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

!pip install openpyxl
import pandas as pd

df = pd.read_excel('atliennonative.xlsx', engine='openpyxl')
# Using pd.read_excel with the 'openpyxl' engine
```

```
Requirement already satisfied: openpyxl in
/usr/local/lib/python3.11/dist-packages (3.1.5)
Requirement already satisfied: et-xmlfile in
/usr/local/lib/python3.11/dist-packages (from openpyxl) (2.0.0)
```

```
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 520,\n  \"fields\": [\
n    {\n      \"column\": \"Direction\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"put\",\n          \"call\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Order\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"num_unique_values\": 520,\n
\"samples\": [\n          \"8d15bf80-a982-4e06-b29e-47ded5917993\",\n
\"2b8e56b7-c7ef-4d8c-b1ee-617382f42c0d\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Expiration\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 17,\n        \"samples\": [\n
\"S45\",\n          \"S60\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Asset\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 91,\n
\"samples\": [\n          \"OMR/CNY OTC\",\n          \"Apple OTC\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Open time\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 518,\n        \"samples\": [\n          \"2024-
08-24 00:10:00\",\n          \"2024-09-17 23:23:08\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Close time\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 518,\n        \"samples\": [\n          \"2024-
08-24 00:10:40\",\n          \"2024-09-17 23:23:53\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Open price\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
8256.682915499929,\n        \"min\": 1.0753e-05,\n        \"max\":
```

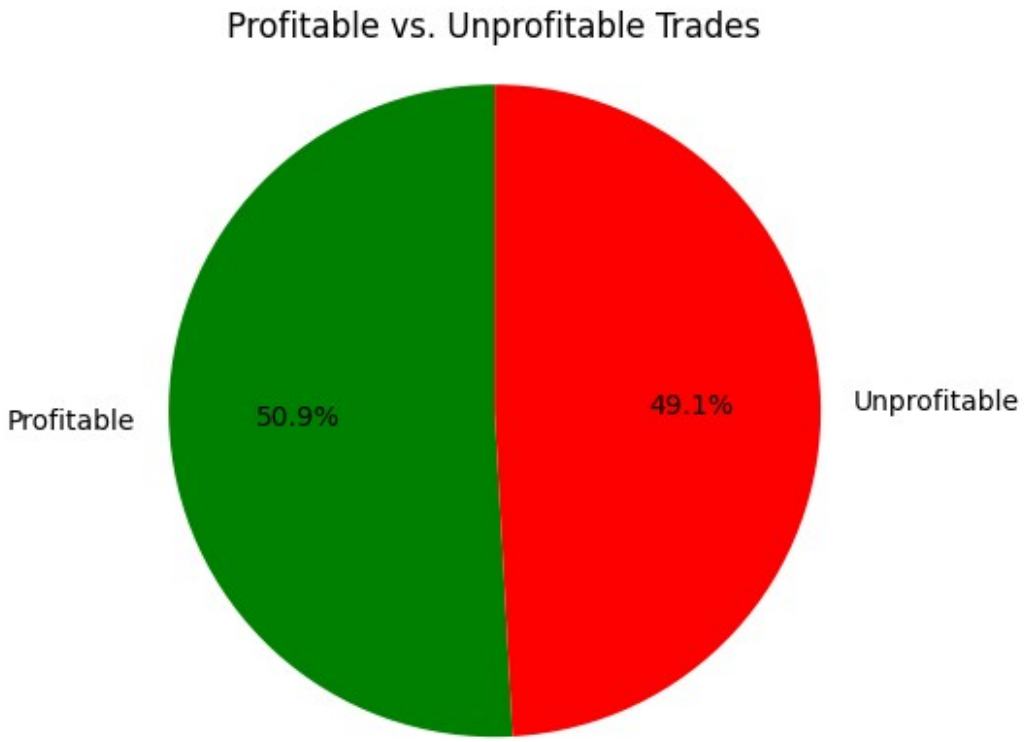
```
89317.048,\n        \"num_unique_values\": 518,\n        \"samples\":
[\n          3790.85,\n          1.30074\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Close price\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
8256.746341583621,\n        \"min\": 1.0696e-05,\n        \"max\":
89317.07,\n        \"num_unique_values\": 518,\n        \"samples\":
[\n          0.93031,\n          1.30064\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Trade amount\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
120.43493955843532,\n        \"min\": 0.0,\n        \"max\": 790.0,\n
\"num_unique_values\": 180,\n        \"samples\": [\n          125.0,\
n          109.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Profit\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 177.14756098600077,\n        \"min\": -682.0,\n
\"max\": 679.88,\n        \"num_unique_values\": 272,\n
\"samples\": [\n          73.6,\n          -169.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Currency\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
1,\n        \"samples\": [\n          \"USD\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


##(First requirement)

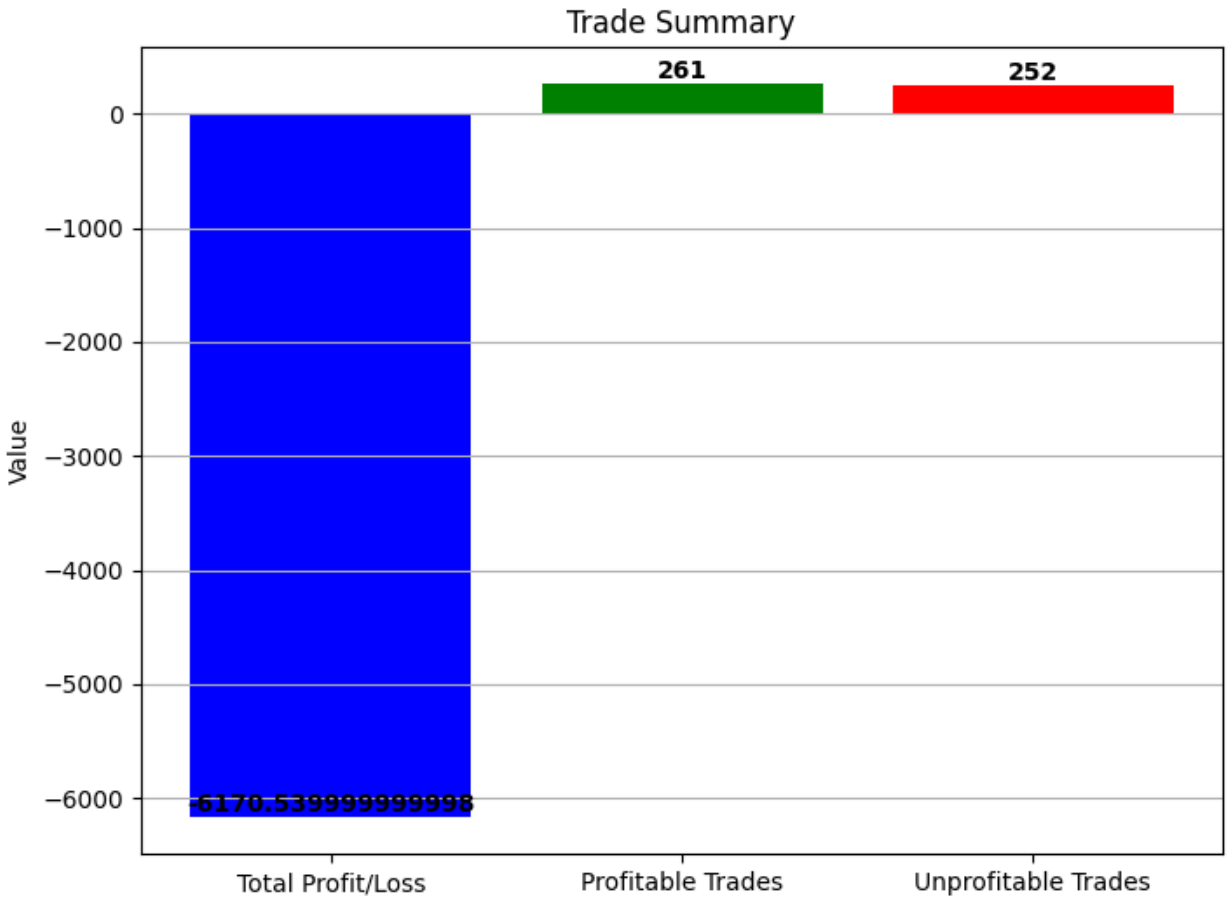
#Analyze the trader's dollar amounts that he put in throughout the year of 2024 to see how he could've adjusted the dollar amount at times to end up profitable at the end of the year#

```
# Assuming 'total_profit' is already calculated as you showed in your
code snippet
total_profit = df["Profit"].sum()

# Create a bar chart
plt.figure(figsize=(6, 4))  # Adjust figure size as needed
plt.bar(["2024"], [total_profit], color=['green' if total_profit > 0
else 'red'])
plt.title('Total Profit/Loss for 2024')
plt.ylabel('Profit/Loss ($)')
plt.grid(axis='y')

# Add the profit/loss value as text on the bar
plt.text(0, total_profit, f'${total_profit:.2f}', ha='center',
va='bottom', color='black', fontweight='bold')

plt.show()
```


```
import matplotlib.pyplot as plt

# ... (Your existing code to calculate profitable_trades and
unprofitable_trades) ...

# Create a pie chart
trade_types = ['Profitable', 'Unprofitable']
trade_counts = [len(profitable_trades), len(unprofitable_trades)]

plt.pie(trade_counts, labels=trade_types, colors=['green', 'red'],
autopct='%1.1f%%', startangle=90)
plt.title('Profitable vs. Unprofitable Trades')
# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')

plt.show()
```

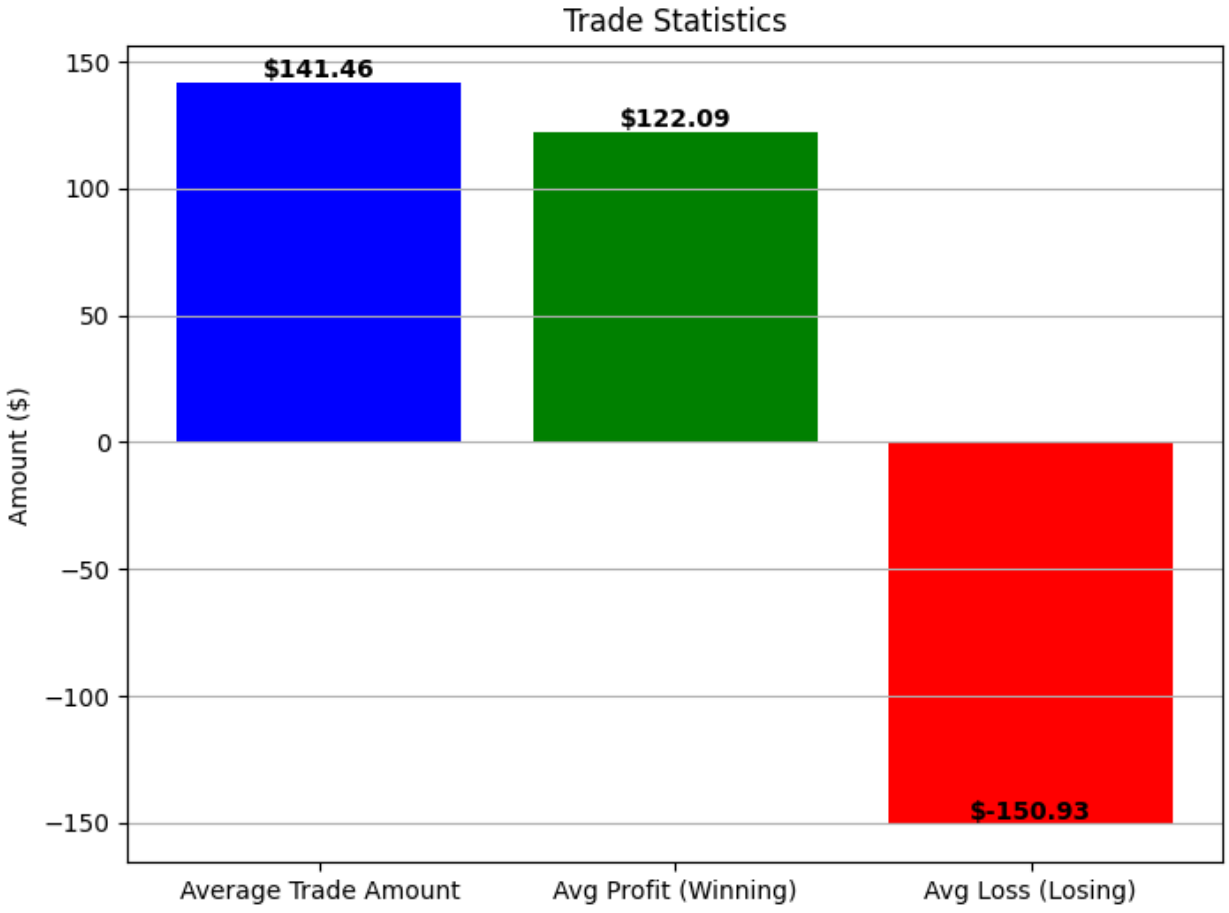



```
# Assuming you have already calculated 'total_profit',
'profitable_trades', and 'unprofitable_trades'
summary_data = {
    "Total Profit/Loss": total_profit,
    "Profitable Trades": len(profitable_trades),
    "Unprofitable Trades": len(unprofitable_trades)
}

categories = list(summary_data.keys())
values = list(summary_data.values())

plt.figure(figsize=(8, 6))  # Adjust figure size if needed
plt.bar(categories, values, color=['blue', 'green', 'red'])
plt.title('Trade Summary')
plt.ylabel('Value')
plt.grid(axis='y')

# Add values on top of bars
for i, v in enumerate(values):
    plt.text(i, v, str(v), ha='center', va='bottom',
fontweight='bold')

plt.show()
```



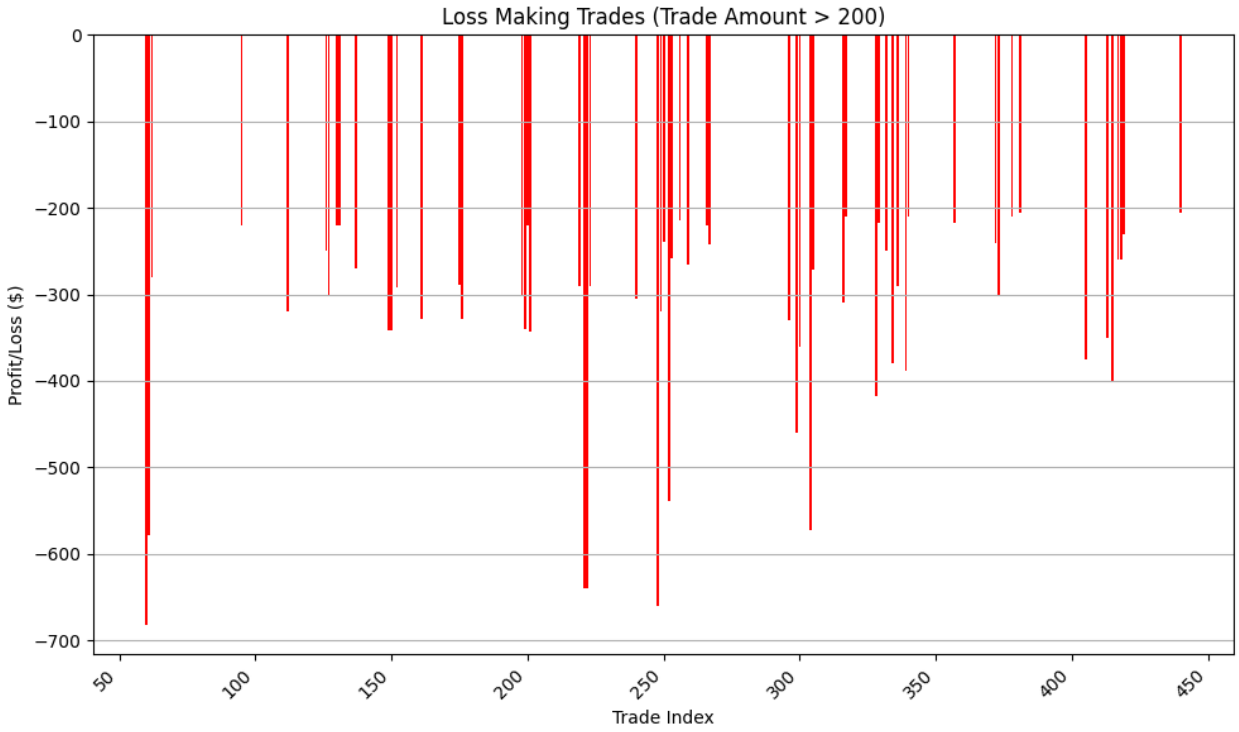

```
import matplotlib.pyplot as plt

# Assuming you have already calculated these values:
average_trade_amount = df['Trade amount'].mean()
avg_profit_winning_trades = profitable_trades['Profit'].mean()
avg_loss_losing_trades = unprofitable_trades['Profit'].mean()

# Create a bar chart
categories = ['Average Trade Amount', 'Avg Profit (Winning)', 'Avg
Loss (Losing)']
values = [average_trade_amount, avg_profit_winning_trades,
avg_loss_losing_trades]

plt.figure(figsize=(8, 6))  # Adjust figure size if needed
plt.bar(categories, values, color=['blue', 'green', 'red'])
plt.title('Trade Statistics')
plt.ylabel('Amount ($)')
plt.grid(axis='y')

# Add values on top of bars for better readability
for i, v in enumerate(values):
    plt.text(i, v, f'${v:.2f}', ha='center', va='bottom',
```

```
fontweight='bold')

plt.show()
```

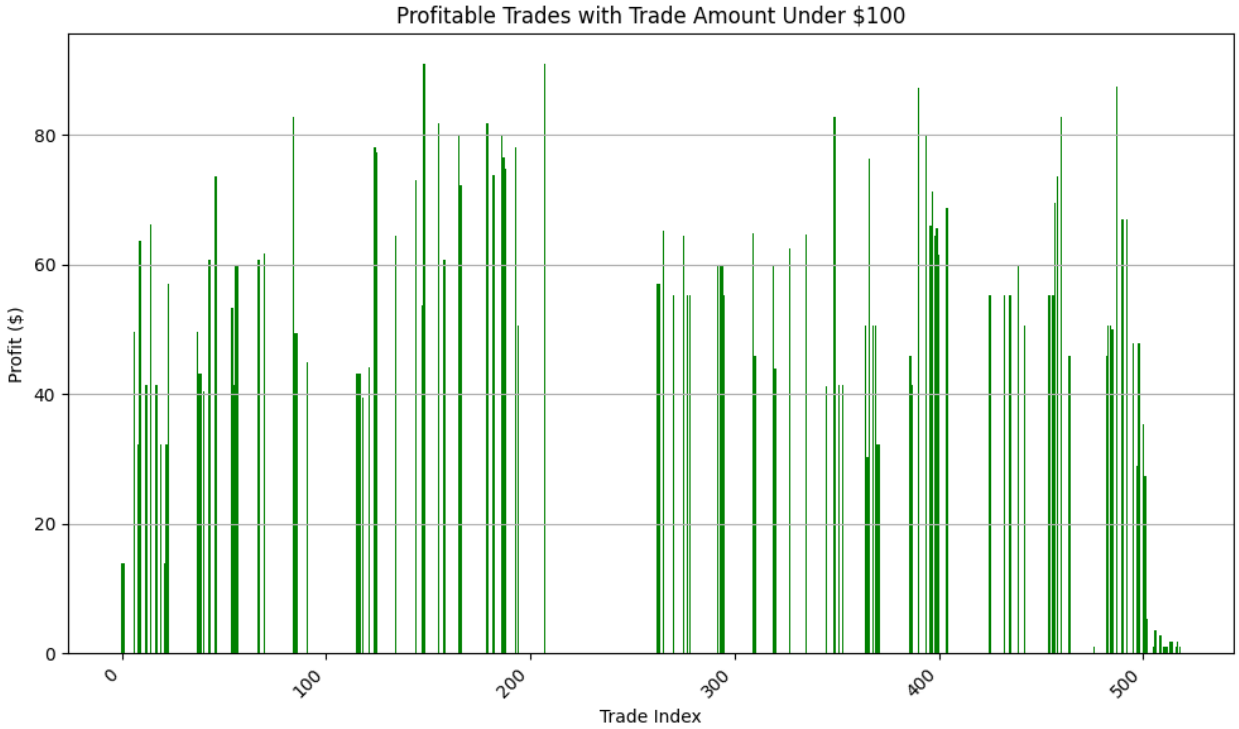



```
# In cell 'ipython-input-20-a35a43b2577a'
# Assign the result of the query to 'loss_trades'
loss_trades = df.query("Profit < 0 and `Trade amount` > 200")  #
Adjust the $200 threshold as needed

# Now, 'ipython-input-36-a35a43b2577a' should work without the
NameError
import matplotlib.pyplot as plt

# Assuming 'loss_trades' DataFrame from the previous step
plt.figure(figsize=(10, 6))  # Adjust figure size if needed
plt.bar(loss_trades.index, loss_trades['Profit'], color='red')
plt.title('Loss Making Trades (Trade Amount > 200)')
plt.xlabel('Trade Index')
plt.ylabel('Profit/Loss ($)')
plt.grid(axis='y')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better
```

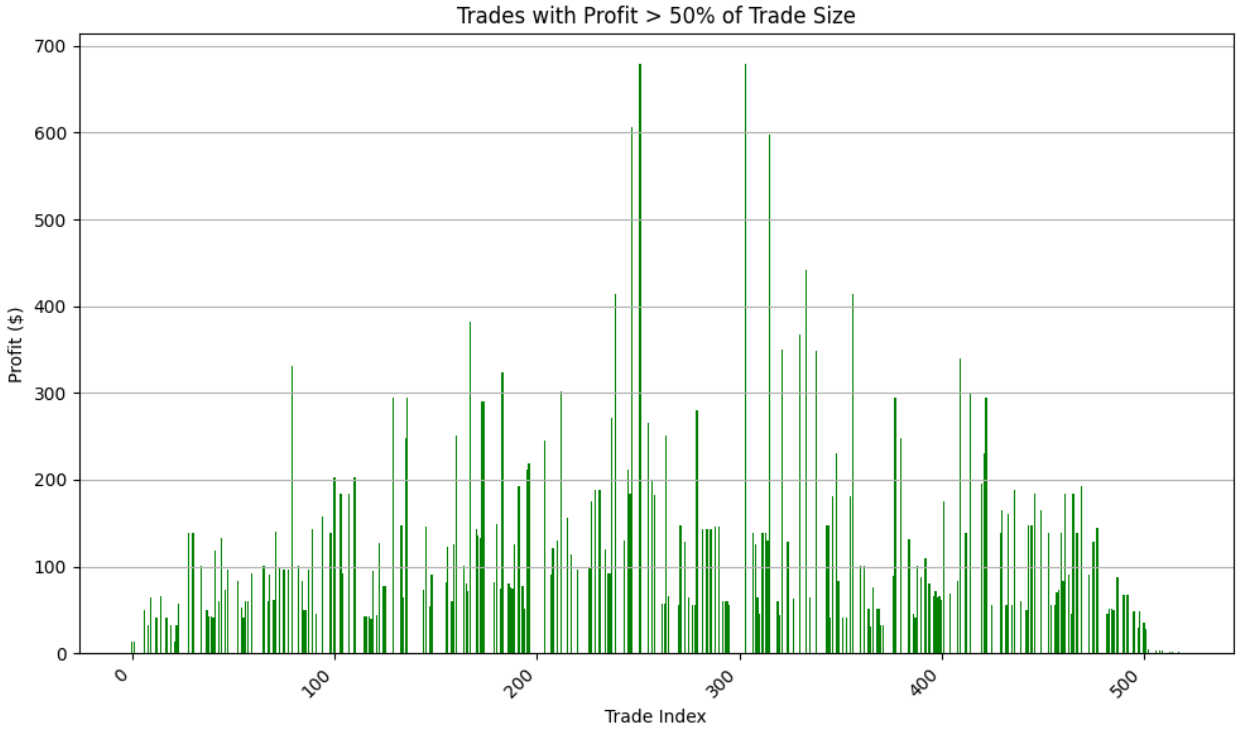
```
readability
plt.tight_layout()  # Adjust layout to prevent labels from overlapping
plt.show()
```



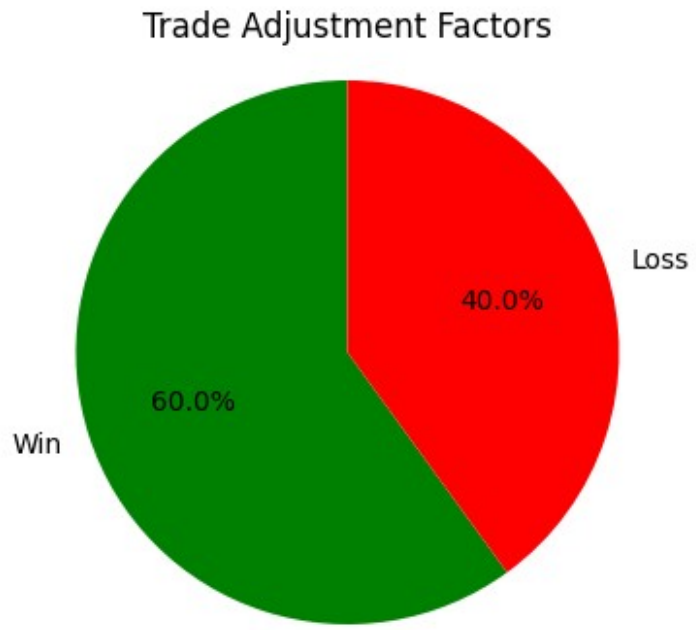

```
import matplotlib.pyplot as plt

# Define profitable_trades_under_100
profitable_trades_under_100 = df.query("Profit > 0 and `Trade amount` 
< 100")  # Adjust the $100 threshold as needed


plt.figure(figsize=(10, 6))  # Adjust figure size as needed
plt.bar(profitable_trades_under_100.index, 
profitable_trades_under_100['Profit'], color='green')
plt.title('Profitable Trades with Trade Amount Under $100')
plt.xlabel('Trade Index')
plt.ylabel('Profit ($)')
plt.grid(axis='y')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better 
readability
plt.tight_layout()  # Adjust layout to prevent labels from overlapping
plt.show()
```



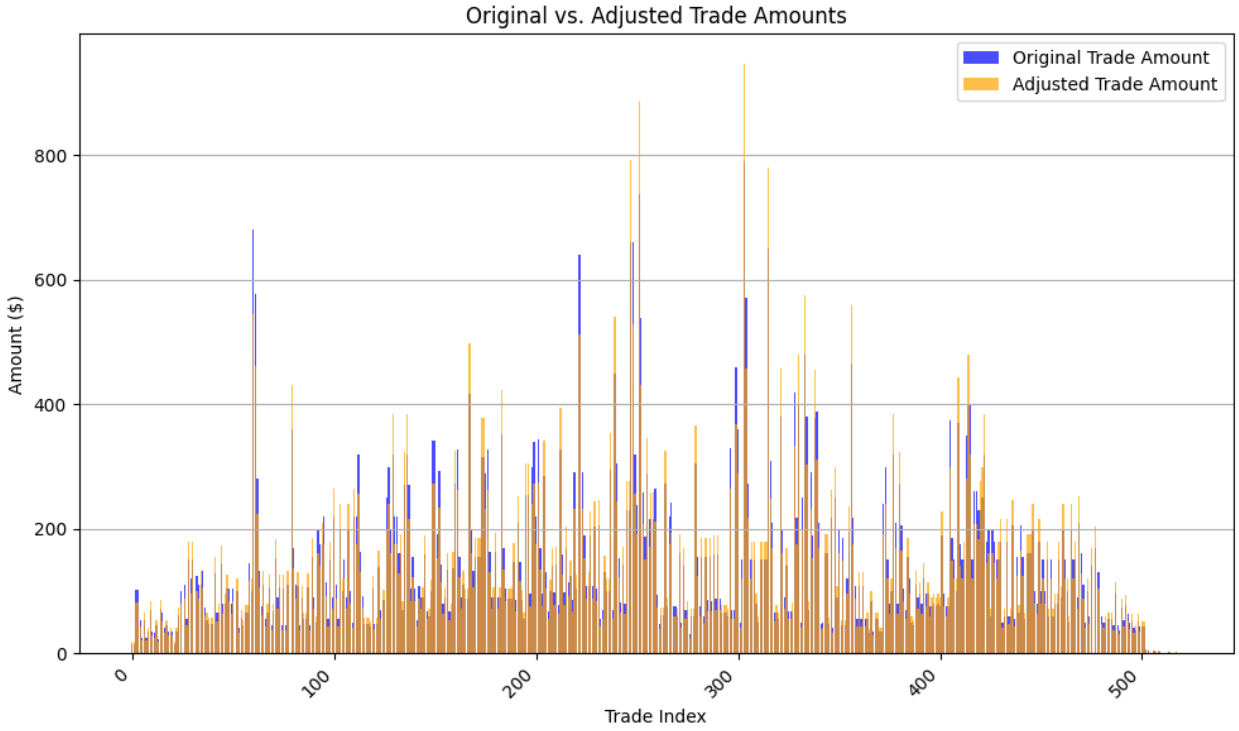

```
import matplotlib.pyplot as plt

# Define profitable_trades_over_50_percent by assigning the result of
the query
profitable_trades_over_50_percent = df.query("Profit / `Trade amount`
> 0.5")  # Trades where profit was >50% of trade size

plt.figure(figsize=(10, 6))  # Adjust figure size as needed
plt.bar(profitable_trades_over_50_percent.index,
profitable_trades_over_50_percent['Profit'], color='green')
plt.title('Trades with Profit > 50% of Trade Size')
plt.xlabel('Trade Index')
plt.ylabel('Profit ($)')
plt.grid(axis='y')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better
readability
plt.tight_layout()  # Adjust layout to prevent labels from overlapping
plt.show()
```



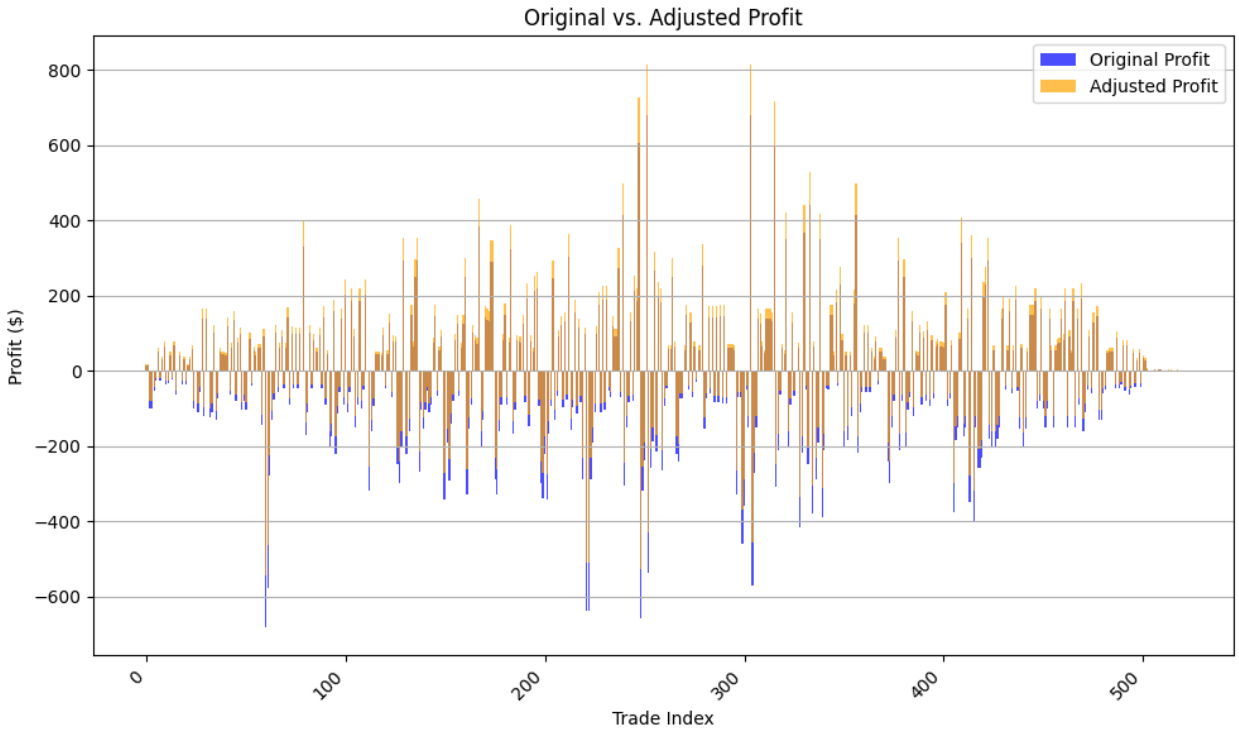

```
import matplotlib.pyplot as plt

trade_types = ['Win', 'Loss']
adjustment_factors = [win_increase_factor, loss_decrease_factor]

# Creating the pie chart
plt.figure(figsize=(6, 4))  # Adjust figure size if needed
plt.pie(adjustment_factors, labels=trade_types, colors=['green',
'red'], autopct='%1.1f%%', startangle=90)
plt.title('Trade Adjustment Factors')
# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')

plt.show()
```



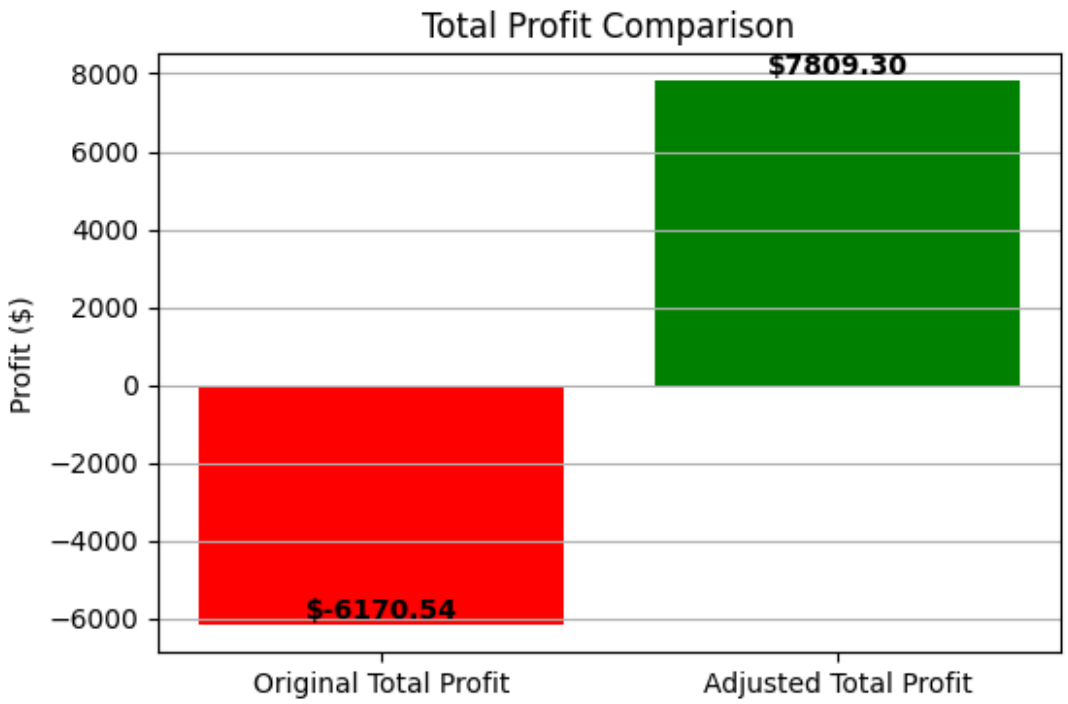

```
trade_types = ['Win', 'Loss']
adjustment_factors = [win_increase_factor, loss_decrease_factor]

import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame with 'Trade amount' and 'Adjusted
Trade Amount' columns

plt.figure(figsize=(10, 6))  # Adjust figure size as needed
plt.bar(df.index, df['Trade amount'], label='Original Trade Amount',
color='blue', alpha=0.7)  # Original amounts in blue
plt.bar(df.index, df['Adjusted Trade Amount'], label='Adjusted Trade
Amount', color='orange', alpha=0.7)  # Adjusted amounts in orange
plt.title('Original vs. Adjusted Trade Amounts')
plt.xlabel('Trade Index')
plt.ylabel('Amount ($)')
plt.legend()
plt.grid(axis='y')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better
readability
plt.tight_layout()  # Adjust layout to prevent labels from overlapping
plt.show()
```



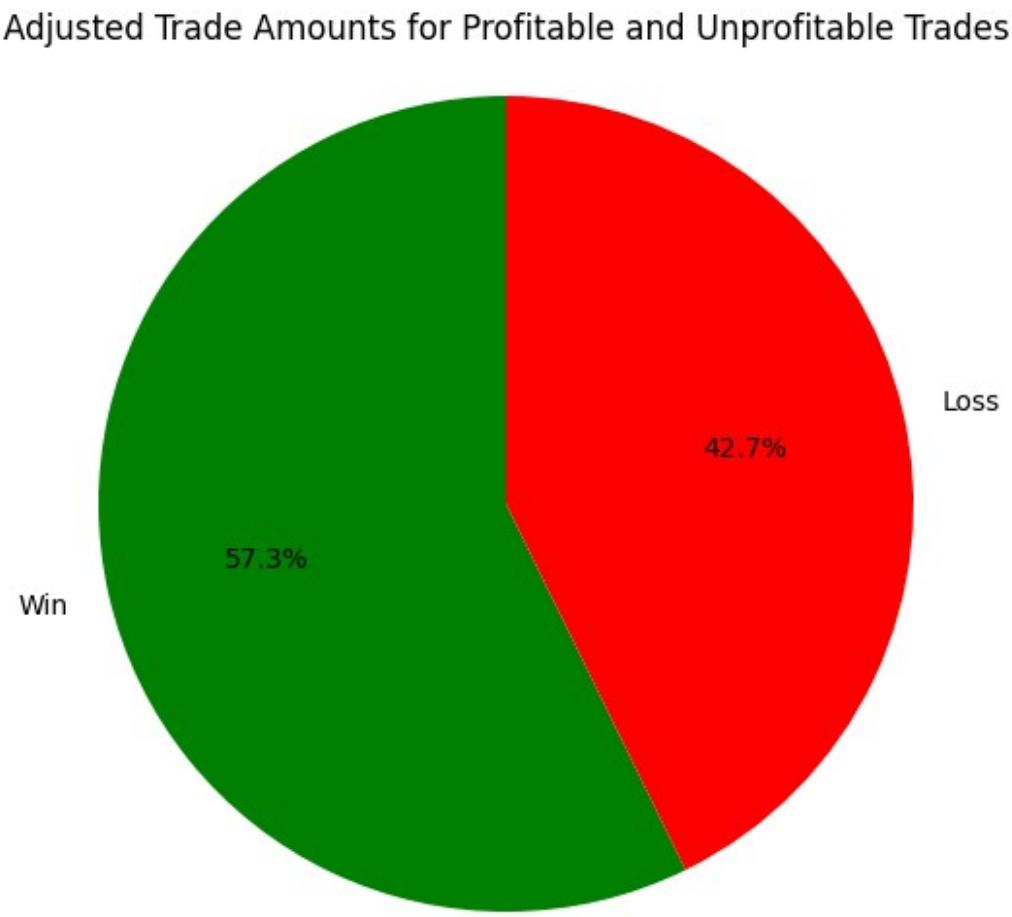

```
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame with 'Profit' and 'Adjusted Profit'
columns

plt.figure(figsize=(10, 6))  # Adjust figure size as needed
plt.bar(df.index, df['Profit'], label='Original Profit', color='blue',
alpha=0.7)  # Original profit in blue
plt.bar(df.index, df['Adjusted Profit'], label='Adjusted Profit',
color='orange', alpha=0.7)  # Adjusted profit in orange
plt.title('Original vs. Adjusted Profit')
plt.xlabel('Trade Index')
plt.ylabel('Profit ($)')
plt.legend()
plt.grid(axis='y')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better
readability
plt.tight_layout()  # Adjust layout to prevent labels from overlapping
plt.show()
```




```
import matplotlib.pyplot as plt

# Assuming you have already calculated 'total_profit' (original) and
'adjusted_total_profit'

profit_types = ['Original Total Profit', 'Adjusted Total Profit']
profit_values = [total_profit, adjusted_total_profit]

# Define colors based on profit/loss
colors = ['green' if v > 0 else 'red' for v in profit_values]

plt.figure(figsize=(6, 4))  # Adjust figure size if needed
plt.bar(profit_types, profit_values, color=colors)  # Use the
calculated colors
plt.title('Total Profit Comparison')
plt.ylabel('Profit ($)')
plt.grid(axis='y')

# Add values on top of bars for better readability
for i, v in enumerate(profit_values):
    plt.text(i, v, f'${v:.2f}', ha='center', va='bottom',
fontweight='bold')

plt.show()
```




```
import matplotlib.pyplot as plt

# Assuming 'trade_types' and 'adjusted_trade_amounts' are already
defined
# For example:
# trade_types = ['Win', 'Loss']
# adjusted_trade_amounts = [120, 80]  # Replace with your actual
values

plt.figure(figsize=(8, 6))  # Adjust figure size if needed
plt.pie(adjusted_trade_amounts, labels=trade_types, colors=['green',
'red'], autopct='%1.1f%%', startangle=90)
plt.title('Adjusted Trade Amounts for Profitable and Unprofitable
Trades')
# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')

plt.show()
```



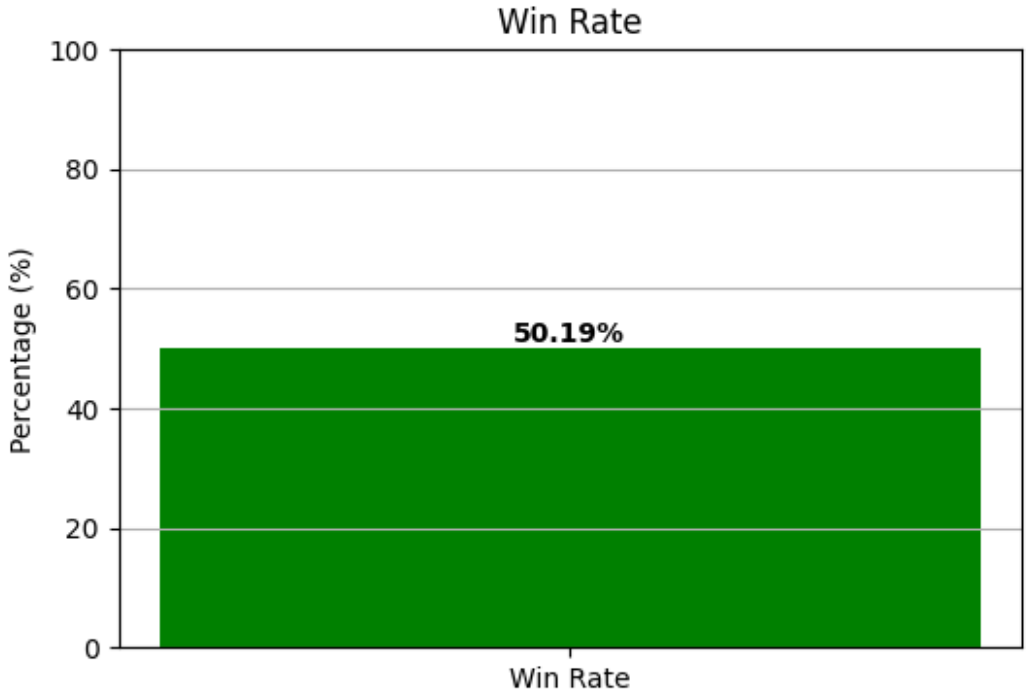
##(Third Requirement)

I don’t need technical analysis advice only how to better use the numbers for profitability at the end of the year …… I’m also trying to find a consistent method that the trader should use according to his typical win/loss patterns in the market

Win/Loss Analysis

```
import matplotlib.pyplot as plt

# ... (Your existing code to calculate win_rate) ...
```

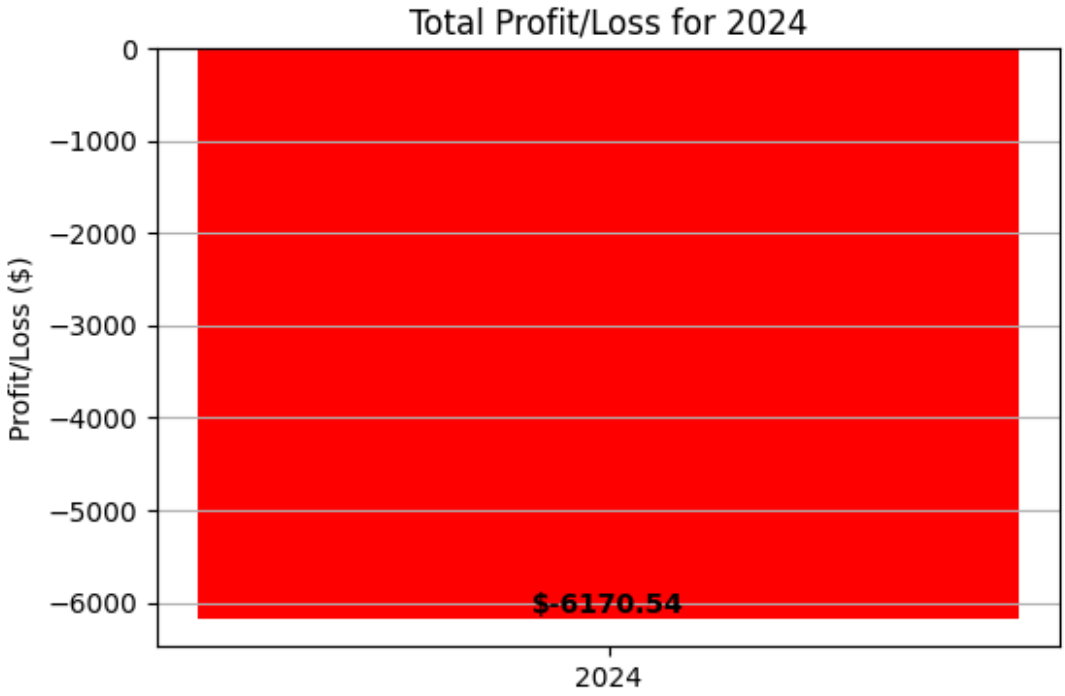
```
# Create a bar chart to visualize the win rate
plt.figure(figsize=(6, 4))  # Adjust figure size as needed
plt.bar(['Win Rate'], [win_rate], color=['green'])
plt.title('Win Rate')
plt.ylabel('Percentage (%)')
plt.ylim(0, 100)  # Set y-axis limits to 0-100%
plt.grid(axis='y')

# Add the win rate value as text on the bar
plt.text(0, win_rate, f'{win_rate:.2f}%', ha='center', va='bottom',
color='black', fontweight='bold')

plt.show()
```



Total Profit/Loss for 2024

```
# Calculate total profit/loss for the year
total_profit = df['Profit'].sum()

# Create a bar chart
plt.figure(figsize=(6, 4))  # Adjust figure size as needed
plt.bar(["2024"], [total_profit], color=['green' if total_profit > 0
else 'red'])
plt.title('Total Profit/Loss for 2024')
plt.ylabel('Profit/Loss ($)')
plt.grid(axis='y')
```

```
# Add the profit/loss value as text on the bar
plt.text(0, total_profit, f'${total_profit:.2f}', ha='center',
va='bottom', color='black', fontweight='bold')

plt.show()
```



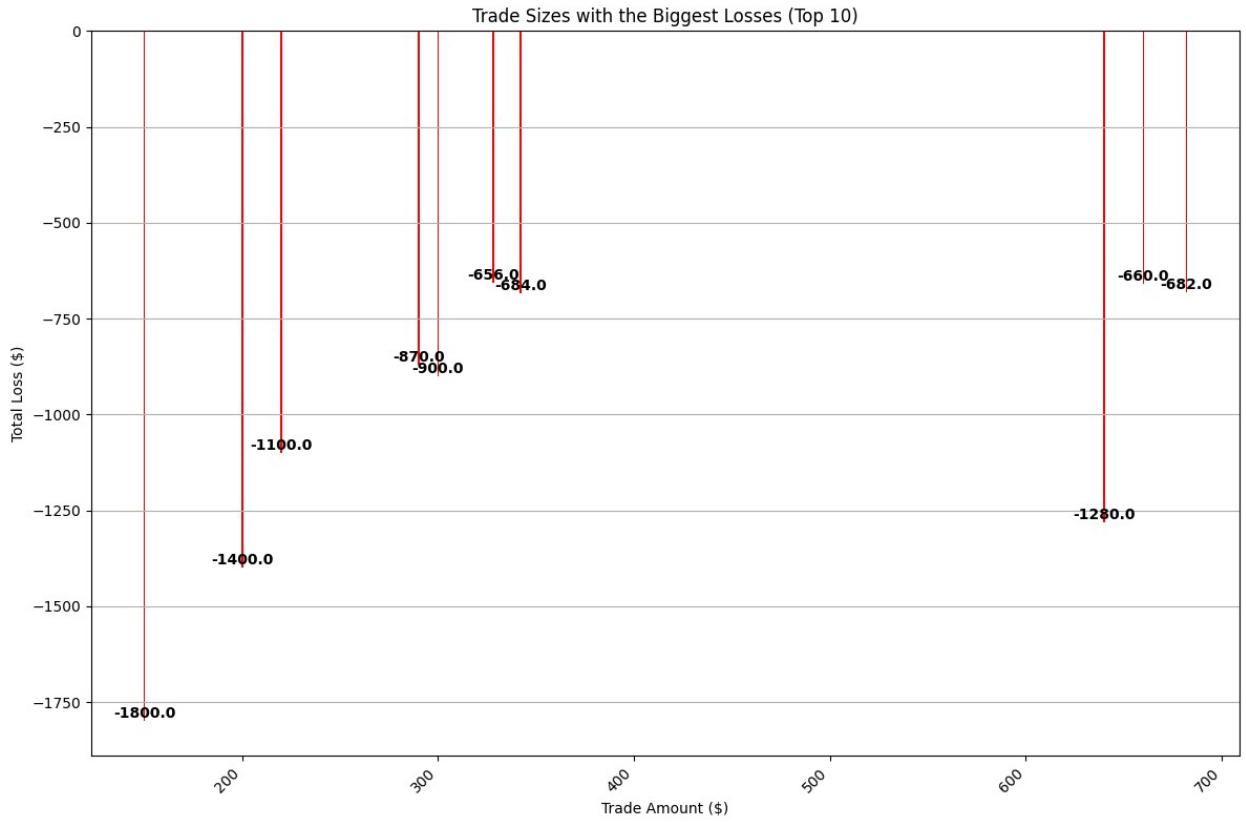
(Fourth Requirement)

#Also it needs to be identified where the Trader is going wrong with the dollar amounts He is putting up since he didn't end up profitable at the end of the year

##Average Trade Size and Risk Assessment This shows if the trader put too much money on bad trades

```
import matplotlib.pyplot as plt

# ... (Your existing code to calculate losing_trades) ...

# Display the worst trade sizes in a chart
plt.figure(figsize=(12, 8))  # Increased figure size (width, height)
bars = plt.bar(losing_trades['Trade amount'].head(10),
losing_trades['sum'].head(10), color='red')  # Use 'sum' for total
loss
plt.title('Average Trade Size and Risk Assessment')
plt.xlabel('Trade Amount ($)')
plt.ylabel('Total Loss ($)')
plt.grid(axis='y')
```

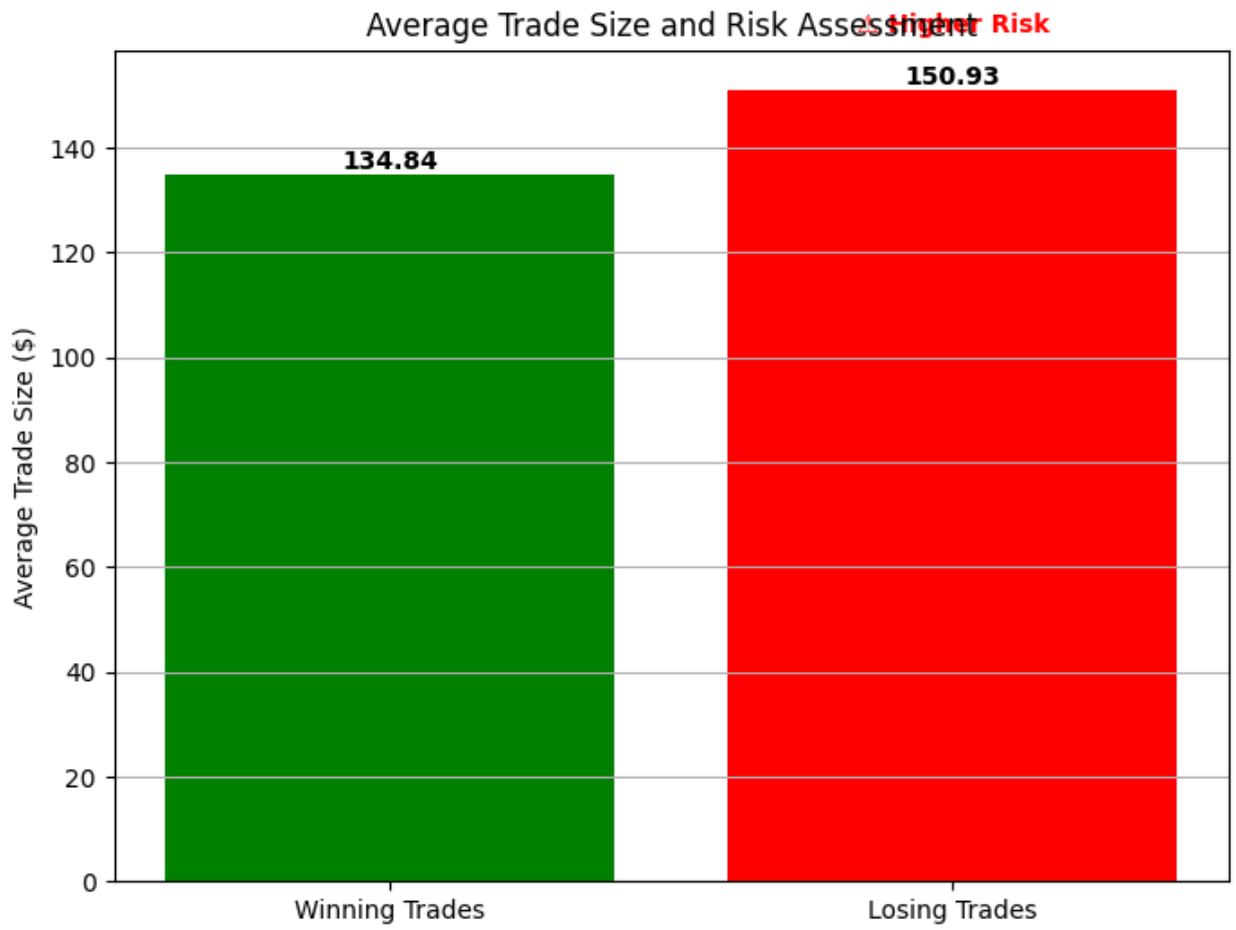
```
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent labels from overlapping

# Add values on top of bars
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), ha='center', va='bottom', color='black', fontweight='bold')

plt.show()
```



##Average Trade Size and Risk Assessment If the trader risked too much on losses, we can fix this by adjusting trade size

```
import matplotlib.pyplot as plt

# Assuming you have already calculated these values:
avg_win_trade = df[df['Profit'] > 0]['Trade amount'].mean()
avg_loss_trade = df[df['Profit'] < 0]['Trade amount'].mean()

# Create a bar chart
trade_types = ['Winning Trades', 'Losing Trades']
avg_trade_sizes = [avg_win_trade, avg_loss_trade]
```

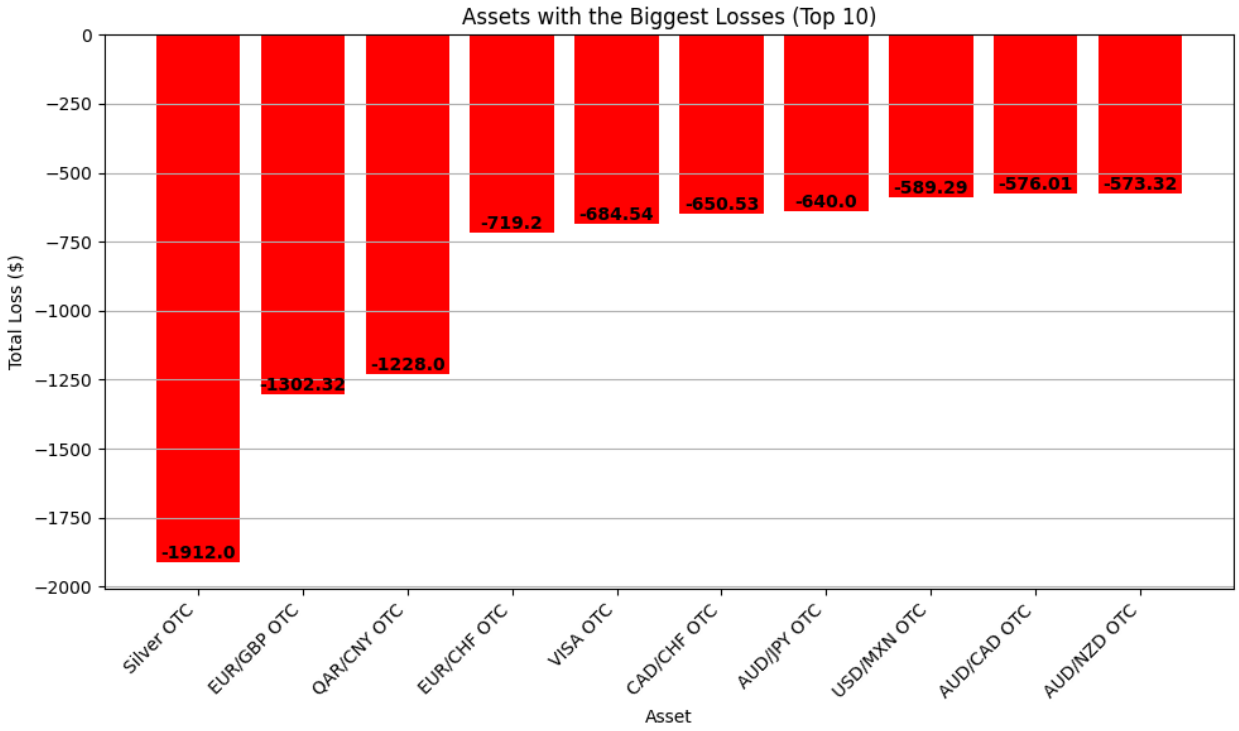
```
plt.figure(figsize=(8, 6))  # Adjust figure size as needed
bars = plt.bar(trade_types, avg_trade_sizes, color=['green', 'red'])
plt.title('Average Trade Size and Risk Assessment')
plt.ylabel('Average Trade Size ($)')
plt.grid(axis='y')

# Add values on top of bars for better readability
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),
ha='center', va='bottom', color='black', fontweight='bold')

# Add risk indicator
if avg_loss_trade > avg_win_trade:
    plt.text(1, avg_loss_trade + 10, "⚠ Higher Risk", ha='center',
va='bottom', color='red', fontweight='bold')  # Adjust position as
needed
else:
    plt.text(1, avg_loss_trade + 10, "□ Lower Risk", ha='center',
va='bottom', color='green', fontweight='bold') # Adjust position as
needed

plt.show()
```



##Assets with the Biggest Losses (Top 10) This shows if the trader focused on assets that didn't work for them

```
import matplotlib.pyplot as plt

# Assuming you have already calculated 'losing_assets'
# ... (your existing code to calculate losing_assets) ...

# Create a bar chart
plt.figure(figsize=(10, 6))  # Adjust figure size as needed
bars = plt.bar(losing_assets['Asset'].head(10),
losing_assets['sum'].head(10), color='red')
plt.title('Assets with the Biggest Losses (Top 10)')
plt.xlabel('Asset')
plt.ylabel('Total Loss ($)')
plt.grid(axis='y')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better
readability
plt.tight_layout()  # Adjust layout to prevent labels from overlapping

# Add values on top of bars
```

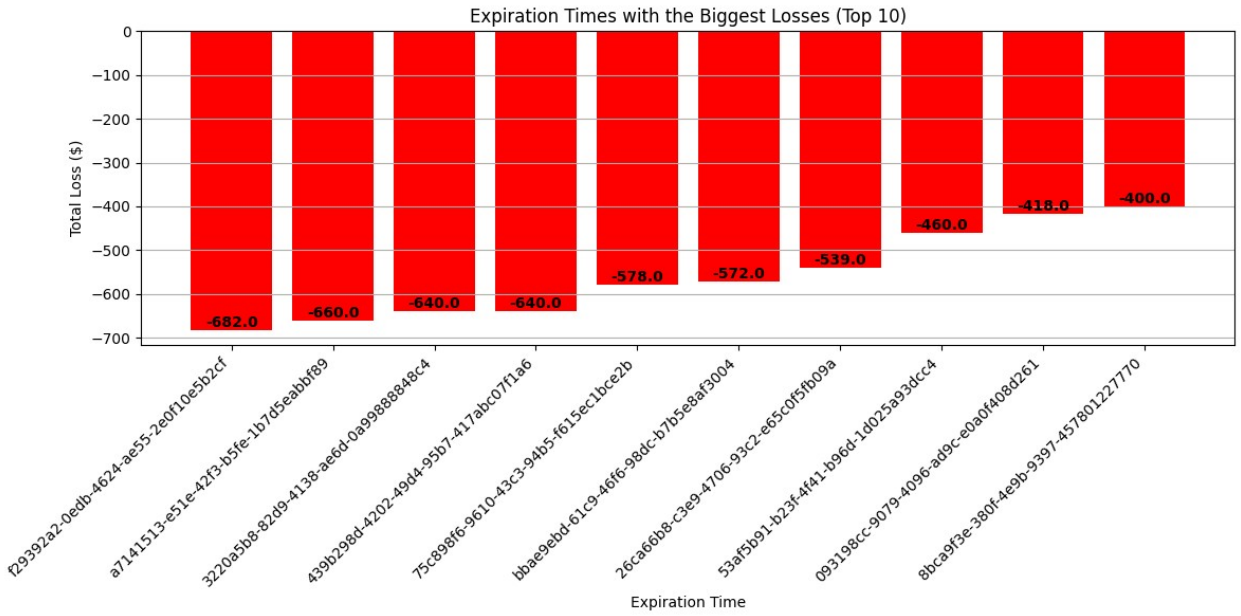
```
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),
ha='center', va='bottom', color='black', fontweight='bold')

plt.show()
```



##Expiration Times with the Biggest Losses (Top 10) If the trader should avoid certain expiration times

```
import matplotlib.pyplot as plt

# Assuming you have already calculated 'losing_expirations'
# ... (your existing code to calculate losing_expirations) ...

plt.figure(figsize=(12, 6))  # Adjust figure size as needed
bars = plt.bar(losing_expirations['Order'].head(10),
losing_expirations['sum'].head(10), color='red')
plt.title('Expiration Times with the Biggest Losses (Top 10)')
plt.xlabel('Expiration Time')
plt.ylabel('Total Loss ($)')
plt.grid(axis='y')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better
readability
plt.tight_layout()  # Adjust layout to prevent labels from overlapping

# Add values on top of bars
```

```
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),
ha='center', va='bottom', color='black', fontweight='bold')

plt.show()
```



(Fifth Requirement)

what should've been the traders average dollar amount per trade if for example he was trading with like $10,000 ..... example If the trader loses a $50 trade what should be the dollar amount on the next trade..... If the trader loses a second trade in a row what should be the dollar amount on that next trade..... If the trader loses three in a row which should be the dollar amount on that next trade..... If the trader loses four trades in a row which should be the next dollar amount on that trade ..... if the trader loses five in a row what should be the next Dollar amount on the next trade....& should the trader keep upping the dollar amount when on a losing streak ..... or go back down to a smaller amount or for example if the traders starting amount is $50 for example on a $10,000 account which that is just .5% should the trader always only do $50 trades or up the amount when they lose a trade to get that $50 back in to get them in the green

##Fixed Trade Size

```
# Parameters
account_balance = 10000
trade_risk_percentage = 0.005  # 0.5% per trade
trade_size = account_balance * trade_risk_percentage  # Always $50
```

```
# Print fixed trade size
print(f"Trade amount per trade: ${trade_size:.2f}")
```

```
Trade amount per trade: $50.00
```

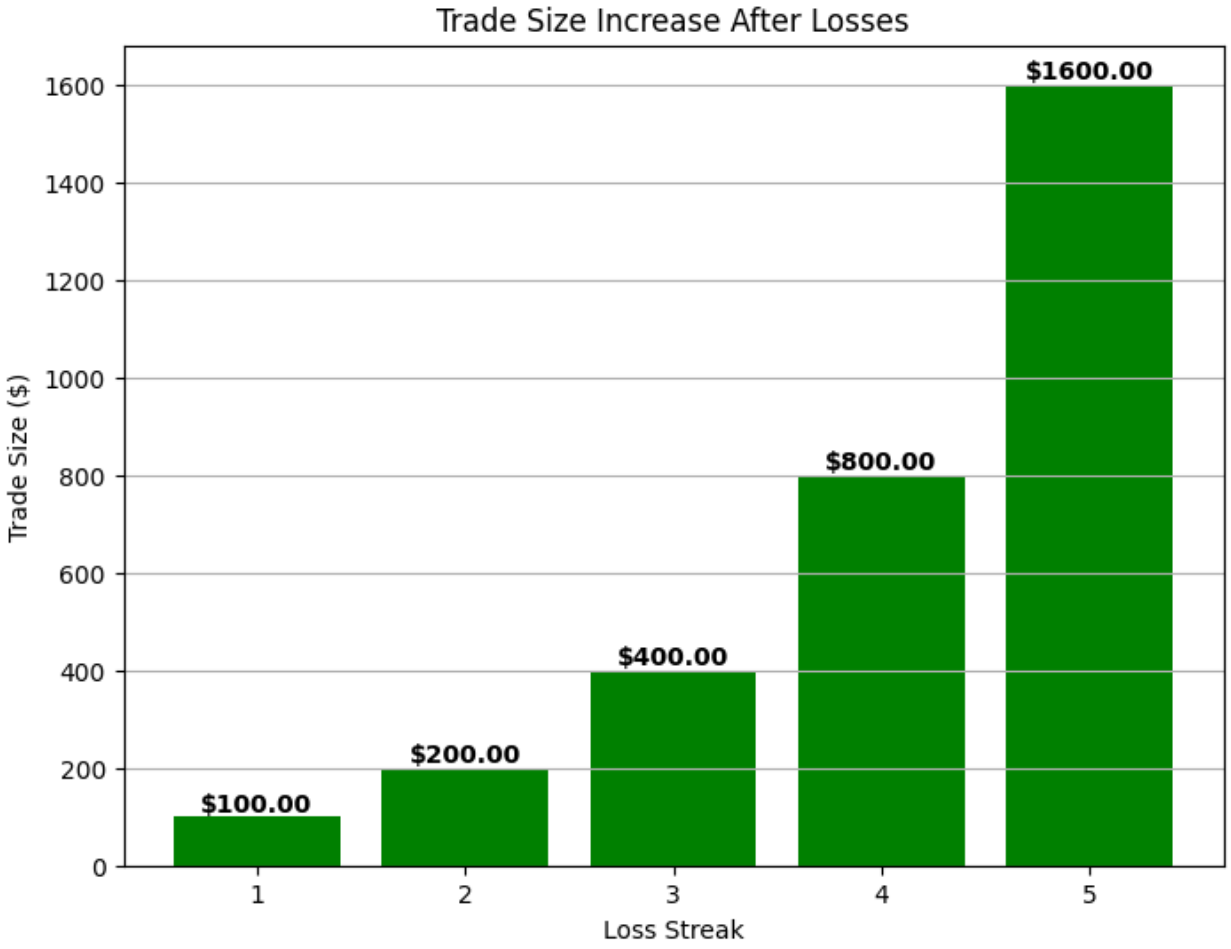
```
# Starting trade size
trade_size = 50
loss_streak = 0

# Simulate 5 losses in a row
for i in range(5):
    trade_size *= 2  # Double after loss
    print(f"After {i+1} losses, next trade amount: ${trade_size:.2f}")
```

```
After 1 losses, next trade amount: $100.00
After 2 losses, next trade amount: $200.00
After 3 losses, next trade amount: $400.00
After 4 losses, next trade amount: $800.00
After 5 losses, next trade amount: $1600.00
```

##Trade Size Increase After Losses

```
import matplotlib.pyplot as plt

# Starting trade size
trade_size = 50
loss_streak = 0
trade_sizes = []  # To store trade sizes for plotting

# Simulate 5 losses in a row and store trade sizes
for i in range(5):
    trade_size *= 2  # Double after loss
    trade_sizes.append(trade_size)

# Create the bar chart
plt.figure(figsize=(8, 6))  # Adjust figure size as needed
plt.bar(range(1, 6), trade_sizes, color='green')  # Bars representing
trade sizes
plt.title('Trade Size Increase After Losses')
plt.xlabel('Loss Streak')
plt.ylabel('Trade Size ($)')
plt.xticks(range(1, 6))  # Set x-axis ticks to represent loss streak
plt.grid(axis='y')

# Add values on top of bars
for i, v in enumerate(trade_sizes):
    plt.text(i + 1, v, f'${v:.2f}', ha='center', va='bottom',
fontweight='bold')

plt.show()
```

```
"If the trader loses 6-7 times in a row, they risk losing most of their account"
```

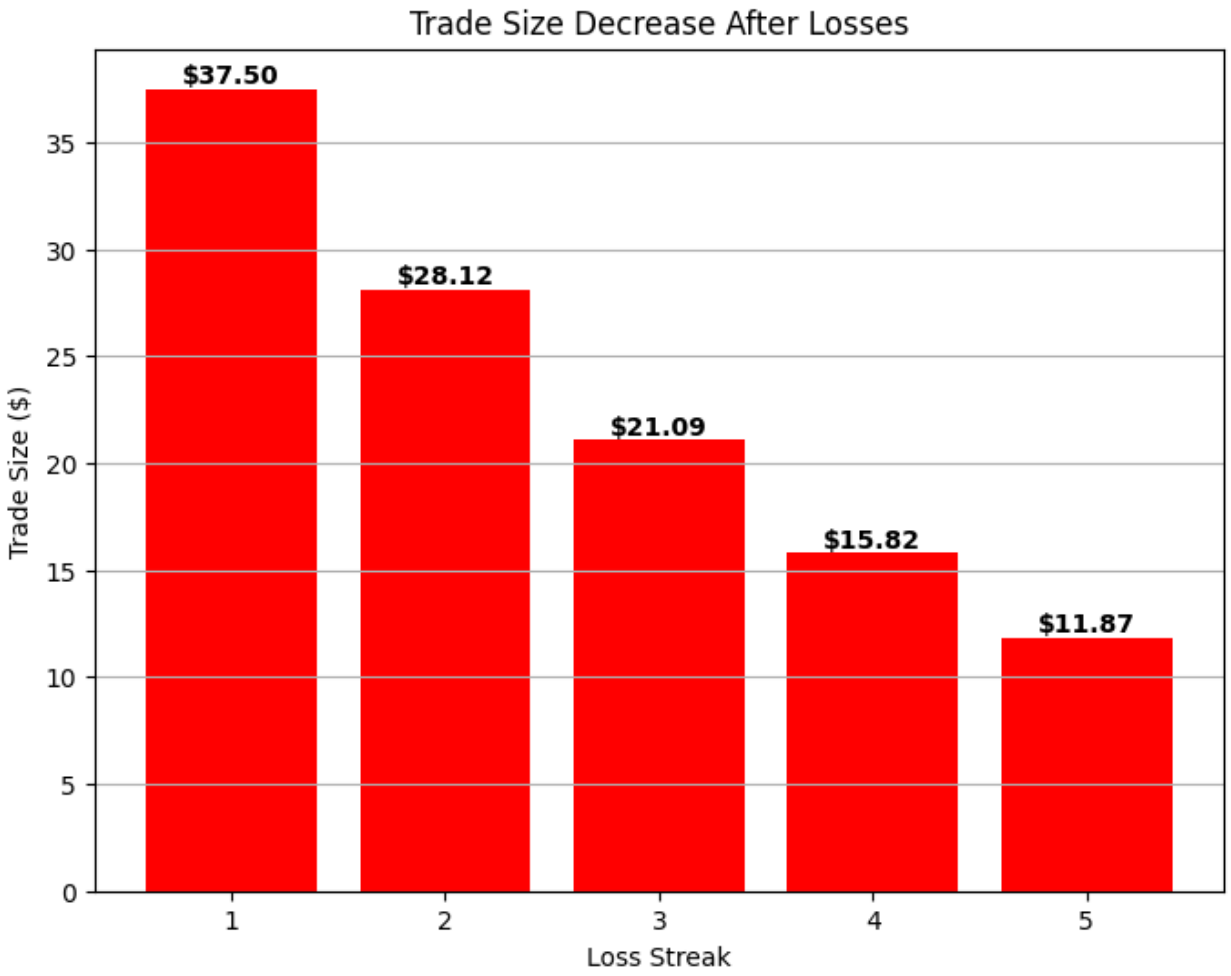



```
{"type":"string"}

# Starting trade size
trade_size = 50
loss_streak = 0

# Simulate 5 losses in a row with decreasing trade size
for i in range(5):
    trade_size *= 0.75  # Reduce size after each loss by 25%
    print(f"After {i+1} losses, next trade amount: ${trade_size:.2f}")
```

```
After 1 losses, next trade amount: $37.50
After 2 losses, next trade amount: $28.12
After 3 losses, next trade amount: $21.09
After 4 losses, next trade amount: $15.82
After 5 losses, next trade amount: $11.87
```

##Trade Size Decrease After Losses

```
import matplotlib.pyplot as plt

# Starting trade size
trade_size = 50
loss_streak = 0
trade_sizes = []  # To store trade sizes for plotting

# Simulate 5 losses in a row with decreasing trade size
for i in range(5):
    trade_size *= 0.75  # Reduce size after each loss by 25%
    trade_sizes.append(trade_size)

# Create the bar chart
plt.figure(figsize=(8, 6))  # Adjust figure size as needed
plt.bar(range(1, 6), trade_sizes, color='red')  # Bars representing
trade sizes
plt.title('Trade Size Decrease After Losses')
plt.xlabel('Loss Streak')
plt.ylabel('Trade Size ($)')
plt.xticks(range(1, 6))  # Set x-axis ticks to represent loss streak
plt.grid(axis='y')

# Add values on top of bars
for i, v in enumerate(trade_sizes):
    plt.text(i + 1, v, f'${v:.2f}', ha='center', va='bottom',
fontweight='bold')

plt.show()
"This method helps limit risk and avoids wiping out the account"
```



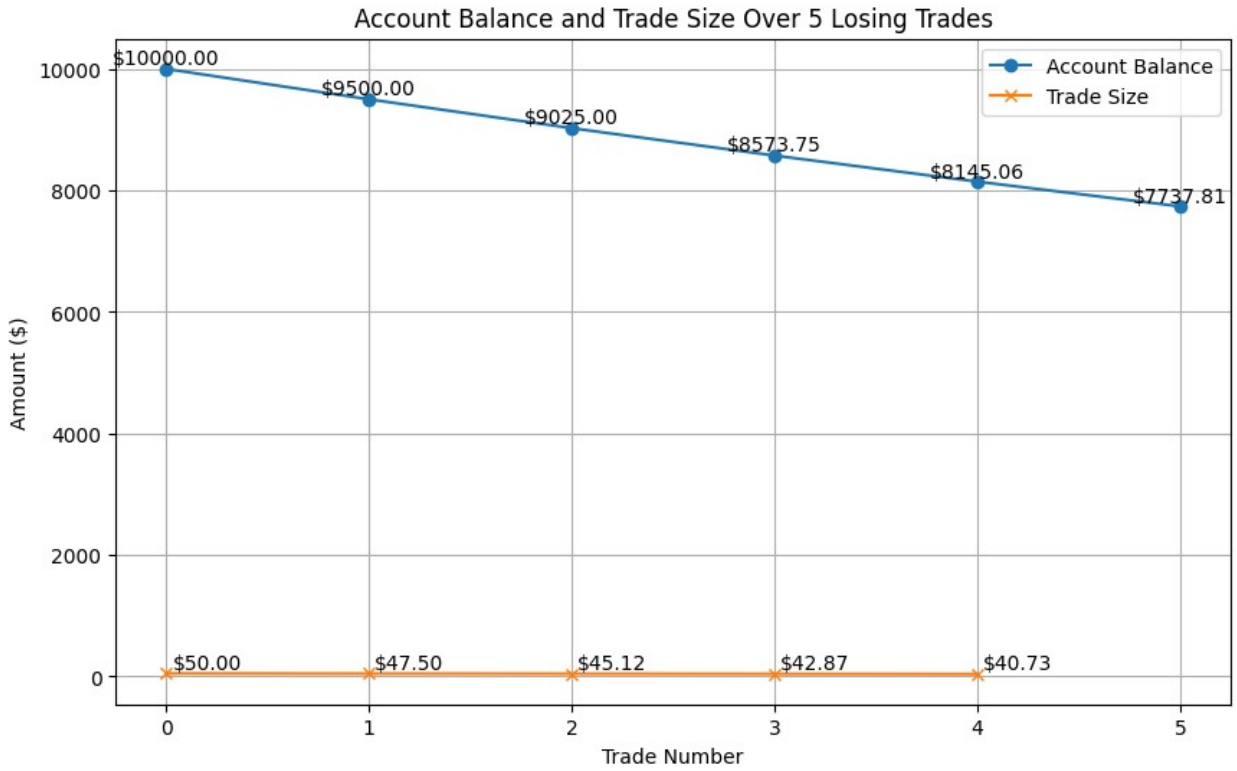

```
{"type":"string"}
```

```
# Parameters
account_balance = 10000
trade_risk_percentage = 0.005  # 0.5% per trade

# Simulate 5 trades where the account loses 5% each time
for i in range(5):
    trade_size = account_balance * trade_risk_percentage
    print(f"Account balance: ${account_balance:.2f} → Next trade size:
${trade_size:.2f}")
    account_balance *= 0.95  # Losing 5% of account per trade
```

```
Account balance: $10000.00 → Next trade size: $50.00
Account balance: $9500.00 → Next trade size: $47.50
Account balance: $9025.00 → Next trade size: $45.12
Account balance: $8573.75 → Next trade size: $42.87
Account balance: $8145.06 → Next trade size: $40.73
```

##Account Balance and Trade Size Over 5 Losing Trades

```python
import matplotlib.pyplot as plt

# Parameters
account_balance = 10000
trade_risk_percentage = 0.005  # 0.5% per trade

# Lists to store data for plotting
account_balances = [account_balance]
trade_sizes = []

# Simulate 5 trades and store data
for i in range(5):
    trade_size = account_balance * trade_risk_percentage
    trade_sizes.append(trade_size)
    account_balance *= 0.95  # Losing 5% of account per trade
    account_balances.append(account_balance)

# Create the chart
plt.figure(figsize=(10, 6))
plt.plot(account_balances, marker='o', label='Account Balance')
plt.plot(trade_sizes, marker='x', label='Trade Size')
plt.title('Account Balance and Trade Size Over 5 Losing Trades')
plt.xlabel('Trade Number')
plt.ylabel('Amount ($)')
plt.xticks(range(len(account_balances)))  # Set x-axis ticks for each
trade
plt.legend()
plt.grid(True)

# Add values to the chart
for i, txt in enumerate(account_balances):
    plt.text(i, txt, f'${txt:.2f}', ha='center', va='bottom')  #
Account balance values
for i, txt in enumerate(trade_sizes):
    plt.text(i + 0.2, txt, f'${txt:.2f}', ha='center', va='bottom')  #
Trade size values (offset slightly)

plt.show()
```



Your data set does not have a column of deposit and withdrawal Because of which the **second condition** is not working

I also want you to go over the withdrawals and deposit that went into the broker for the year so that you can better understand the traders number how in google colab python with query

#**Analysis Complete**
