# Chi-Squared Test: Jeopardy Questions

Jeopardy is a popular TV show in the US where participants answer questions to win money. It's been running for many years, and is a major force in popular culture. In this project, I will work with a dataset of Jeopardy questions to figure out some patterns in the questions that could help a person on the show win.

The dataset that I am using contains 20,000 rows from a full dataset of Jeopardy questions that is located at this link: https://www.reddit.com/r/datasets/comments/1uvd0t/200000_jeopardy_questions_in_a_json_file/

## Exploring the Data

Now, I will explore the dataset to get a idea of what it looks like.

In [42]:
```
import pandas as pd
jeopardy = pd.read_csv('jeopardy.csv')
```

In [43]:
```
jeopardy.head()
```

Out[43]:

| | Show Number | Air Date | Round | Category | Value | Question | Answer |
|---|---|---|---|---|---|---|---|
| 0 | 4680 | 2004-12-31 | Jeopardy! | HISTORY | $200 | For the last 8 years of his life, Galileo was ... | Copernicus |
| 1 | 4680 | 2004-12-31 | Jeopardy! | ESPN's TOP 10 ALL-TIME ATHLETES | $200 | No. 2: 1912 Olympian; football star at Carlisl... | Jim Thorpe |
| 2 | 4680 | 2004-12-31 | Jeopardy! | EVERYBODY TALKS ABOUT IT... | $200 | The city of Yuma in this state has a record av... | Arizona |
| 3 | 4680 | 2004-12-31 | Jeopardy! | THE COMPANY LINE | $200 | In 1963, live on "The Art Linkletter Show", th... | McDonald's |
| 4 | 4680 | 2004-12-31 | Jeopardy! | EPITAPHS & TRIBUTES | $200 | Signer of the Dec. of Indep., framer of the Co... | John Adams |

In [44]:
```
jeopardy.columns
```

Out[44]:
```
Index([' Show Number', ' Air Date', ' Round', ' Category', ' Value',
       ' Question', ' Answer'],
      dtype='object')
```

I can see that there is a space in front of each column name. I will remove the spaces and reassign the columns names back to the dataframe.

In [45]:
```
removed_space = jeopardy.rename(columns=lambda x: x.strip(), inplace=True)
jeopardy.columns[removed_space]
```

Out[45]:
```
array([' Show Number', ' Air Date', 'Round', 'Category', 'Value',
       'Question', 'Answer']), dtype=object)
```

## Normalizing Text Columns

Now, I will normalize the `Question` and `Answer` columns.

In [46]:
```
from string import punctuation
import pandas as pd

def normalize_text(text):
    # Remove punctuation
    translator = str.maketrans('', '', punctuation)
    text = text.translate(translator)

    # Lowercase
    text = text.lower()

    return text

jeopardy['Question_clean'] = jeopardy['Question'].apply(lambda x: normalize_text(x))
jeopardy['Answer_clean'] = jeopardy['Answer'].apply(lambda x: normalize_text(x))
```

In [47]:
```
jeopardy.head()
```

Out[47]:

| | Show Number | Air Date | Round | Category | Value | Question | Answer | Question_clean | Answer_clean |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 4680 | 2004-12-31 | Jeopardy! | HISTORY | $200 | For the last 8 years of his life, Galileo was ... | Copernicus | for the last 8 years of his life galileo was u... | copernicus |
| 1 | 4680 | 2004-12-31 | Jeopardy! | ESPN's TOP 10 ALL-TIME ATHLETES | $200 | No. 2: 1912 Olympian; football star at Carlisl... | Jim Thorpe | no 2 1912 olympian football star at carlisle i... | jim thorpe |
| 2 | 4680 | 2004-12-31 | Jeopardy! | EVERYBODY TALKS ABOUT IT... | $200 | The city of Yuma in this state has a record av... | Arizona | the city of yuma in this state has a record av... | arizona |
| 3 | 4680 | 2004-12-31 | Jeopardy! | THE COMPANY LINE | $200 | In 1963, live on "The Art Linkletter Show", th... | McDonald's | in 1963 live on the art linkletter show this c... | mcdonalds |
| 4 | 4680 | 2004-12-31 | Jeopardy! | EPITAPHS & TRIBUTES | $200 | Signer of the Dec. of Indep., framer of the Co... | John Adams | signer of the dec of indep framer of the const... | john adams |

## Normalizing Other Columns

Now I will normalize the `Value` column by removing the '$' symbol. Then, I will normalize the `Air Date` columns by converting it to datetime.

In [48]:
```
import pandas as pd
import re

def normalize_dollar_value(value_str):
    value_str = re.sub(r'[^\d]', '', str(value_str))  # Convert to string before applying re.sub
    try:
        value_int = int(value_str)
        return value_int
    except ValueError:
        return 0

# Apply the normalize_dollar_value function to the "Value" column
jeopardy['clean_value'] = jeopardy['Value'].apply(normalize_dollar_value)

# Print the updated dataframe
jeopardy.head()
```

Out[48]:

| | Show Number | Air Date | Round | Category | Value | Question | Answer | Question_clean | Answer_clean | clean_value |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4680 | 2004-12-31 | Jeopardy! | HISTORY | $200 | For the last 8 years of his life, Galileo was ... | Copernicus | for the last 8 years of his life galileo was u... | copernicus | 200 |
| 1 | 4680 | 2004-12-31 | Jeopardy! | ESPN's TOP 10 ALL-TIME ATHLETES | $200 | No. 2: 1912 Olympian; football star at Carlisl... | Jim Thorpe | no 2 1912 olympian football star at carlisle i... | jim thorpe | 200 |
| 2 | 4680 | 2004-12-31 | Jeopardy! | EVERYBODY TALKS ABOUT IT... | $200 | The city of Yuma in this state has a record av... | Arizona | the city of yuma in this state has a record av... | arizona | 200 |
| 3 | 4680 | 2004-12-31 | Jeopardy! | THE COMPANY LINE | $200 | In 1963, live on "The Art Linkletter Show", th... | McDonald's | in 1963 live on the art linkletter show this c... | mcdonalds | 200 |
| 4 | 4680 | 2004-12-31 | Jeopardy! | EPITAPHS & TRIBUTES | $200 | Signer of the Dec. of Indep., framer of the Co... | John Adams | signer of the dec of indep framer of the const... | john adams | 200 |

In [49]:
```
jeopardy['Air Date'] = pd.to_datetime(jeopardy['Air Date'])
jeopardy.head()
```

Out[49]:

| | Show Number | Air Date | Round | Category | Value | Question | Answer | Question_clean | Answer_clean | clean_value |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4680 | 2004-12-31 | Jeopardy! | HISTORY | $200 | For the last 8 years of his life, Galileo was ... | Copernicus | for the last 8 years of his life galileo was u... | copernicus | 200 |
| 1 | 4680 | 2004-12-31 | Jeopardy! | ESPN's TOP 10 ALL-TIME ATHLETES | $200 | No. 2: 1912 Olympian; football star at Carlisl... | Jim Thorpe | no 2 1912 olympian football star at carlisle i... | jim thorpe | 200 |
| 2 | 4680 | 2004-12-31 | Jeopardy! | EVERYBODY TALKS ABOUT IT... | $200 | The city of Yuma in this state has a record av... | Arizona | the city of yuma in this state has a record av... | arizona | 200 |
| 3 | 4680 | 2004-12-31 | Jeopardy! | THE COMPANY LINE | $200 | In 1963, live on "The Art Linkletter Show", th... | McDonald's | in 1963 live on the art linkletter show this c... | mcdonalds | 200 |
| 4 | 4680 | 2004-12-31 | Jeopardy! | EPITAPHS & TRIBUTES | $200 | Signer of the Dec. of Indep., framer of the Co... | John Adams | signer of the dec of indep framer of the const... | john adams | 200 |

## Answers in Questions

Now that the data is ready for analysis, I will start by answering the question:

- How often is the answer contained in the question?

In [50]:
```
def answer_in_question(jeopardy):
    answer_in_question_count = 0
    total_count = 0

    for i in range(len(jeopardy)):
        if jeopardy["Answer_clean"][i] in jeopardy["Question_clean"][i]:
            answer_in_question_count += 1
        total_count += 1

    return answer_in_question_count / total_count * 100

answer_in_question_percent = answer_in_question(jeopardy.copy())

print("The percentage of answers that are contained in the question is {}%".format(answer_in_question_percent))
```

```
The percentage of answers that are contained in the question is 0.7200360180009%
```

Using the function above, I was able to find out that the answer is contained in the question only .72% of the time. This low probability shows that relying on the question is not a good strategy to get questions right.

## Recycled Questions

Now, I will look into another possible strategy to get the questions right, which is to study the questions because they may be recycled.

In [51]:
```
def question_reused(jeopardy):
    question_counts = jeopardy['Question_clean'].value_counts()
    question_reused_count = (question_counts > 1).sum()
    total_count = len(jeopardy)

    return question_reused_count / total_count * 100

question_reused_percent = question_reused(jeopardy)

print("The probability that a question is recycled is {:.2f}%".format(question_reused_percent))
```

```
The probability that a question is recycled is 0.05%
```

From the code above, I can see that studying old questions in hopes that they will be recycled on the show is also not a good stragtegy to win because the questions are only recycled .05% of the time.

## Low Value vs High Value Questions

In [53]:
```
question_overlap = []
terms_used = set()

jeopardy = jeopardy.sort_values("Air Date")

for i, row in jeopardy.iterrows():
    split_question = row["Question_clean"].split(" ")
    split_question = [q for q in split_question if len(q) > 5]
    match_count = 0
    for word in split_question:
        if word in terms_used:
            match_count += 1
    for word in split_question:
        terms_used.add(word)
    if len(split_question) > 0:
        match_count /= len(split_question)
    question_overlap.append(match_count)
jeopardy["question_overlap"] = question_overlap

jeopardy["question_overlap"].mean()
```

Out[53]:
```
0.6872014306280388
```

In [54]:
```
def determine_value(row):
    value = 0
    if row['clean_value'] > 800:
        value = 1
    return value

jeopardy["high_value"] = jeopardy.apply(determine_value, axis=1)
```

In [57]:
```
def count_usage(term):
    low_count = 0
    high_count = 0
    for i, row in jeopardy.iterrows():
        if term in row['Question_clean'].split(" "):
            if row["high_value"] == 1:
                high_count += 1
            else:
                low_count += 1
    return high_count, low_count
```

In [58]:
```
from random import choice

terms_used_list = list(terms_used)
comparison_terms = [choice(terms_used_list) for _ in range(10)]

observed_expected = []

for term in comparison_terms:
    observed_expected.append(count_usage(term))

observed_expected
```

Out[58]:
```
[(0, 1),
 (0, 1),
 (2, 1),
 (0, 1),
 (1, 1),
 (2, 2),
 (0, 2),
 (0, 1),
 (0, 1),
 (0, 1)]
```

In [59]:
```
from scipy.stats import chisquare
import numpy as np

high_value_count = jeopardy[jeopardy["high_value"] == 1].shape[0]
low_value_count = jeopardy[jeopardy["high_value"] == 0].shape[0]

chi_squared = []
for obs in observed_expected:
    total = sum(obs)
    total_prop = total / jeopardy.shape[0]
    high_value_exp = total_prop * high_value_count
    low_value_exp = total_prop * low_value_count

    observed = np.array([obs[0], obs[1]])
    expected = np.array([high_value_exp, low_value_exp])
    chi_squared.append(chisquare(observed, expected))

chi_squared
```

Out[59]:
```
[Power_divergenceResult(statistic=0.4019628461268844, pvalue=0.5260772985705469),
 Power_divergenceResult(statistic=0.4019628461268844, pvalue=0.5260772985705469),
 Power_divergenceResult(statistic=2.1177104393031944, pvalue=0.14560406860264344),
 Power_divergenceResult(statistic=0.4019628461268844, pvalue=0.5260772985705469),
 Power_divergenceResult(statistic=0.4448774816612795, pvalue=0.5047776487545996),
 Power_divergenceResult(statistic=0.8897549632322559, pvalue=0.3455437191483468),
 Power_divergenceResult(statistic=0.8039256922953768, pvalue=0.3699222378079571),
 Power_divergenceResult(statistic=0.4019628461268844, pvalue=0.5260772985705469),
 Power_divergenceResult(statistic=0.4019628461268844, pvalue=0.5260772985705469),
 Power_divergenceResult(statistic=0.4019628461268844, pvalue=0.5260772985705469)]
```

Upon analyzing the results of the Chi Square Test, it appears that many of the terms have relatively high p-values around 0.5. This suggests that the observed differences in the occurrence of these terms in high-value and low-value questions could be attributed to random variability rather than a meaningful association.

In summary, based on the current outcomes, the analysis does not find strong evidence to support a significant association between the terms and question values. Using the data, I am unable to find any relationships that can increase the chances of winning.