Data Handling & Exploration

The dataset has 59317 rows and 14 columns.

Data Distribution:

1. Open Price and Close Price columns: Both histograms are highly right-skewed indicating the majority of the values are very small.
2. Reason Column: The histogram shows distinct peaks at specific values. The highest count appears at reason code 16, indicating that this is the most common trading reason.
3. Volume: The volume distribution is highly skewed, with most of the data concentrated near zero. A few values are very large (close to 100,000), but they are rare. This indicates that the majority of volume values are very small,
4. Profit: The profit distribution is tightly centered around zero, meaning most of the values are close to zero. This suggests that most of the records have minimal profit or loss.

Data Quality Issues:

- **Missing Values:** The dataset has no missing values.
- **Duplicate Values:** The dataset has no duplicate values.
- **Garbage Values:** The dataset has no garbage values.
- **Inconsistent Data:** The type column has inconsistent data.
- **Incorrect Data Type:** The login, ticket, symbol, type, open_time, and close_time columns have incorrect data types.
- **Structural Error:** The open_time and close_time columns have structural errors.
- **Outliers:** The open_price, close_price, stop loss, take profit, pips, volume, and profit columns have outliers.
- **Relationship between Columns:** The following columns have a strong positive correlation.
  open_price & close_price: 0.999888
  open_price & stop loss: 0.799446
  close_price & stop loss: 0.798975
  open_price & take profit: 0.710189
  close_price & take profit: 0.710091
  stop loss & take profit: 0.663627

  The profit column has neither a strong positive nor a strong negative correlation with any other columns, as all correlation coefficients are close to zero.
- **Inconsistency in close_time Column:** This column contained 2024, 2025, and 1970 years. It is not correct to have 1970 between 2024 and 2025.

Handling Strategies:

- **Handling Inconsistent Data:** The type column had Buy, Sell, buy, and sell values. Replaced all the buy values to Buy and sell values to Sell.
- **Handling Incorrect Data Type:** The symbol, type, open_time, and close_time columns were object type. The symbol and type columns were converted to string, while the

open_time and close_time columns were converted to datetime. Also converted the login and ticket columns to string since they are unique IDs.

- **Handling Structural Error:** The open_time data was originally in the format YYYY.MM.DD HH:MM:SS (e.g., 2024.07.30 11:05:29) and was converted to YYYY-MM-DD HH:MM:SS (e.g., 2024-07-30 11:05:29). The close_time column contained mixed formats (YYYY.MM.DD HH:MM:SS and YYYY-MM-DD HH:MM:SS), so the entire column was standardized to the YYYY-MM-DD HH:MM:SS format(e.g., 2024-07-30 11:05:29).
- **Handling Outliers:** The columns contained outliers. I assumed the columns represented the following:
  login – The unique identifier for a trader's account.
  ticket – The unique trade ID assigned to each transaction.
  symbol – The financial instrument being traded.
  type – The type of trade executed.
  open_time – The exact timestamp when the trade was opened.
  close_time – The exact timestamp when the trade was closed.
  open_price – The price at which the asset was bought or sold when the trade was initiated.
  close_price – The price at which the asset was bought or sold when the trade was closed.
  stop loss – The predefined price level where the trade would automatically close to minimize losses.
  take profit – The predefined price level where the trade would automatically close to secure profits.
  pips – The profit/loss measured in pips (smallest price movement in trading).
  reason – The reason for trade closure.
  volume – The lot size or contract size of the trade.
  profit – The total monetary profit/loss from the trade.

  Based on these definitions, I think values in the open_price, close_price, profit, and other columns can be both very small and very large due to the different values in the symbol column and other factors. These created outliers. They are not necessarily errors. Therefore, I kept the outliers as they were.
- **Handling Unwanted Columns:** Since all correlation coefficients for the profit column with other columns are close to zero, so, I did not remove any column.
- **Handling close_time Column Inconsistency:** If I deleted all the rows containing 1970 years, it would delete three unique logins. Since open_time and close_time should have the same date so I replaced all the 1970 year's dates with that row's open_time date. This kept all the unique logins and also removed the inconsistency in the close_time column.

Profitability Analysis

1.

```python
#1. Conduct an in-depth analysis to identify the most and least profitable logins.

most_profitable = df.loc[df["profit"].idxmax()]
least_profitable = df.loc[df["profit"].idxmin()]

print("Most Profitable Login:\n", most_profitable)
print("\nLeast Profitable Login:\n", least_profitable)
```

```
Most Profitable Login:
 login                     13387006
ticket                    75588886
symbol                      USDJPY
type                          Sell
open_time      2025-01-23 03:59:36
close_time     2025-01-24 06:17:30
open_price                 156.558
close_price                155.078
stop loss                      0.0
take profit                 155.08
pips                        1478.0
reason                           4
volume                        2000
profit                     19061.1
Name: 43333, dtype: object

Least Profitable Login:
 login                     13047591
ticket                    34731674
symbol                      XAUUSD
type                          Sell
open_time      2024-10-24 05:18:45
close_time     2024-10-24 15:29:01
open_price                 2721.73
close_price                2739.23
stop loss                  2741.75
take profit                2719.15
pips                       -1750.0
reason                          16
volume                         700
profit                    -12250.0
Name: 3247, dtype: object
```

2.

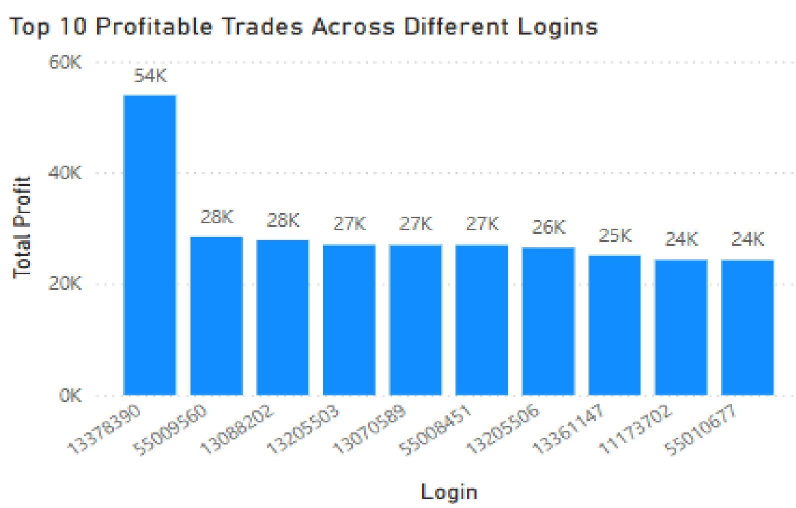
```python
#2. Compute cumulative profits per login and rank them based on profitability.
profits_per_login = df.groupby("login")["profit"].sum().reset_index()
profits_per_login["rank"] = profits_per_login["profit"].rank(ascending=False, method="dense")
profits_per_login = profits_per_login.sort_values(by="rank")
profits_per_login = profits_per_login[["rank", "login", "profit"]]

profits_per_login
```

	rank	login	profit
396	1.0	13378390	53891.98
514	2.0	55009560	28475.44
50	3.0	13088202	27848.61
146	4.0	13205503	27049.34
40	5.0	13070589	27023.68
...	...	...	...
193	596.0	13251499	-11405.24
23	597.0	13018096	-12194.31
542	598.0	55011482	-12215.00
329	599.0	13333728	-13868.00
61	600.0	13103928	-14778.82

600 rows × 3 columns

3. There were 600 unique logins, making it impractical to display all of them. That is why I displayed the top 10 profits across different logins.

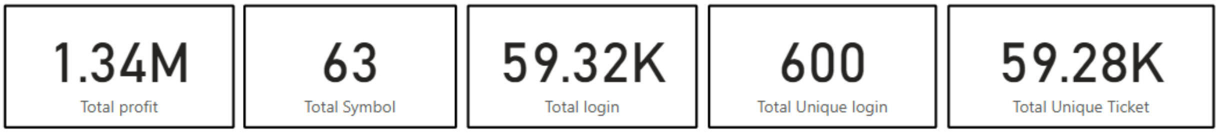


Top 10 Profitable Trades Across Logins:

- Highest: Login ID 13378390 with 54K
- Followed by Login ID 55009560 with 28K

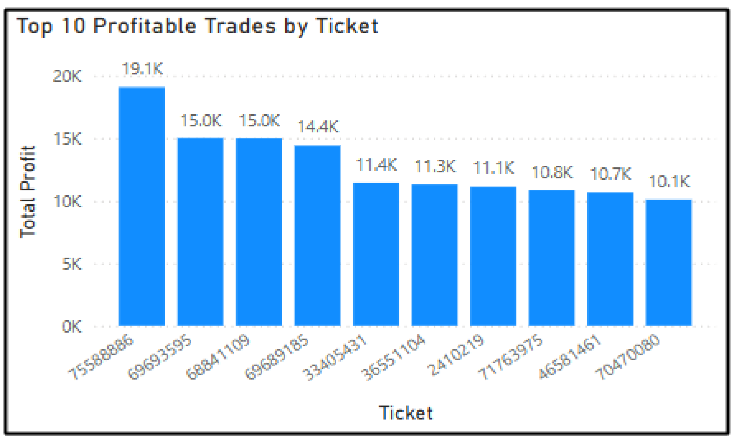
4.

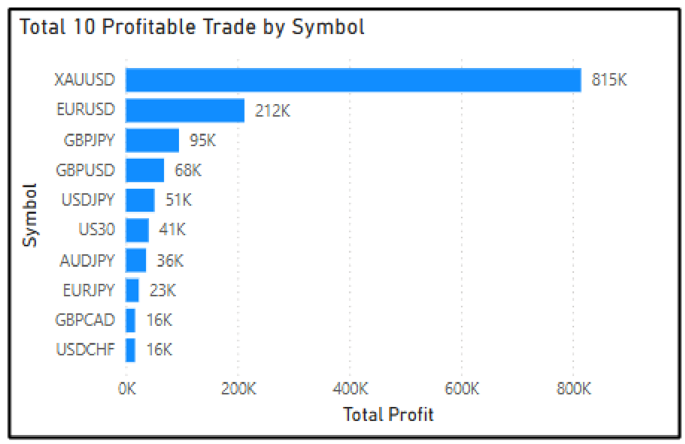
ANALYSIS



Total Profit Analysis:

- **Overall Total Profit:** 1.34M
- **Total Symbols:** 63
- **Total Login:** 59.32K
- **Total Unique Login:** 600
- **Total Unique Tickets:** 59.28K



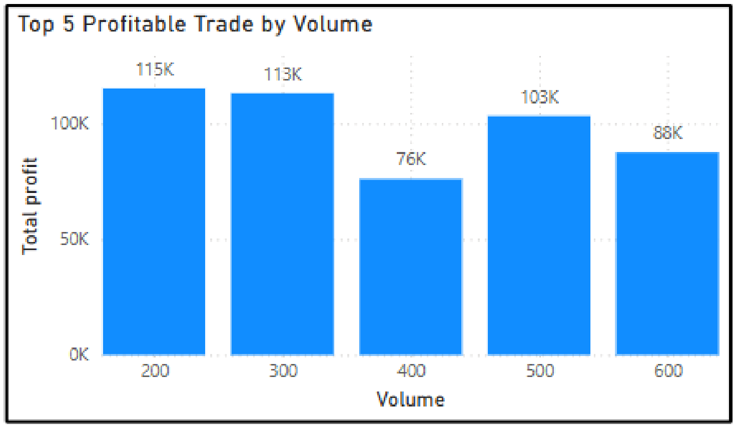


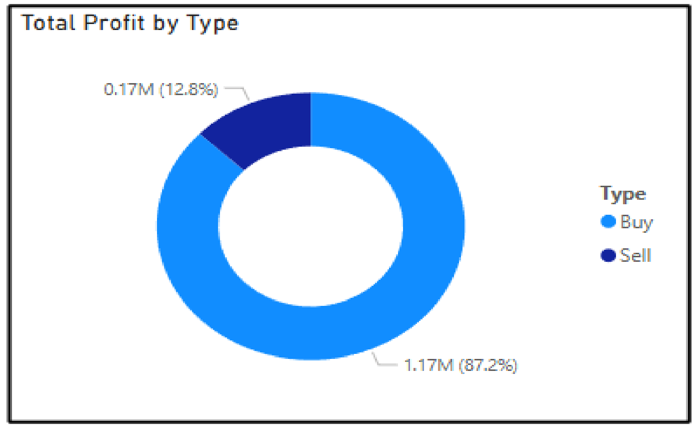
Top 10 Profitable Trades by Ticket:

- **Highest:** Ticket ID 75588886 with 19.1K
- **Second Highest:** Ticket ID 69693595 with 15K

Top 10 Profitable Trades by Symbol:

- **Highest:** XAUUSD - 815K
- **Second Highest:** EURUSD - 212K
- **Third Highest:** GBPJPY - 95K



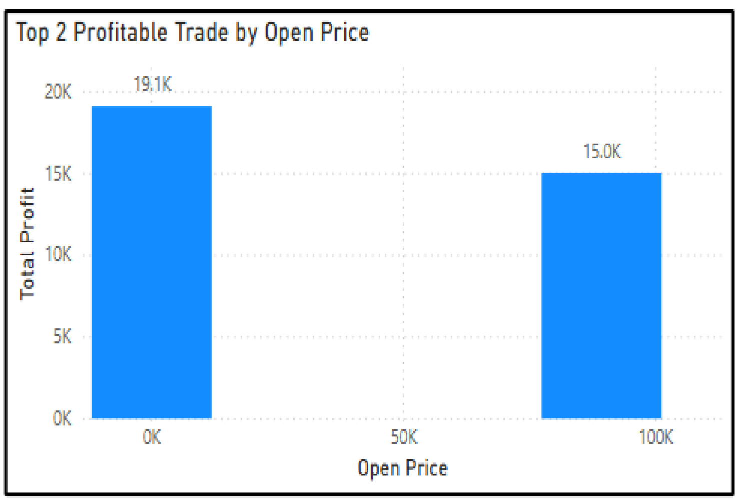


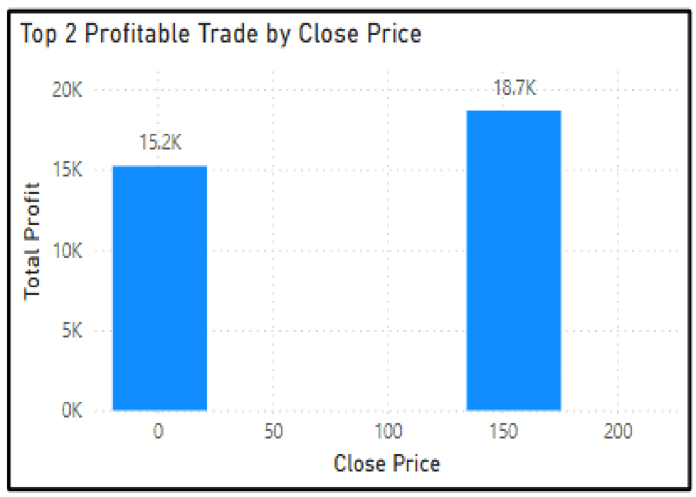
Top 5 Profitable Trades by Volume:

- **Highest:** Volume 200 - 115K
- **Second Highest:** Volume 300 - 113K

Trade Type Analysis:

- **Buy:** 1.17M (87.2%)
- **Sell:** 0.17M (12.8%)



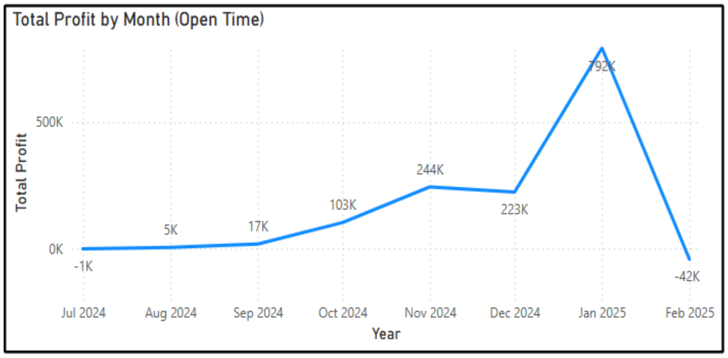


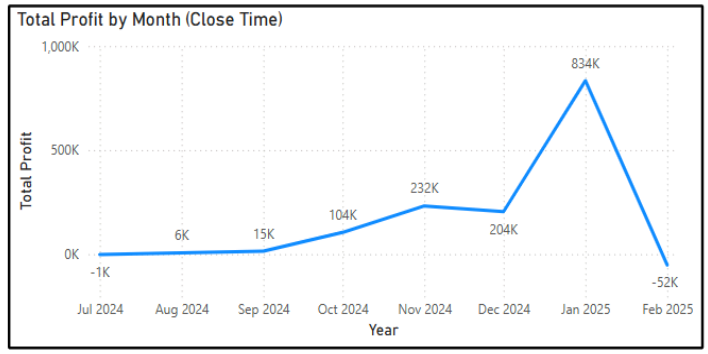
Top 2 Profitable Trades by Open Price:

- 156.56: 19.1K
- 89,400: 15K

Top 2 Profitable Trades by Close Price:

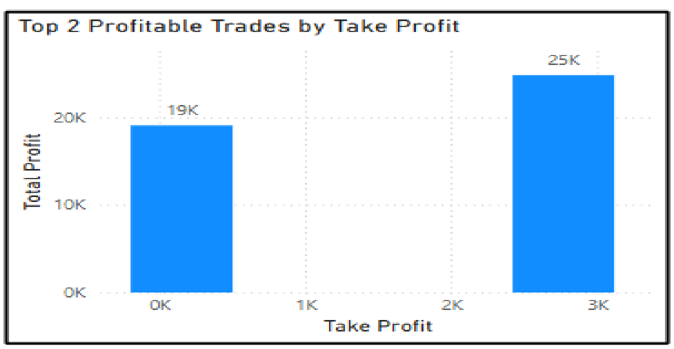
- 1.02: 15.2K
- 155.08: 18.7K





__Profit by Month:__

- **Open Time:**
  - Highest: January 2025 with 792K
  - Lowest: July 2024 with a 1K loss
- **Close Time:**
  - Highest: January 2025 with 834K
  - Lowest: July 2024 with a 1K loss
- Steady recovery and increase in profits post-March. Not counting February 2025 in the lowest since we don't have the whole month's data.

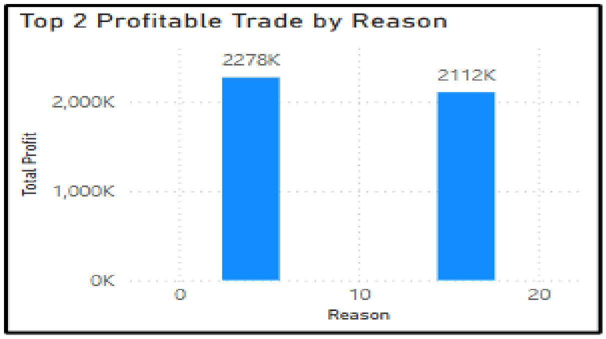


Top 2 Profitable Trades by Take Profit:

- 155.08: 19K
- 2770: 25K

Top 2 Profitable Trades by Stop Loss:

- 0.00: 418K
- 2739: 20K



Top 2 Profitable Trades by Reasons:

- 4: 2.28M
- 16: 2.11M

Key Insights:

1. **Strong Overall Profitability:**
   - Total profits exceeded $1.3M, driven by effective trading across symbols like XAUUSD and EURUSD.
2. **Significant Contribution by Volume (200 and 300) Trades:**
   - Volume 200 and Volume 300 contributed the most profits.
3. **Majority Profits from Buy Trades:**
   - Buy trades dominate, contributing over 87% of the total profits.
4. **Month-wise Analysis:**
   - January 2025 was the most profitable month.
   - July 2024 shows a significant drop, indicating potential challenges during this period.

Recommendations:

1. **Enhance Trading Strategies for Low-Profit Periods:**
   - Investigate factors causing losses in February to reduce risks in future trading cycles.
2. **Leverage High-Performing Symbols:**
   - Focus on XAUUSD and EURUSD as they generate the highest profits. Explore scaling strategies for other promising symbols like GBPJPY and GBPUSD.
3. **Optimize Buy Trades:**
   - Given the dominance of buy trades, further refine strategies to maximize profits in this type.
4. **Focus on Top Logins:**
   - Encourage top-performing logins (e.g., Login ID 13378390) to maintain or enhance profitability.

Issues

I initially tried accessing the Google Sheet file in Jupyter Notebook using the sheet link and sheet name. However, the close_time column displayed 12,149 values as NaN, even though they were present in the data file. To resolve this, I downloaded the file to my laptop and accessed it via the drive link in Jupyter Notebook, which preserved the values and prevented them from appear
ing as NaN.

```
df.isnull().sum()
```

```
login              0
ticket             0
symbol             0
type               0
open_time          0
close_time     12149
open_price         0
close_price        0
stop loss          0
take profit        0
pips               0
reason             0
volume             0
profit             0
dtype: int64
```