Instructions

* Copy the dbn column in hs\_directory into a new column called DBN.
* Create a new column called padded\_csd in the class\_size data set.
  + Use the [pandas.Series.apply()](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.apply.html" \t "_blank) method along with a custom function to generate this column.
    - Make sure to apply the function along the data["class\_size"]["CSD"] column.
* Use the addition operator (+) along with the padded\_csd and SCHOOL CODE columns of class\_size, then assign the result to the DBN column of class\_size.
* Display the first few rows of class\_size to double check the DBN column.

The answer

# Initial design and hints.

# help code

# df = df.rename(columns={'oldName1': 'newName1', 'oldName2': 'newName2'})

# Or rename the existing DataFrame (rather than creating a copy)

# df.rename(columns={'oldName1': 'newName1', 'oldName2': 'newName2'}, inplace=True)

## copy and rename a new column in a dataset

# data = data[hs\_directory].rename(columns={'dbn': 'DBN'})

# to just rename

data["hs\_directory"]["DBN"] = data["hs\_directory"]["dbn"]

## create a function to transform any number to str from a given dataset column and add a leading zero to make a panded CSD.

def CSD\_transformer(data\_num):

num2str = str(data\_num)

if len(num2str) > 1:

return num2str

else:

return num2str.zfill(2)

# apply function to the dataset to add the new column (format is tricky) ---> pandas.Series.apply()

# here we create the panded\_csd

data["class\_size"]["padded\_csd"] = data["class\_size"]["CSD"].apply(CSD\_transformer)

# example code to finally put the new BDN in place

# data["num2str"] = data["column\_one"] + dataframe["column\_two"]

# this is simple but brilliant and will generate the DBN.

data["class\_size"]["DBN"] = data["class\_size"]["padded\_csd"] + data["class\_size"]["SCHOOL CODE"]

# print five first lines

print(data["class\_size"].head())

Just two questions about the function in step (11. Inserting DBN fields)

(Q1) Why I don’t need the iteration in the function in this case? :

For example i wrote the code below :

def CSD\_transformer(data\_num):

**for i in data\_num:**

num2str = str(i)

if len(num2str) > 1:

return num2str

else:

**return num2str.zfill(1)**

(Q2) Why do I need to put zfil(2) for the code to make it work , I have the impression that I must have one leading zero if the length of the string is less than 2. It should be zfill(1) and not zfill(2)

Thank you in advance Dataquest team,

Odysseas