In [ ]:

```
#########################################################################
## Pandas Cheat Sheet compiled by Paschal Chukwuemeka Amah Version 2.##
##           Release Date: Sunday February 23, 2020                    ##
#########################################################################

#Create an empty dataframe with three columns
df = pd.DataFrame(columns = ["x", "y", "z"])

#Create an empty dataframe, no columns
df = pd.DataFrame()

#Make a df from a python dictionary. PS: dict values should be python lists
aDict = {'names': ['Paschal', 'Ifeanyi', 'Aliyu']}
df = pd.DataFrame.from_dict(aDict)

#Add a column to an empty df. Rows should be a python list
df['newColName'] = [1, 3, 4]

#Add a column to a df. The rows should be a python list and must be same length as oth
df['newColName'] = [list of equal len as other cols]

#Peep a df showing the first 5 rows. Choose another number by adding it as an argument
df.head()

#Peep a df showing the last 5 rows. Choose another number by adding it as an argument.
df.tail()

#Peep a df showing a randon row. Choose another number by adding it as an argument.
df.sample()

#Quick summary of a dataframe: number of columns and rows
df.shape

#See df columns
df.columns

#Change col names across df
df.columns = {'newName':'oldName', ... 'newNAme':'oldNAme'}

#Delete a df column
del df['colName']

#Change a specific colName
df.rename(columns={'oldNAme': 'newName'}, inplace=True)

#Tranpose a df. Switch rows/cols position. Handy for adding extra rows. T, add a new c
df.T

#Quick summary of all numeric columns showing distribution. Transpose for better view
df.describe()
df.describe().T

#List of the data type of each column of a df
df.dtypes

#Read a comma seperated file
df = pd.read_csv(filename.csv)

#Read a file with a designated seperator other than comma egs: / \ | * -
```

```
df = pd.read_csv(file, sep = 'designated_desperator')

#Read a space delimited file
df = pd.read_csv("filename.txt", delim_whitespace=True)

#Write a csv file. To write without index, add the argument 'index = None'
pd.to_csv(filename.csv)

#Read an ExcelFile
df = pd.read_excel('file.extension')

#Read sheets from an excel file
xls = pd.ExcelFile("file.extension")

#Peep the excel sheets
xls.sheet_names

#Load a sheet
df = xls.parse("sheetName")

#Write to excel
from pandas import ExcelWriter

#Create a write object #Sample is the name of file written
writer = ExcelWriter('sample.xlsx')

#Write to a sheet 'sheet1'
df.to_excel(writer,'Sheet1')
writer.save()

#Read a json file -  especially one sourced from a social media like Twitter
df = pd.read_json("filename.json", lines = True, encoding = "utf-8")

#Merge two dfs on top of each other and reset the index: Condtition is that both must
df_list = [df1, df2]
new_df = pd.concat(df_list).reset_index(drop=True)

######################
## Clean up moves ##
####################

#Show a column of a df (with no spaces in colName OR with space in colName)
df.colName OR df['ColName']

#Change the data type of a column to str
df.colName.astype('str')

#Change the data type of a column to int
df.colName.astype('int')

#Change the data type of a column to float
df.colName.astype('float')

#Convert a col to datetime
pd.to_datetime(df.colName)

#Convert df column to python list
aList = df.colName.tolist()

#Pick particular columns of df by colName
df_new = df_old[[python list of colNames]]
```

```
121
122 #Remove non-characters from column
123 df['colName']= df['colName'](lambda x: x.replace('[^a-zA-Z]', ''))
124
125 #Convert whole dataframe to TitleCase
126 df = df.apply(lambda x: x.astype(str).str.title())
127
128 #To lowercase
129 df = df.apply(lambda x: x.astype(str).str.lower())
130
131 #To UPPERCASE
132 df = df.apply(lambda x: x.astype(str).str.upper())
133
134 #Drop duplicates. By defualt, leaves the first occurence
135 df.drop_duplicates()
136
137 #Replace parts of column content for all rows. Here, all items but the first 4 items a
138 df.loc[:,"colName"].map(lambda x: x.replace(x[4:], ''))
139
140 #Replace parts of column content for all rows. Here, first 9 items are replaced with n
141 df.loc[:,"colName"].map(lambda x: x.replace(x[:9], ''))
142
143 #Replace parts of a column content for all rows. Here, all items but the first 4 items
144 df.loc[:,"colName"].map(lambda x: x.replace(x[4:], '')+"abc")
145
146 #Remove whitespaces in a column for all rows
147 df.loc[:,"colName"].map(lambda x: "".join(x.split()))
148
149 #Merge two columns as str into a new column
150 df["NewColName"] = df["colName_x"].map(str)+ df["colName_y"].map(str)
151
152 #Return only the first x columns of a df
153 df.iloc[:,:x]
154
155 #Return only the first x columns and first x rows of a df
156 df.iloc[:x,:x]
157
158 #Drop the first x columns of a df
159 df.iloc[:,x:]
160
161 #Drop the first x columns and first x rows of a df
162 df.iloc[x:,x:]
163
164 #Make a column of numbers from 0 to to macth the length of a df after importing numpy
165 df["NewCol"] = np.arange(0,len(df))
166
167 #Sort a dataframe according to a column in descending order and reset the index
168 df.sort_values("colName", ascending = True).reset_index(drop=True)
169
170 #Show a column with its rows sorted. for descendinf order, set ascending = False
171 df.colName.sort_values()
172
173 #Show the value count of each item in a column
174 df.colName.sort_values()
175
176 #Show the value count of each item in a column and sort it
177 df.user.value_counts().sort_values()
178
179 #Return parts of df that satisfy a condition in a column
180 df[df['colName'] criteria] # eg1 df[df['colName'] == 'lagos'] eg2 df[df['colName'] > 2
181
```

```
#Drop part of a dataframe tail - 4 here
df.drop(df.tail(4).index)

#Drop part of a dataframe head - 2 here
df.drop(df.head(2).index)

#For a column, replace x ie row with y for all rows, if str in row/line, else return r
df['colName'].map(lamba x: replace(x, y) if 'str' in x else x)

#With random imported, replace the occurence of an (here kano) item with a random memb
an_array = ['obi', 'ada', 'emma', 'yakubu', 'eniye', 'ola']
df['colName'].map(lamba x: replace(x, random.choice(an_array)) if 'kano' in x else x)

#Group df by 2 columns, count and sort in descending order
df.groupby("colName").count().sort_values("Other", ascending = False)

#Group dataframe by 2 columns, count and sort in descending order
df.groupby(["colName1", "colName2"]).count().sort_values("Other", ascending = False)
```