Activity: Basic Exploration

Introduction

In this activity you will practice using exploration methods on a data set containing games of online chess*. This activity includes some or all of the following, not necessarily in this order:

- Viewing the data
- Finding the mean
- Finding the median
- Standard deviation
- Aggregations
- Grouping

*The data set is from Kaggle (https://www.kaggle.com/datasets/mysarahmadbhat/online-chess-games).

Note

This data set is larger than those used in previous activities. Please run the cell below which uses the `info()` method to get a sense of the data before you begin.

In [1]:
```
import pandas as pd

df = pd.read_csv('chess_games.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20058 entries, 0 to 20057
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   game_id            20058 non-null  int64 
 1   rated              20058 non-null  bool  
 2   turns              20058 non-null  int64 
 3   victory_status     20058 non-null  object
 4   winner             20058 non-null  object
 5   time_increment     20058 non-null  object
 6   white_id           20058 non-null  object
 7   white_rating       20058 non-null  int64 
 8   black_id           20058 non-null  object
 9   black_rating       20058 non-null  int64 
 10  moves              20058 non-null  object
 11  opening_code       20058 non-null  object
 12  opening_moves      20058 non-null  int64 
 13  opening_fullname   20058 non-null  object
 14  opening_shortname  20058 non-null  object
 15  opening_response   1207 non-null   object
 16  opening_variation  14398 non-null  object
dtypes: bool(1), int64(5), object(11)
memory usage: 2.5+ MB
```

Question 1

Create two `DataFrame` objects called `first_three` and `last_three` and assign the first and last three rows of the data set to them, respectively.

In [8]:
```
# Your code here

first_three = df.head(3)

last_three = df.tail(3)


df.head(2)
```

Out[8]:

	game_id	rated	turns	victory_status	winner	time_increment	white_id	white_rating	blac
0	1	False	13	Out of Time	White	15+2	bourgris	1500	
1	2	True	16	Resign	Black	5+10	a-00	1322	skinn

In [4]:
```
# Question 1 Grading Checks

assert first_three.shape == (3, 17), 'Make sure that you chose only the fir
st three rows.'
assert last_three.shape == (3, 17), 'Make sure that you chose only the last
three rows.'
```

Question 2

Create two new `DataFrame` objects called `white_lower_rating` and `white_higher_rating` that are assigned the rows of data where the white player's rating is less than 1200 and greater than or equal to 1800, respectively.

In [14]:
```
# Your code here


white_lower_rating = df[df['white_rating'] < 1200]

white_higher_rating = df[df['white_rating'] >= 1800]
```

In [15]:
```
# Question 2 Grading Checks

assert isinstance(white_lower_rating, pd.DataFrame), 'Make sure that you ar
e creating a DataFrame object called white_lower_rating.'
assert isinstance(white_higher_rating, pd.DataFrame), 'Make sure that you a
re creating a DataFrame object called white_higher_rating.'
```

Question 3

Using the `black_rating` column, create a `DataFrame` object called `top_10_percent_black` which is assigned the top 10% of black players by rating. That is, the only rows of where the `black_rating` is higher than 90% of all the `black_rating` values.

In [29]:
```
# Your code here

#df.black_rating.max()


top_10_percent_black = df[df.black_rating >= df.black_rating.quantile(0.9)]

#top_10_percent_black
```

In [30]:
```
# Question 3 Grading Checks

assert top_10_percent_black.shape == (2011, 17), 'Make sure that you are se
lecting the top 10% of black players by rating. Hint: Try using a condition
al statement to check which black_rating values are in the top 10%.'
```