In [80]:
```python
import pandas as pd
import numpy as np
path="C:\\Users\\Uroš\\Desktop\\Python Jupyter\\jeopardy.csv"
jeopardy_data=pd.read_csv(path)
pd.set_option('display.max_colwidth', None)
jeopardy_data.tail()
```

Out[80]:

	Show Number	Air Date	Round	Category	Value	Question	Answer
216925	4999	2006-05-11	Double Jeopardy!	RIDDLE ME THIS	$2000	This Puccini opera turns on the solution to 3 riddles posed by the heroine	Turandot
216926	4999	2006-05-11	Double Jeopardy!	"T" BIRDS	$2000	In North America this term is properly applied to only 4 species that are crested, including the tufted	a titmouse
216927	4999	2006-05-11	Double Jeopardy!	AUTHORS IN THEIR YOUTH	$2000	In Penny Lane, where this "Hellraiser" grew up, the barber shaves another customer--then flays him alive!	Clive Barker
216928	4999	2006-05-11	Double Jeopardy!	QUOTATIONS	$2000	From Ft. Sill, Okla. he made the plea, Arizona is my land, my home, my father's land, to which I now ask to... return"	Geronimo
216929	4999	2006-05-11	Final Jeopardy!	HISTORIC NAMES	None	A silent movie title includes the last name of this 18th c. statesman & favorite of Catherine the Great	Grigori Alexandrovich Potemkin

In [81]:
```python
jeopardy_data.info()
jeopardy_data.columns
#some column names have extra space in their name
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 216930 entries, 0 to 216929
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype 
---  ------       --------------   ----- 
 0   Show Number  216930 non-null  int64 
 1    Air Date    216930 non-null  object
 2    Round       216930 non-null  object
 3    Category    216930 non-null  object
 4    Value       216930 non-null  object
 5    Question    216930 non-null  object
 6    Answer      216928 non-null  object
dtypes: int64(1), object(6)
memory usage: 6.6+ MB
```

Out[81]:
```
Index(['Show Number', ' Air Date', ' Round', ' Category', ' Value',
       ' Question', ' Answer'],
      dtype='object')
```

In [82]:
```python
#renaming columns to remove extra space
jeopardy_data.columns=["Show Number","Air Date","Round","Category","Value","Question","Answer"]
jeopardy_data.columns
```

Out[82]:
```
Index(['Show Number', 'Air Date', 'Round', 'Category', 'Value', 'Question',
       'Answer'],
      dtype='object')
```

In [83]:
```python
#searching through column "Question" for string patern
d=jeopardy_data[(jeopardy_data["Question"].str.contains("[Kk]ing", regex=True)&\
              (jeopardy_data["Question"].str.contains("[Ee]ngland", regex=True)))][["Question","Answer"]]
d
```

Out[83]:

	Question	Answer
4953	Both England's King George V & FDR put their stamp of approval on this "King of Hobbies"	Philately (stamp collecting)
6337	In retaliation for Viking raids, this "Unready" king of England attacks Norse areas of the Isle of Man	Ethelred
9191	This king of England beat the odds to trounce the French in the 1415 Battle of Agincourt	Henry V
11710	This Scotsman, the first Stuart king of England, was called "The Wisest Fool in Christendom"	James I
13454	It's the number that followed the last king of England named William	4
...	...	...
208295	In 1066 this great-great grandson of Rollo made what some call the last Viking invasion of England	William the Conqueror
208742	Dutch-born king who ruled England jointly with Mary II & is a tasty New Zealand fish	William of Orange roughy
213870	In 1781 William Herschel discovered Uranus & initially named it after this king of England	George III
216021	His nickname was "Bertie", but he used this name & number when he became king of England in 1901	Edward VII
216789	This kingdom of England grew from 2 settlements, one founded around 495 by Cerdic & his son Cynric	Wessex

152 rows × 2 columns

In [84]:
```python
#search patern for "King","King's","king" and "king's"
#Added space at the end and begging of the word so it will only search for a string not substring
jeopardy_data[jeopardy_data["Question"].str.contains(" [Kk]ing\'?s? ", regex=True)]
```

Out[84]:

	Show Number	Air Date	Round	Category	Value	Question	Answer
40	4680	2004-12-31	Double Jeopardy!	DR. SEUSS AT THE MULTIPLEX	$1200	Ripped from today's headlines, he was a turtle king gone mad; Mack was the one good turtle who'd bring him down	Yertle
781	4335	2003-06-06	Jeopardy!	MY PLACE?	$200	A Norman could say, "I'm the king of the motte-and-bailey style of" this	castle
811	4335	2003-06-06	Double Jeopardy!	"S"-OTERICA	$400	Examples of this TV format include "Leave It to Beaver" & "The King of Queens"	sitcom
896	3834	2001-04-12	Jeopardy!	AIN'T THAT AMERICA	$200	This state was named for a man who was a European king from 1643 to 1715	Louisiana
1074	4085	2002-05-10	Jeopardy!	CENTRAL PARK	$400	Central Park has a statue of King Wladyslaw II Jagiello of this country, who was also Grand Duke of Lithuania	Poland
...	...	...	...	...	...	...	...
216572	2046	1993-06-28	Jeopardy!	TRAVEL U.S.A.	$100	You have to stand up to ride Shockwave, this type of thrill ride at Kings Dominion in Virginia	a roller coaster
216675	3940	2001-10-19	Double Jeopardy!	"CAB"	$600	Stephen King's 1980 overview of the horror genre was called "Danse" this	Macabre
216744	6044	2010-12-16	Double Jeopardy!	YOUR MOMMA!	$2000	French kings Francis II, Charles IX & Henry III	Catherine de Medici
216752	5070	2006-09-29	Jeopardy!	SIGNING OFF	$200	Upon signing his name in 1776 he said, "There, King George will be able to read that without his spectacles"	(John) Hancock
216777	5070	2006-09-29	Double Jeopardy!	ANCIENT HISTORY	$400	The first one of these tombs was built about 2650 B.C. by Imhotep for King Zoser & rose about 200 feet using steps	a pyramid (the pyramids accepted)

2053 rows × 7 columns

In [85]:
```python
#inspecting column "Value"
#these are string type so if we want to convert them to float we need
#first to remove "$";"," as well to handle None values
jeopardy_data["Value"].unique()
```

Out[85]:
```
array(['$200', '$400', '$600', '$800', '$2,000', '$1000', '$1200',
       '$1600', '$2000', '$3,200', 'None', '$5,000', '$100', '$300',
       '$500', '$1,000', '$1,500', '$1,200', '$4,800', '$1,800', '$1,100',
       '$2,200', '$3,400', '$3,000', '$4,000', '$1,600', '$6,800',
       '$1,900', '$3,100', '$700', '$1,400', '$2,800', '$8,000', '$6,000',
       '$2,400', '$12,000', '$3,800', '$2,500', '$6,200', '$10,000',
       '$7,000', '$1,492', '$7,400', '$1,300', '$7,200', '$2,600',
       '$3,300', '$5,400', '$4,500', '$2,100', '$900', '$3,600', '$2,127',
       '$367', '$4,400', '$3,500', '$2,900', '$3,900', '$4,100', '$4,600',
       '$10,800', '$2,300', '$5,600', '$1,111', '$8,200', '$5,800',
       '$750', '$7,500', '$1,700', '$9,000', '$6,100', '$1,020', '$4,700',
       '$2,021', '$5,200', '$3,389', '$4,200', '$5', '$2,001', '$1,263',
       '$4,637', '$3,201', '$6,600', '$3,700', '$2,990', '$5,500',
       '$14,000', '$2,700', '$6,400', '$350', '$8,600', '$6,300', '$250',
       '$3,989', '$8,917', '$9,500', '$1,246', '$6,435', '$8,800',
       '$2,222', '$2,746', '$10,400', '$7,600', '$6,700', '$5,100',
       '$13,200', '$4,300', '$1,407', '$12,400', '$5,401', '$7,800',
       '$1,183', '$1,203', '$13,000', '$11,600', '$14,200', '$1,809',
       '$8,400', '$8,700', '$11,000', '$5,201', '$1,801', '$3,499',
       '$5,700', '$601', '$4,008', '$50', '$2,344', '$2,811', '$18,000',
       '$1,777', '$3,599', '$9,800', '$796', '$3,150', '$20', '$1,810',
       '$22', '$9,200', '$1,512', '$8,500', '$585', '$1,534', '$13,800',
       '$5,001', '$4,238', '$16,400', '$1,347', '$2547', '$11,200'],
      dtype=object)
```

In [86]:
```python
#1st I'll remove "$" and ","
mylambda=lambda x: x.replace("$","").replace(",","")
jeopardy_data["Value_float"]=jeopardy_data["Value"].apply(mylambda)
jeopardy_data["Value_float"].unique()
```

Out[86]:
```
array(['200', '400', '600', '800', '2000', '1000', '1200', '1600', '3200',
       'None', '5000', '100', '300', '500', '1500', '4800', '1800',
       '1100', '2200', '3400', '3000', '4000', '6800', '1900', '3100',
       '700', '1400', '2800', '8000', '6000', '2400', '12000', '3800',
       '2500', '6200', '10000', '7000', '1492', '7400', '1300', '7200',
       '2600', '3300', '5400', '4500', '2100', '900', '3600', '2127',
       '367', '4400', '3500', '2900', '3900', '4100', '4600', '10800',
       '2300', '5600', '1111', '8200', '5800', '750', '7500', '1700',
       '9000', '6100', '1020', '4700', '2021', '5200', '3389', '4200',
       '5', '2001', '1263', '4637', '3201', '6600', '3700', '2990',
       '5500', '14000', '2700', '6400', '350', '8600', '6300', '250',
       '3989', '8917', '9500', '1246', '6435', '8800', '2222', '2746',
       '10400', '7600', '6700', '5100', '13200', '4300', '1407', '12400',
       '5401', '7800', '1183', '1203', '13000', '11600', '14200', '1809',
       '8400', '8700', '11000', '5201', '1801', '3499', '5700', '601',
       '4008', '50', '2344', '2811', '18000', '1777', '3599', '9800',
       '796', '3150', '20', '1810', '22', '9200', '1512', '8500', '585',
       '1534', '13800', '5001', '4238', '16400', '1347', '2547', '11200'],
      dtype=object)
```

In [87]:
```python
#after having problems with some functions I realised that None values in the column is actually a string type!
jeopardy_data[jeopardy_data["Value_float"]=="None"]
```

Out[87]:

	Show Number	Air Date	Round	Category	Value	Question	Answer	Value_float
55	4680	2004-12-31	Final Jeopardy!	THE SOLAR SYSTEM	None	Objects that pass closer to the sun than Mercury have been named for this mythological figure	Icarus	None
116	5957	2010-07-06	Final Jeopardy!	HISTORIC WOMEN	None	She was born in Virginia around 1596 & died in Kent, England in 1617	Pocahontas	None
174	3751	2000-12-18	Final Jeopardy!	SPORTS LEGENDS	None	If Joe DiMaggio's hitting streak had gone one more game in 1941, this company would have given him a $10,000 contract	H.J. Heinz (Heinz 57 Varieties)	None
235	3673	2000-07-19	Final Jeopardy!	THE MAP OF EUROPE	None	Bordering Italy, Austria, Hungary & Croatia, it's one of the world's newest independent countries	Slovenia	None
296	4931	2006-02-06	Final Jeopardy!	FAMOUS SHIPS	None	On December 27, 1831 it departed Plymouth, England to map the coastline of South America	the HMS Beagle	None
...	...	...	...	...	...	...	...	...
216686	3940	2001-10-19	Final Jeopardy!	MAJOR LEAGUE BASEBALL TEAM NAMES	None	This team received its name after an 1890 incident in which it "stole" away an important player from another team	Pittsburgh Pirates	None
216746	6044	2010-12-16	Final Jeopardy!	SKYSCRAPERS	None	After a construction boom fueled by oil & gas money, this capital city now has Europe's tallest building	Moscow	None
216807	5070	2006-09-29	Final Jeopardy!	NATIONAL CAPITALS	None	This city's website calls it "the last divided capital in Europe"	Nicosia	None
216868	5195	2007-03-23	Final Jeopardy!	BESTSELLING AUTHORS	None	He had the year's bestselling novel a record 7 years in a row with 7 different titles, ending in 2000	John Grisham	None
216929	4999	2006-05-11	Final Jeopardy!	HISTORIC NAMES	None	A silent movie title includes the last name of this 18th c. statesman & favorite of Catherine the Great	Grigori Alexandrovich Potemkin	None

3634 rows × 8 columns

In [88]:
```python
#handling "None" values
jeopardy_data["Value_float"].replace({"None":np.nan}, inplace=True)
jeopardy_data["Value_float"].unique()
```

Out[88]:
```
array(['200', '400', '600', '800', '2000', '1000', '1200', '1600', '3200',
       nan, '5000', '100', '300', '500', '1500', '4800', '1800', '1100',
       '2200', '3400', '3000', '4000', '6800', '1900', '3100', '700',
       '1400', '2800', '8000', '6000', '2400', '12000', '3800', '2500',
       '6200', '10000', '7000', '1492', '7400', '1300', '7200', '2600',
       '3300', '5400', '4500', '2100', '900', '3600', '2127', '367',
       '4400', '3500', '2900', '3900', '4100', '4600', '10800', '2300',
       '5600', '1111', '8200', '5800', '750', '7500', '1700', '9000',
       '6100', '1020', '4700', '2021', '5200', '3389', '4200', '5',
       '2001', '1263', '4637', '3201', '6600', '3700', '2990', '5500',
       '14000', '2700', '6400', '350', '8600', '6300', '250', '3989',
       '8917', '9500', '1246', '6435', '8800', '2222', '2746', '10400',
       '7600', '6700', '5100', '13200', '4300', '1407', '12400', '5401',
       '7800', '1183', '1203', '13000', '11600', '14200', '1809', '8400',
       '8700', '11000', '5201', '1801', '3499', '5700', '601', '4008',
       '50', '2344', '2811', '18000', '1777', '3599', '9800', '796',
       '3150', '20', '1810', '22', '9200', '1512', '8500', '585', '1534',
       '13800', '5001', '4238', '16400', '1347', '2547', '11200'],
      dtype=object)
```

In [89]:
```python
#converting to float
jeopardy_data["Value_float"]= jeopardy_data["Value_float"].astype(float)
```

In [90]:
```python
#taking value $ as question's difficulty
#We can calculate difficulty of some topics, in this case, questions that contains "King"
king=jeopardy_data[jeopardy_data["Question"].str.contains(" [Kk]ing\'?s? ")]
king["Value_float"].mean
```

Out[90]:
```
<bound method Series.mean of 40        1200.0
781        200.0
811        400.0
896        200.0
1074       400.0
           ...  
216572     100.0
216675     600.0
216744    2000.0
216752     200.0
216777     400.0
Name: Value_float, Length: 2053, dtype: float64>
```

In [91]:
```python
#Most common answer to questions that contains "King"
king=jeopardy_data[jeopardy_data["Question"].str.contains(" [Kk]ing\'?s? ")]
king["Answer"].value_counts()
```

Out[91]:
```
Henry VIII        41
Solomon           24
Richard III       21
David             20
Norway            19
                  ..
Edward (VIII)      1
Love Me Tender     1
a whale            1
Tony Randall       1
Limerick           1
Name: Answer, Length: 1178, dtype: int64
```

In [92]:
```python
#filtering the word "computer"
computer=jeopardy_data[jeopardy_data["Question"].str.contains("[Cc]omputer\'?s? ", regex=True)]
computer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 332 entries, 342 to 216533
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Show Number  332 non-null    int64  
 1   Air Date     332 non-null    object 
 2   Round        332 non-null    object 
 3   Category     332 non-null    object 
 4   Value        332 non-null    object 
 5   Question     332 non-null    object 
 6   Answer       332 non-null    object 
 7   Value_float  329 non-null    float64
dtypes: float64(1), int64(1), object(6)
memory usage: 15.6+ KB
```

In [93]:
```python
#there's 332 entries for questions containing word "computer"
#sorting values by year
computer.sort_values(by="Air Date")
```

Out[93]:

	Show Number	Air Date	Round	Category	Value	Question	Answer	Value_float
167950	68	1984-12-12	Jeopardy!	INVENTIONS	$300	1899 device for attaching papers that can ruin a computer disc when magnetized	a paper clip	300.0
112957	484	1986-10-16	Double Jeopardy!	WW III	$400	Child's game which convinced computer in "War Games" of the futility of nuclear war	tic tac toe	400.0
58098	739	1987-11-19	Double Jeopardy!	ODD ALPHABETS	$600	In ASCII, this kind of computer code, A is 01000001	Binary Code	600.0
177634	965	1988-11-11	Jeopardy!	SEWING	$500	It's fabric that gives body & shape to a collar, not two computers talking	interfacing	500.0
189977	1002	1989-01-03	Jeopardy!	GAMES	$300	A computer version of this, the world's most popular board game, cuts playing time by ⅔	Monopoly	300.0
...	...	...	...	...	...	...	...	...
133791	6266	2011-12-12	Double Jeopardy!	THE RAND CORPORATION	$1200	In the first book on this, computers doing tasks associated with human intellect, 6 chapters had been Rand reports	Artificial Intelligence	1200.0
117246	6289	2012-01-12	Jeopardy!	SPELL IT: -ABLE OR -IBLE?	$600	Easily carried, such as a small TV or software that can run on multiple computers with the same operating system	P-O-R-T-A-B-L-E	600.0
145140	6290	2012-01-13	Jeopardy!	MAKING CONNECTIONS	$1000	In "Sixteen Candles" the geek predicts, "Me and her will" do this, the connection of a computer & an outside entity	interface	1000.0
168665	6298	2012-01-25	Jeopardy!	20th CENTURY WORDS & PHRASES	$1000	It follows "sex" to mean a certain urge; once the computer age hit, it also followed "hard"	drive	1000.0
168647	6298	2012-01-25	Jeopardy!	20th CENTURY WORDS & PHRASES	$400	This 2-word term for the idea that a computer could think like a human was coined in the 1950s	artificial intelligence	400.0

332 rows × 8 columns

In [94]:
```python
#inspecting did questions containing the word computer increase through years
#I'll make groups as follows: 80's, 90's and 2000's
computer_80s=computer[computer["Air Date"].str.contains("198\d")]
computer_80s.info()
computer_80s.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5 entries, 58098 to 189977
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Show Number  5 non-null      int64  
 1   Air Date     5 non-null      object 
 2   Round        5 non-null      object 
 3   Category     5 non-null      object 
 4   Value        5 non-null      object 
 5   Question     5 non-null      object 
 6   Answer       5 non-null      object 
 7   Value_float  5 non-null      float64
dtypes: float64(1), int64(1), object(6)
memory usage: 240.0+ bytes
```

Out[94]:

	Show Number	Air Date	Round	Category	Value	Question	Answer	Value_float
58098	739	1987-11-19	Double Jeopardy!	ODD ALPHABETS	$600	In ASCII, this kind of computer code, A is 01000001	Binary Code	600.0
112957	484	1986-10-16	Double Jeopardy!	WW III	$400	Child's game which convinced computer in "War Games" of the futility of nuclear war	tic tac toe	400.0
167950	68	1984-12-12	Jeopardy!	INVENTIONS	$300	1899 device for attaching papers that can ruin a computer disc when magnetized	a paper clip	300.0
177634	965	1988-11-11	Jeopardy!	SEWING	$500	It's fabric that gives body & shape to a collar, not two computers talking	interfacing	500.0
189977	1002	1989-01-03	Jeopardy!	GAMES	$300	A computer version of this, the world's most popular board game, cuts playing time by ⅔	Monopoly	300.0

In [95]:
```python
computer_90s=computer[computer["Air Date"].str.contains("199\d")]
computer_90s.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 70 entries, 5077 to 216533
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Show Number  70 non-null     int64  
 1   Air Date     70 non-null     object 
 2   Round        70 non-null     object 
 3   Category     70 non-null     object 
 4   Value        70 non-null     object 
 5   Question     70 non-null     object 
 6   Answer       70 non-null     object 
 7   Value_float  69 non-null     float64
dtypes: float64(1), int64(1), object(6)
memory usage: 3.3+ KB
```

In [96]:
```python
computer_2000s=computer[computer["Air Date"].str.contains("200\d")]
computer_2000s.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 206 entries, 342 to 216299
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Show Number  206 non-null    int64  
 1   Air Date     206 non-null    object 
 2   Round        206 non-null    object 
 3   Category     206 non-null    object 
 4   Value        206 non-null    object 
 5   Question     206 non-null    object 
 6   Answer       206 non-null    object 
 7   Value_float  204 non-null    float64
dtypes: float64(1), int64(1), object(6)
memory usage: 9.7+ KB
```

In [107…
```python
#let's see did questions difficulty change as well
computer_80s["Value_float"].describe()
```

Out[107…
```
count      5.000000
mean     420.000000
std      130.384048
min      300.000000
25%      300.000000
50%      400.000000
75%      500.000000
max      600.000000
Name: Value_float, dtype: float64
```

In [106…
```python
computer_90s["Value_float"].describe()
```

Out[106…
```
count      69.000000
mean      462.318841
std       255.585515
min       100.000000
25%       200.000000
50%       400.000000
75%       600.000000
max      1000.000000
Name: Value_float, dtype: float64
```

In [105…
```python
computer_2000s["Value_float"].describe()
```

Out[105…
```
count     204.00000
mean      838.72549
std       595.73223
min       100.00000
25%       400.00000
50%       800.00000
75%      1000.00000
max      4000.00000
Name: Value_float, dtype: float64
```

In [112…
```python
#perhapse mean is not the best measurement of difficulty in this case because there is difference in
#samples (numbers of questions through years)
#let's group questions by difficulties adn plot them
computer_80s_group=computer_80s.groupby("Value_float").count()
computer_80s_group
```

Out[112…

Value_float	Show Number	Air Date	Round	Category	Value	Question	Answer
300.0	2	2	2	2	2	2	2
400.0	1	1	1	1	1	1	1
500.0	1	1	1	1	1	1	1
600.0	1	1	1	1	1	1	1

In [114…
```python
computer_90s_group=computer_90s.groupby("Value_float").count()
computer_90s_group
```

Out[114…

Value_float	Show Number	Air Date	Round	Category	Value	Question	Answer
100.0	2	2	2	2	2	2	2
200.0	18	18	18	18	18	18	18
300.0	7	7	7	7	7	7	7
400.0	10	10	10	10	10	10	10
500.0	10	10	10	10	10	10	10
600.0	9	9	9	9	9	9	9
800.0	7	7	7	7	7	7	7
1000.0	6	6	6	6	6	6	6

In [115…
```python
computer_2000s_group=computer_2000s.groupby("Value_float").count()
computer_2000s_group
```

Out[115…

Value_float	Show Number	Air Date	Round	Category	Value	Question	Answer
100.0	2	2	2	2	2	2	2
200.0	24	24	24	24	24	24	24
300.0	1	1	1	1	1	1	1
400.0	41	41	41	41	41	41	41
500.0	4	4	4	4	4	4	4
600.0	28	28	28	28	28	28	28
800.0	34	34	34	34	34	34	34
1000.0	23	23	23	23	23	23	23
1200.0	16	16	16	16	16	16	16
1600.0	13	13	13	13	13	13	13
2000.0	14	14	14	14	14	14	14
2400.0	1	1	1	1	1	1	1
3000.0	2	2	2	2	2	2	2
4000.0	1	1	1	1	1	1	1