**Numeric Type Conversions for the Installations & Price Data**

Challenge

How many apps had over 1 billion (that's right - BILLION) installations? How many apps just had a single install?

1. Check the datatype of the Installs column.
2. Count the number of apps at each level of installations.
3. Convert the number of installations (the Installs column) to a numeric data type. Hint: this is a 2-step process. You'll have to make sure you remove non-numeric characters first.

.

.

..

…

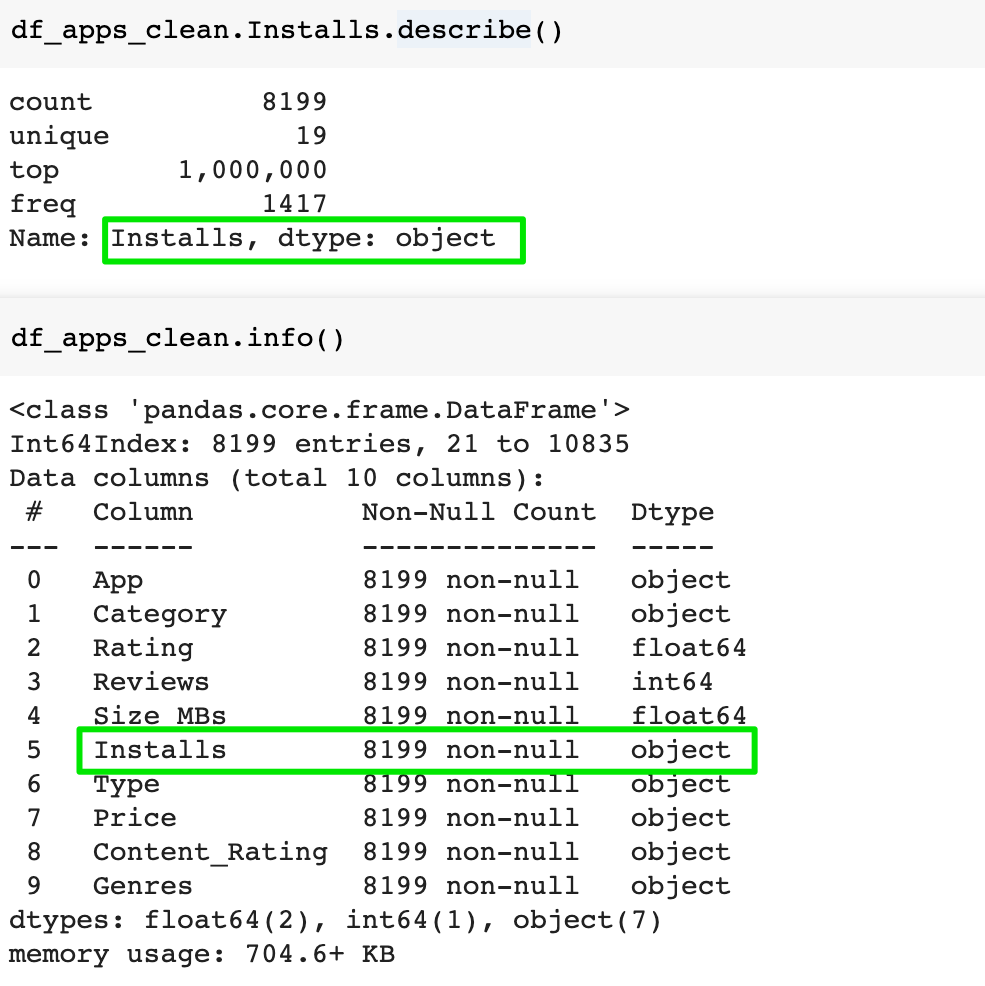
..

.

.

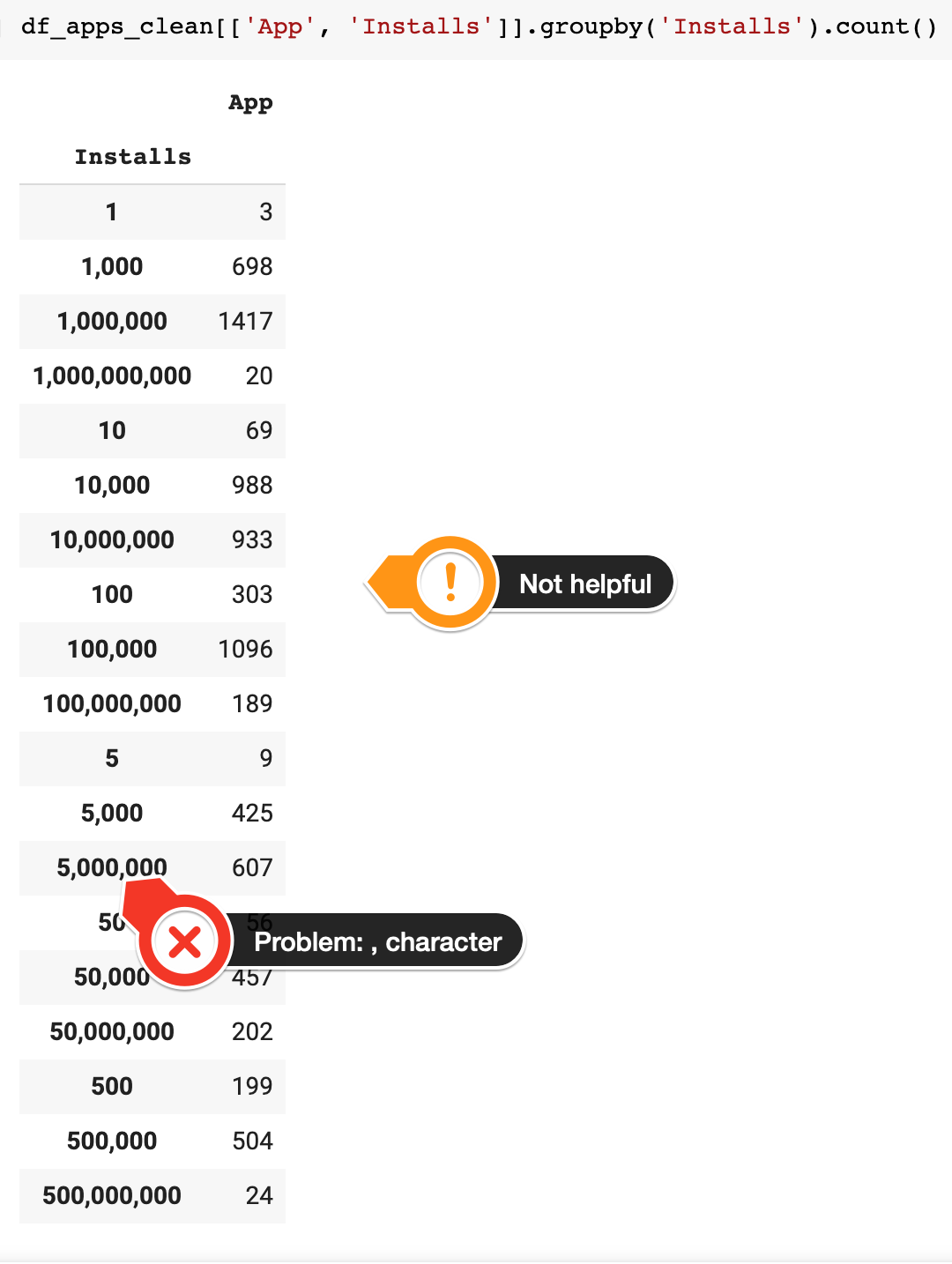
**Solution: Data Cleaning & Converting Data to Numeric Types**

To check the data types you can either use .describe() on the column or .info() on the DataFrame.



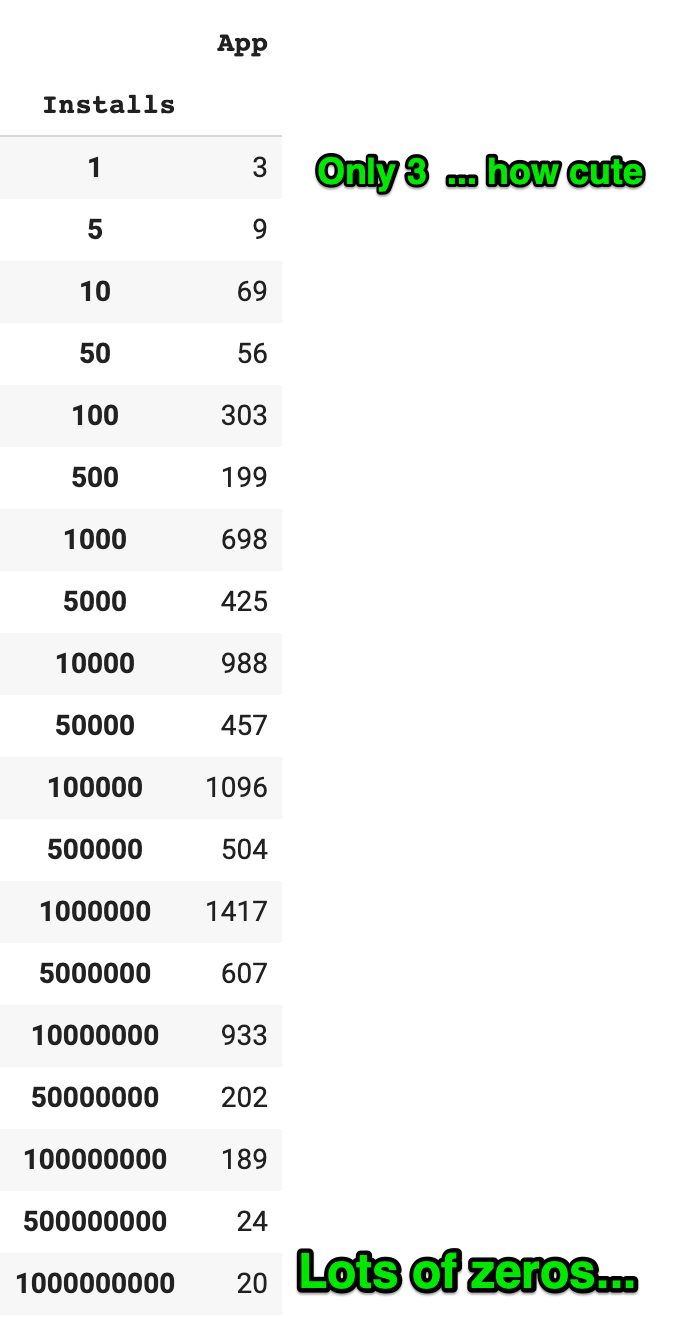
Both of these show that we are dealing with a non-numeric data type. In this case, the type is "object".

If we take two of the columns, say Installs and the App name, we can count the number of entries per level of installations with .groupby() and .count(). However, because we are dealing with a non-numeric data type, the ordering is not helpful. The reason Python is not recognising our installs as numbers is because of the comma (,) characters.



We can remove the comma (,) character - or any character for that matter - from a DataFrame using the string’s[.replace()](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.str.replace.html) method. Here we’re saying: “replace the , with an empty string”. This completely removes all the commas in the Installs column. We can then convert our data to a number using .to\_numeric().

1. df\_apps\_clean.Installs = df\_apps\_clean.Installs.astype(str).str.replace(',', "")
2. df\_apps\_clean.Installs = pd.to\_numeric(df\_apps\_clean.Installs)
3. df\_apps\_clean[['App', 'Installs']].groupby('Installs').count()



Let's examine the Price column more closely.

Challenge

Convert the price column to numeric data. Then investigate the top 20 most expensive apps in the dataset.

Remove all apps that cost more than $250 from the df\_apps\_clean DataFrame.

Add a column called 'Revenue\_Estimate' to the DataFrame. This column should hold the price of the app times the number of installs. What are the top 10 highest-grossing paid apps according to this estimate? Out of the top 10, how many are games?

.

.

..

…

..

.

.

**Solution: Finding the most Expensive Apps and Filtering out the Junk**

If you look at the data type of the price column:

1. df\_apps\_clean.Price.describe()

You also see that is of type object. The reason is the dollar $ signs that we’ve spotted before. To convert the price column to numeric data we use the .replace() method once again, but this time we filter out the dollar sign.

1. df\_apps\_clean.Price = df\_apps\_clean.Price.astype(str).str.replace('$', "")
2. df\_apps\_clean.Price = pd.to\_numeric(df\_apps\_clean.Price)
4. df\_apps\_clean.sort\_values('Price', ascending=False).head(20)

Here’s what we see:



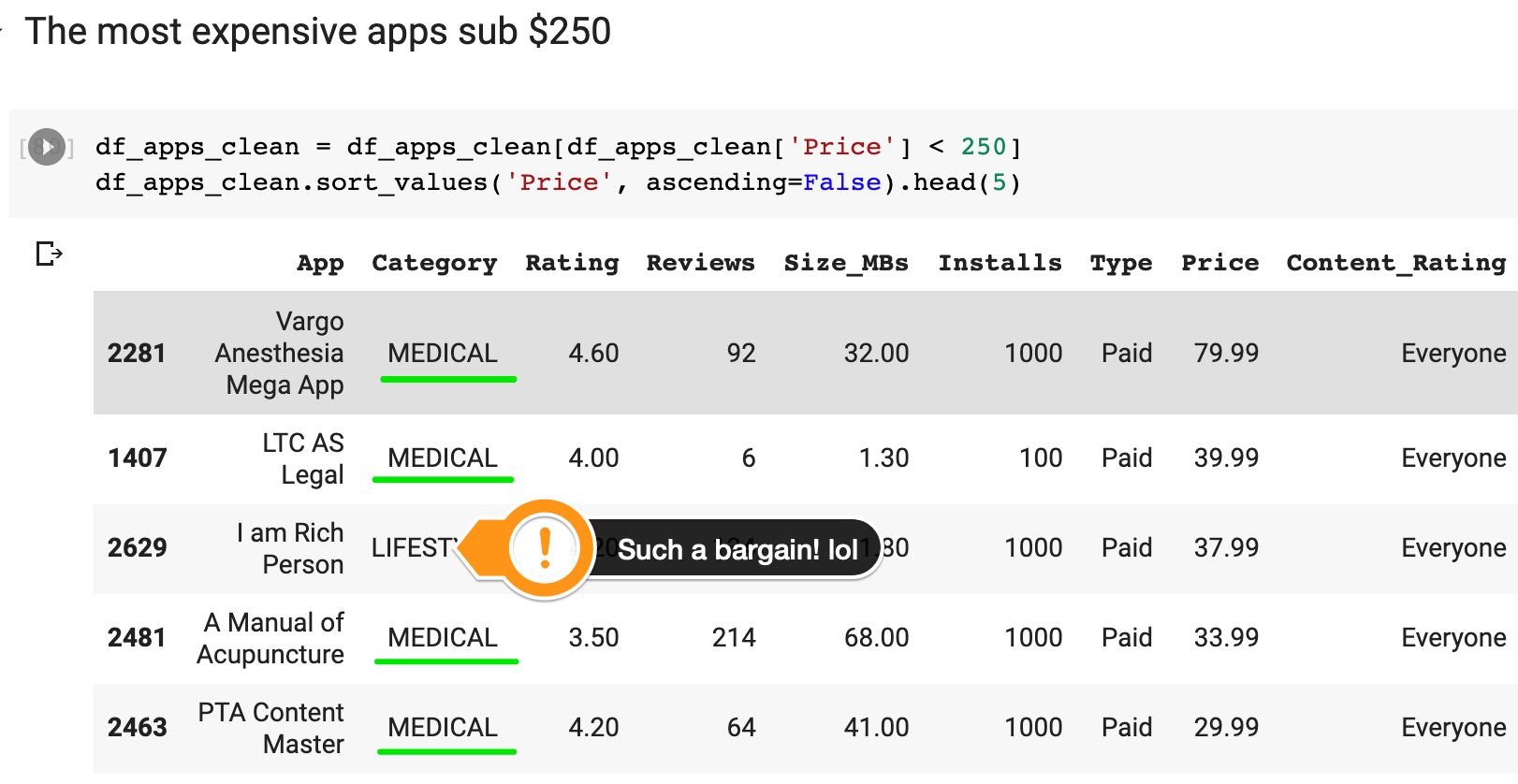
What’s going on here? There are 15 I am Rich Apps in the Google Play Store apparently. They all cost $300 or more, which is the main point of the app. The story goes that in 2008, Armin Heinrich released the very first I am Rich app in the iOS App Store for $999.90. The app does absolutely nothing. It just displays the picture of a gemstone and can be used to prove to your friends how rich you are. Armin actually made a total of 7 sales before the app was hastily removed by Apple. Nonetheless, it inspired a bunch of copycats on the Android App Store, but if you search today, you’ll find all of these apps have disappeared as well. The high installation numbers are likely gamed by making the app was available for free at some point to get reviews and appear more legitimate.



Leaving this bad data in our dataset will misrepresent our analysis of the most expensive 'real' apps. Here’s how we can remove these rows:

1. df\_apps\_clean = df\_apps\_clean[df\_apps\_clean['Price'] < 250]
2. df\_apps\_clean.sort\_values('Price', ascending=False).head(5)

When we look at the top 5 apps now, we see that 4 out of 5 are medical apps.



We can work out the highest grossing paid apps now. All we need to do is multiply the values in the price and the installs column to get the number:

1. df\_apps\_clean['Revenue\_Estimate'] = df\_apps\_clean.Installs.mul(df\_apps\_clean.Price)
2. df\_apps\_clean.sort\_values('Revenue\_Estimate', ascending=False)[:10]

This generously assumes of course that all the installs would have been made at the listed price, which is unlikely, as there are always promotions and free give-aways on the App Stores.



The top spot of the highest-grossing paid app goes to … Minecraft at close to $70 million. It’s quite interesting that Minecraft (along with Bloons and Card Wars) is actually listed in the Family category rather than in the Game category. If we include these titles, we see that 7 out the top 10 highest-grossing apps are games. The Google Play Store seems to be quite flexible with its category labels.

