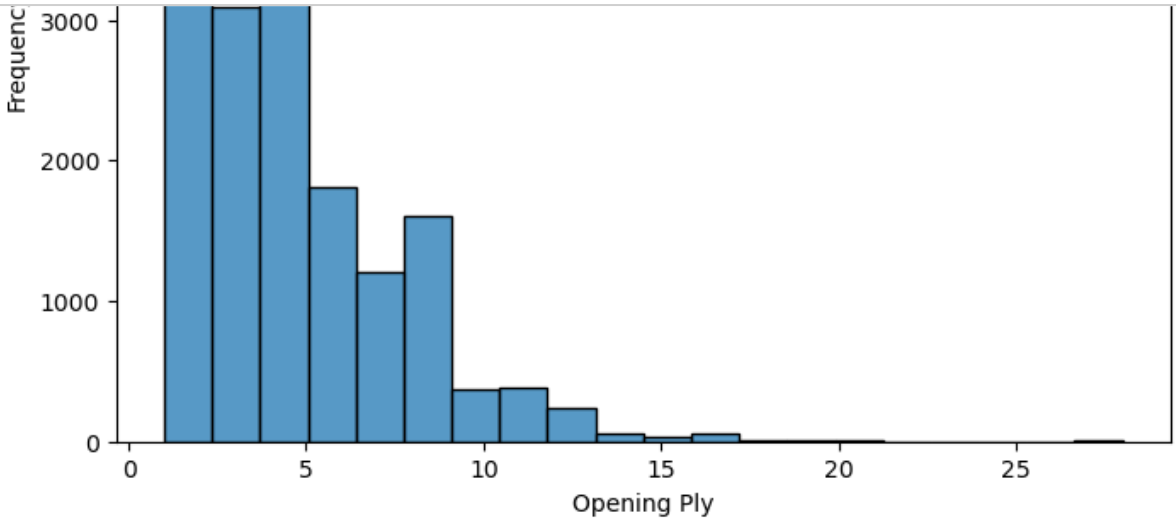
In [15]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

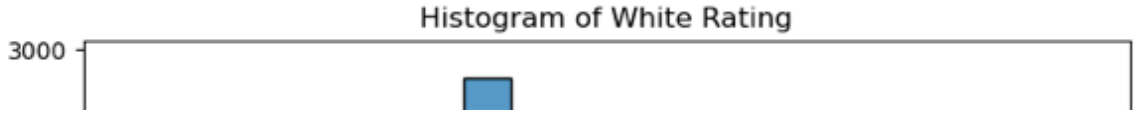
In [20]:
```
games = pd.read_csv(r"games_clean.csv")

print(games)
```

```
       Unnamed: 0   Game ID  Rated  Turns Victory Status Winner  \
0               1  l1NXvwaE   True     16         resign  black
1               2  mIICvQHh   True     61           mate  white
2               3  kWKvrqYL   True     61           mate  white
3               4  9tXo1AUZ   True     95           mate  white
4               5  MsoDV9wj  False      5           draw   draw
...           ...       ...    ...    ...            ...    ...
18373       20053  EfqH7VVH   True     24         resign  white
18374       20054  WSJDhbPl   True     82           mate  black
18375       20055  yrAas0Kj   True     35           mate  white
18376       20056  b0v4tRyF   True    109         resign  white
18377       20057  N8G2JHGG   True     78           mate  black

      Increment Code       White ID  White Rating            Black
ID  \
0               5+10           a-00          1322           skinne
rua
1               5+10         ischia          1496                a
-00
2               20+0  daniamurashov          1439        adivanov2
009
3               30+3      nik221107          1523        adivanov2
009
4               10+0      trelynn17          1250       franklin14
532
...              ...            ...           ...
...
18373          10+10        belcolt          1691            jambo
ger
18374           10+0       jamboger          1233  farrukhasomiddi
nov
18375           10+0       jamboger          1219        schaaksmu
rf3
18376           10+0   marcodisogno          1360            jambo
ger
18377           10+0       jamboger          1235              ff
bob

       Black Rating                                              M
oves  \
0              1261  d4 Nc6 e4 e5 f4 f6 dxe5 fxe5 fxe5 Nxe5 Qd4 Nc
6...
1              1500  e4 e5 d3 d6 Be3 c6 Be2 b5 Nd2 a5 a4 c5 axb5 N
```

```
c...
2             1454  d4 d5 Nf3 Bf5 Nc3 Nf6 Bf4 Ng4 e3 Nc6 Be2 Qd7
0...

3             1469  e4 e5 Nf3 d6 d4 Nc6 d5 Nb4 a3 Na6 Nc3 Be7 b4
N...
4             1002                                  e4 c5 Nf3 Qa
5 a3
...            ...
...
18373         1220  d4 f5 e3 e6 Nf3 Nf6 Nc3 b6 Be2 Bb7 O-O Be7 Ne
5...
18374         1196  d4 d6 Bf4 e5 Bg3 Nf6 e3 exd4 exd4 d5 c3 Bd6 B
d...
18375         1286  d4 d5 Bf4 Nc6 e3 Nf6 c3 e6 Nf3 Be7 Bd3 O-O Nb
d...
18376         1227  e4 d6 d4 Nf6 e5 dxe5 dxe5 Qxd1+ Kxd1 Nd5 c4 N
b...
18377         1339  d4 d5 Bf4 Na6 e3 e6 c3 Nf6 Nf3 Bd7 Nbd2 b5 Bd
3...

      Opening ECO                            Opening Name  Opening
Ply
0             B00  Nimzowitsch Defense: Kennedy Variation
4
1             C20   King's Pawn Game: Leonardis Variation
3
2             D02  Queen's Pawn Game: Zukertort Variation
3
3             C41                       Philidor Defense
5
4             B27    Sicilian Defense: Mongoose Variation
4
...           ...                                   ...
...
18373         A80                           Dutch Defense
2
18374         A41                            Queen's Pawn
2
18375         D00         Queen's Pawn Game: Mason Attack
3
18376         B07                            Pirc Defense
4
18377         D00         Queen's Pawn Game: Mason Attack
3

[18378 rows x 15 columns]
```

In [12]:
```
# Selecting the variables of interest
variables = ['Turns', 'Opening ECO', 'Opening Ply', 'White Rating',

# Plot histograms for each variable
for var in variables:
    plt.figure(figsize=(8, 6))
    sns.histplot(games[var], kde=False, bins=20)
    plt.title(f'Histogram of {var}')
    plt.xlabel(var)
    plt.ylabel('Frequency')
    plt.show()

# Identify and handle outliers
# Turns variable
q1 = games['Turns'].quantile(0.25)
q3 = games['Turns'].quantile(0.75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
outliers_turns = games[(games['Turns'] < lower_bound) | (games['Tur
print("Outliers in Turns variable:")
print(outliers_turns)

# Handle outliers by removing them
games = games[(games['Turns'] >= lower_bound) & (games['Turns'] <=
```

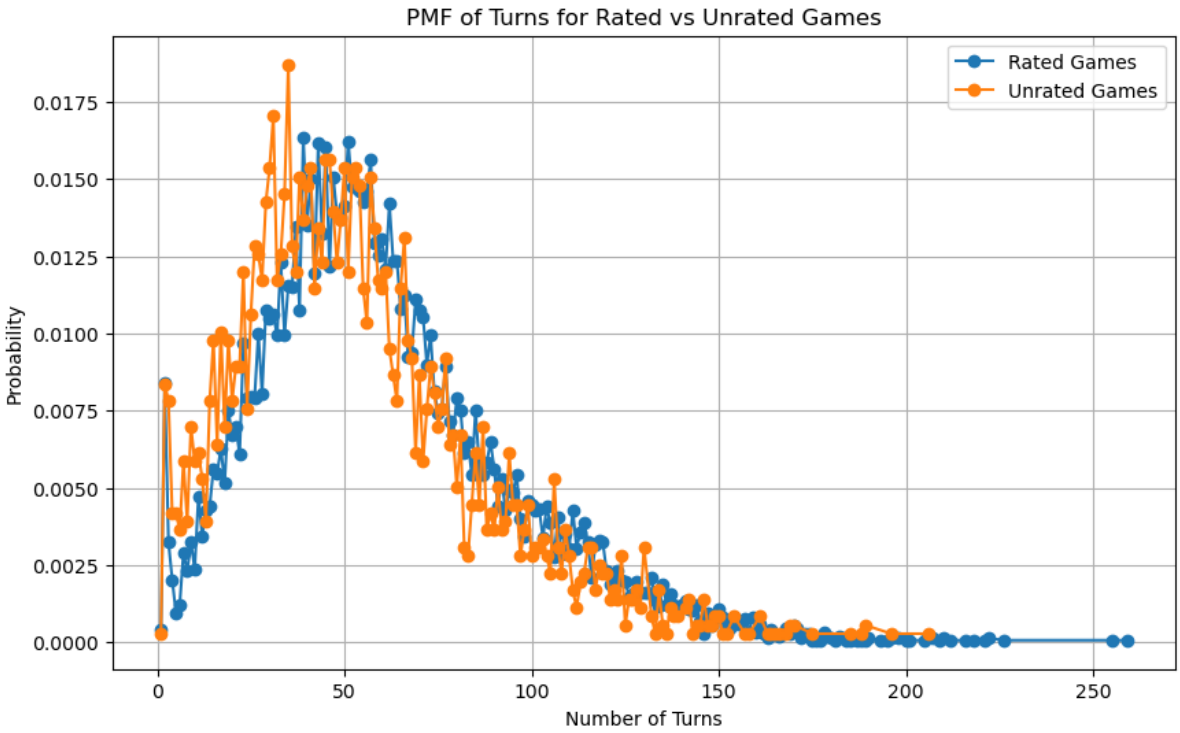




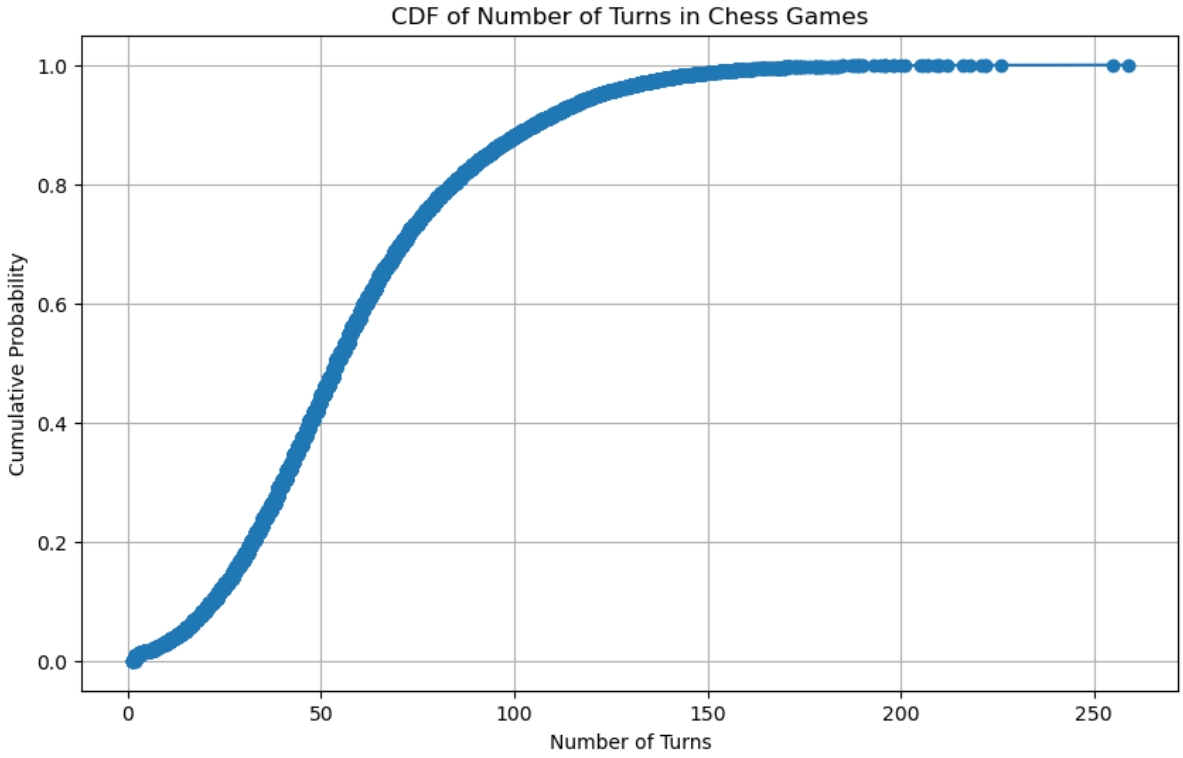
In [13]:
```
# Calculate mean for each variable
mean_turns = df['Turns'].mean()
mean_white_rating = df['White Rating'].mean()
mean_black_rating = df['Black Rating'].mean()

# Calculate mode for each variable
mode_opening_eco = df['Opening ECO'].mode()[0]  # Assuming Opening
mode_opening_ply = df['Opening Ply'].mode()[0]  # Assuming Opening

# Calculate spread (standard deviation) for each variable
std_turns = df['Turns'].std()
std_white_rating = df['White Rating'].std()
std_black_rating = df['Black Rating'].std()

# Calculate tails (skewness) for each variable
skewness_turns = df['Turns'].skew()
skewness_white_rating = df['White Rating'].skew()
skewness_black_rating = df['Black Rating'].skew()

# Print the results
print("Mean Turns:", mean_turns)
print("Mean White Rating:", mean_white_rating)
print("Mean Black Rating:", mean_black_rating)
print("Mode Opening ECO:", mode_opening_eco)
print("Mode Opening Ply:", mode_opening_ply)
print("Spread (Std) Turns:", std_turns)
print("Spread (Std) White Rating:", std_white_rating)
print("Spread (Std) Black Rating:", std_black_rating)
print("Tails (Skewness) Turns:", skewness_turns)
print("Tails (Skewness) White Rating:", skewness_white_rating)
print("Tails (Skewness) Black Rating:", skewness_black_rating)
```

```
Mean Turns: 59.34372619436282
Mean White Rating: 1595.1694961366852
Mean Black Rating: 1586.9201762977473
Mode Opening ECO: A00
Mode Opening Ply: 3
Spread (Std) Turns: 32.79123920738702
Spread (Std) White Rating: 290.281812118375
Spread (Std) Black Rating: 290.6106468642933
Tails (Skewness) Turns: 0.8965955362314061
Tails (Skewness) White Rating: 0.29543523362766727
Tails (Skewness) Black Rating: 0.26523329742734925
```

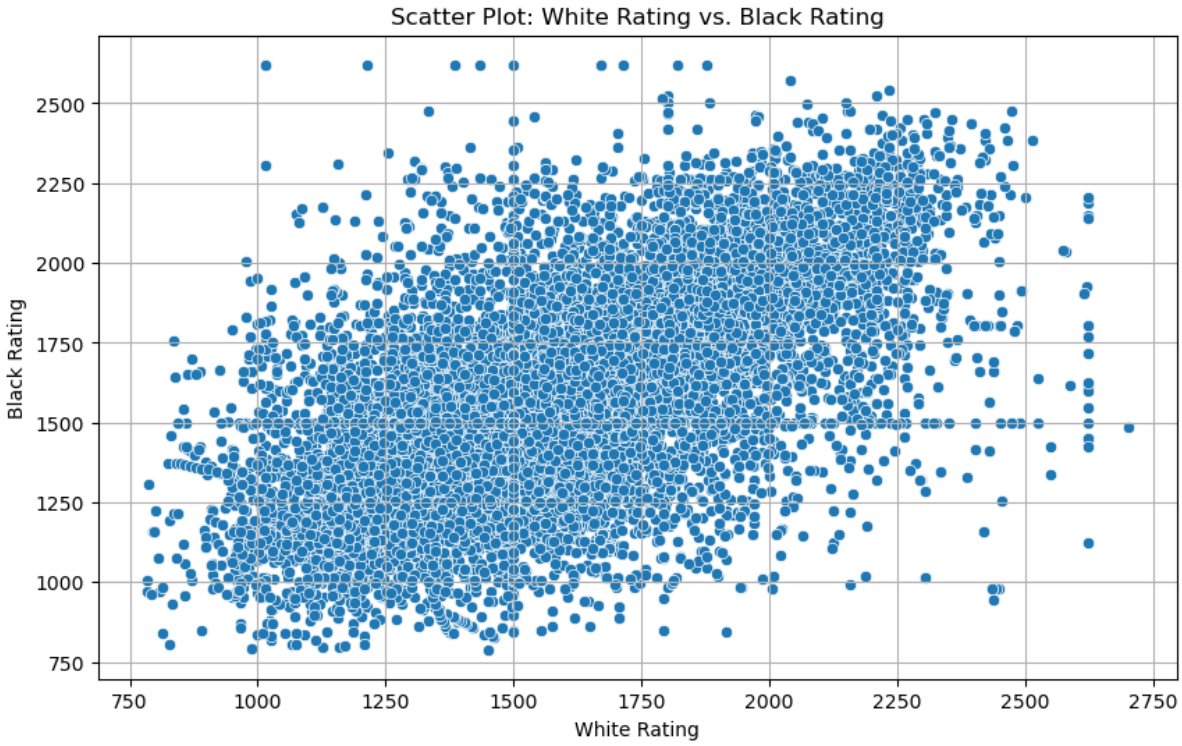
In [14]:
```
# Separate the data into two scenarios: rated and unrated games
rated_games = df[df['Rated'] == True]
unrated_games = df[df['Rated'] == False]

# Create PMFs for 'Turns' in each scenario
rated_pmf = rated_games['Turns'].value_counts(normalize=True).sort_
unrated_pmf = unrated_games['Turns'].value_counts(normalize=True).s

# Plot PMFs for comparison
plt.figure(figsize=(10, 6))
plt.plot(rated_pmf.index, rated_pmf.values, label='Rated Games', ma
plt.plot(unrated_pmf.index, unrated_pmf.values, label='Unrated Game
plt.xlabel('Number of Turns')
plt.ylabel('Probability')
plt.title('PMF of Turns for Rated vs Unrated Games')
plt.legend()
plt.grid(True)
plt.show()
```



In [16]:
```
# Calculate the CDF
turns_sorted = np.sort(df['Turns'])
cumulative_prob = np.linspace(0, 1, len(turns_sorted))

# Plot the CDF
plt.figure(figsize=(10, 6))
plt.plot(turns_sorted, cumulative_prob, marker='o', linestyle='-')
plt.xlabel('Number of Turns')
plt.ylabel('Cumulative Probability')
plt.title('CDF of Number of Turns in Chess Games')
plt.grid(True)
plt.show()
```



In [17]:
```
# Scatter plot for White Rating vs. Black Rating
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='White Rating', y='Black Rating')
plt.title('Scatter Plot: White Rating vs. Black Rating')
plt.xlabel('White Rating')
plt.ylabel('Black Rating')
plt.grid(True)
plt.show()

# Calculate covariance
covariance = np.cov(df['White Rating'], df['Black Rating'])[0, 1]
print(f'Covariance between White Rating and Black Rating: {covarian

# Calculate Pearson's correlation coefficient
correlation = np.corrcoef(df['White Rating'], df['Black Rating'])[0
print(f'Pearson\'s correlation coefficient: {correlation}')

# Assess non-linear relationship using rank correlation (Spearman's
spearman_corr = df[['White Rating', 'Black Rating']].corr(method='s
print(f'Spearman\'s correlation coefficient: {spearman_corr}')
```




```
Covariance between White Rating and Black Rating: 53197.7730125065
06
Pearson's correlation coefficient: 0.6306118179469812
Spearman's correlation coefficient: 0.6481905218167254
```

In [18]:
```
from scipy.stats import ttest_ind

# Extracting the ratings for white and black players
white_ratings = df['White Rating']
black_ratings = df['Black Rating']

# Performing two-sample t-test
t_statistic, p_value = ttest_ind(white_ratings, black_ratings)

# Printing the results
print(f'Two-sample t-test results:')
print(f'T-statistic: {t_statistic}')
print(f'P-value: {p_value}')

# Interpret the results
alpha = 0.05  # significance level
if p_value < alpha:
    print('Reject the null hypothesis. There is a significant diffe
else:
    print('Fail to reject the null hypothesis. There is no signific
```

```
Two-sample t-test results:
T-statistic: 2.7226163951832634
P-value: 0.006479775639613337
Reject the null hypothesis. There is a significant difference betw
een the mean ratings of white and black players.
```

In [19]:
```
import statsmodels.api as sm

# Define the dependent variable (y) and explanatory variables (X)
y = df['Turns']
X = df[['White Rating', 'Black Rating']]

# Add a constant term to the explanatory variables
X = sm.add_constant(X)

# Fit the multiple linear regression model
model = sm.OLS(y, X).fit()

# Print the regression results
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Turns   R-squared:
0.024
Model:                            OLS   Adj. R-squared:
0.024
Method:                 Least Squares   F-statistic:
229.5
```

```
Date:                Sat, 02 Mar 2024   Prob (F-statistic):
3.48e-99
Time:                        16:41:45   Log-Likelihood:
-89992.
No. Observations:               18378   AIC:
1.800e+05
Df Residuals:                   18375   BIC:
1.800e+05
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.0
25      0.975]
------------------------------------------------------------------------------
const          29.0410      1.469     19.767      0.000      26.1
61      31.921
White Rating     0.0052      0.001      4.879      0.000       0.0
03       0.007
Black Rating     0.0139      0.001     13.115      0.000       0.0
12       0.016
==============================================================================
Omnibus:                     2294.309   Durbin-Watson:
1.802
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
3443.387
Skew:                           0.916   Prob(JB):
0.00
Kurtosis:                       4.067   Cond. No.
1.40e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the error
s is correctly specified.
[2] The condition number is large, 1.4e+04. This might indicate th
at there are
strong multicollinearity or other numerical problems.
```

In [ ]: