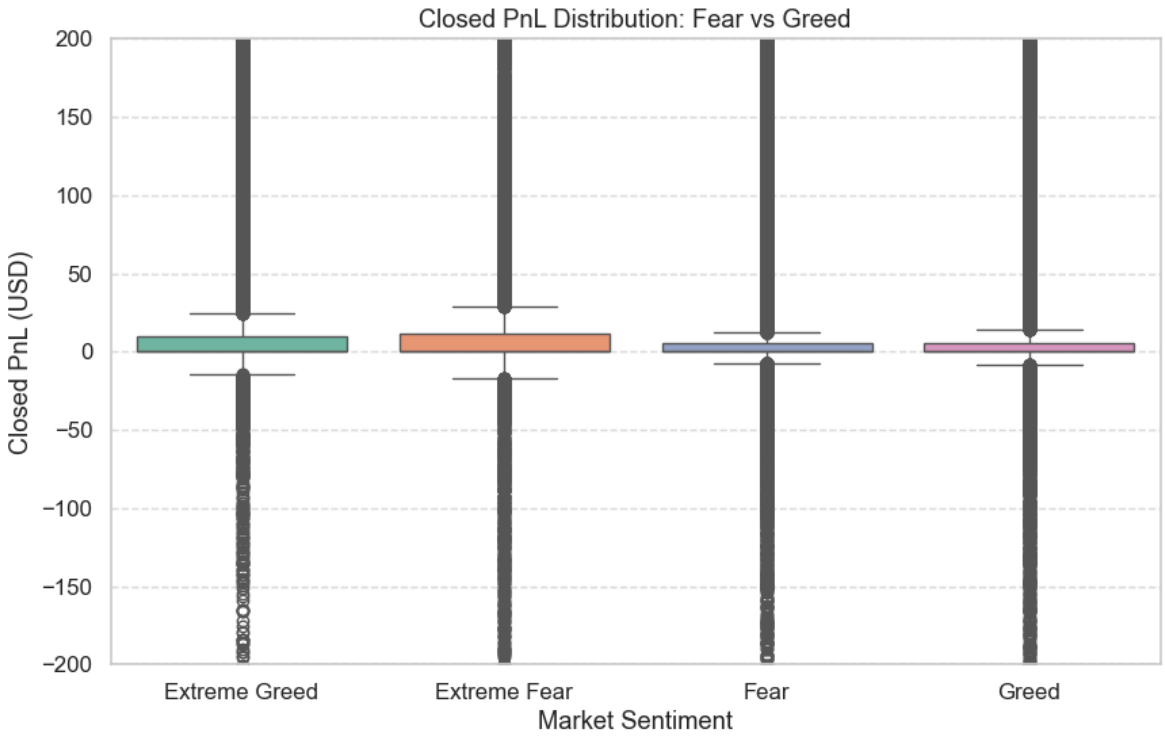
```
N fear: 82813 N greed: 88848
Mann-Whitney U stat: 3648592583.5 p-value: 0.001625431202902836
```

In [33]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Select Fear & Greed trades
pnl_fear = merged.loc[merged['classification'].str.contains('Fear', na=Fa
pnl_greed = merged.loc[merged['classification'].str.contains('Greed', na=

# Statistical Test
print(f"N Fear trades: {len(pnl_fear)}, N Greed trades: {len(pnl_greed)}"

u_stat, p_value = stats.mannwhitneyu(pnl_fear, pnl_greed, alternative='tw
print(f"Mann-Whitney U statistic: {u_stat:.2f}, p-value: {p_value:.6f}")

if p_value < 0.05:
```

```
    print("Statistically significant difference in PnL between Fear & Gre
else:
    print("No significant difference in PnL between Fear & Greed periods.

# Visualization
plt.figure(figsize=(10,6))
sns.boxplot(x=merged['classification'].where(merged['classification'].str
            y=merged['closed_pnl'], palette='Set2')
plt.ylim(-200, 200)  # Limit extreme outliers for better visibility
plt.title("Closed PnL Distribution: Fear vs Greed")
plt.xlabel("Market Sentiment")
plt.ylabel("Closed PnL (USD)")
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.show()
```

```
N Fear trades: 82813, N Greed trades: 88848
Mann-Whitney U statistic: 3648592583.50, p-value: 0.001625
Statistically significant difference in PnL between Fear & Greed periods.
```

```
/var/folders/vb/bg19thz9629cgzg6f4mnrlg00000gn/T/ipykernel_45042/111657300
9.py:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.

  sns.boxplot(x=merged['classification'].where(merged['classification'].st
r.contains('Fear|Greed', na=False)),
```



In [28]:
```
if 'leverage' in merged.columns:
    lev_by_sent = merged.groupby('classification')['leverage'].describe()
    display(lev_by_sent)
    lev_by_sent.to_csv(os.path.join(CSV_DIR,'leverage_by_sentiment.csv'),
```

In [29]:
```
# top 1% by trade value
threshold = merged['trade_value'].quantile(0.99)
large_trades = merged[merged['trade_value'] >= threshold]
```

```
print("Large trades count:", large_trades.shape[0])
print(large_trades['classification'].value_counts())
```

```
Large trades count: 2113
classification
Fear             984
Greed            466
Neutral          313
Extreme Fear     223
Extreme Greed    127
Name: count, dtype: int64
```

In [30]:
```
merged['trade_date'] = pd.to_datetime(merged['trade_date'])
daily = merged.groupby('trade_date').agg(
    daily_trades=('account','count'),
    daily_volume=('trade_value','sum'),
    daily_avg_pnl=('closed_pnl','mean'),
    daily_avg_sentiment=('value','mean')
).reset_index().sort_values('trade_date')

# rolling 7-day
daily['vol_7d'] = daily['daily_volume'].rolling(7, min_periods=1).mean()
daily['pnl_7d'] = daily['daily_avg_pnl'].rolling(7, min_periods=1).mean()

fig = plt.figure()
plt.plot(daily['trade_date'], daily['vol_7d'])
plt.title('7-day rolling volume')
plt.xticks(rotation=30)
save_fig(fig, 'rolling_volume_7d.png')
plt.show()
```

```
Saved: /Users/ranjeetamashal/Desktop/ds_ranjeetamashal/outputs/rolling_vol
ume_7d.png
```