File Reading, Index and Filtering

In [ ]:
```
#Here we are importing panadas library mostly used to work on data cleaning, manipulat
import pandas as pd
#Below we are reading files present in memory for analysis
df = pd.read_csv(r"C:\Users\shashank verma\Downloads\Projects\world_population.csv")
df5 = pd.read_csv(r"C:\Users\shashank verma\Downloads\Projects\Ice Cream Ratings.csv")

df
```

In [28]:
```
df.info() # to get data of all columns datatype
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234 entries, 0 to 233
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Rank                         234 non-null    int64  
 1   CCA3                         234 non-null    object 
 2   Country                      234 non-null    object 
 3   Capital                      234 non-null    object 
 4   Continent                    234 non-null    object 
 5   2022 Population              230 non-null    float64
 6   2020 Population              233 non-null    float64
 7   2015 Population              230 non-null    float64
 8   2010 Population              227 non-null    float64
 9   2000 Population              227 non-null    float64
 10  1990 Population              229 non-null    float64
 11  1980 Population              229 non-null    float64
 12  1970 Population              230 non-null    float64
 13  Area (km²)                   232 non-null    float64
 14  Density (per km²)            230 non-null    float64
 15  Growth Rate                  232 non-null    float64
 16  World Population Percentage  234 non-null    float64
dtypes: float64(12), int64(1), object(4)
memory usage: 31.2+ KB
```

In [27]:
```
df.shape
```

Out[27]:
```
(234, 17)
```

In [ ]:
```
df.head(10) # print top 10 rows
```

In [ ]:
```
df.tail(10) # print bottom 10 rows
```

In [ ]:
```
pd.set_option('display.max.rows', 235)
pd.set_option('display.max.columns', 40)
df
```

In [ ]:
```
df[df['Capital'].str.contains('Delhi')] #Capital column having data Delhi
```

In [ ]:
```
df[df['Rank']>30]
```

In [ ]:
```
df[df['Rank'] <= 10]
```

In [ ]:
```
dfb = df.set_index('Rank', inplace = True) # to set the column as index and commit the
dfb
```

In [ ]:
```
dfa = df.set_index('Country')
dfa
```

In [ ]:
```
df.filter(items = ['Country','Capital']) # to show the specific columns from table
```

In [ ]:
```
df[df['Country'].str.contains('United')] # To show Country column containing United st
```

In [ ]:
```
df.filter(like = 'Europe', axis = 1)
```

In [ ]:
```
df.iloc[5] # to displat 5th row
```

In [ ]:
```
df[df['Rank']> 200]
```

In [ ]:
```
df.loc['Uzbekistan']
```

In [ ]:
```
df.loc['Albania']
```

In [ ]:
```
df.sort_index()
```

In [4]:
```
df[df['Rank'] < 10].sort_values(by=['Country'],ascending=[True]) # ordering the table
```

Out[4]:

	Rank	CCA3	Country	Capital	Continent	2022 Population	2020 Population	2015 Population	F
16	8	BGD	Bangladesh	Dhaka	Asia	1.711864e+08	1.674210e+08	1.578300e+08	1.4
27	7	BRA	Brazil	Brasilia	South America	2.153135e+08	2.131963e+08	2.051882e+08	1.9
41	1	CHN	China	Beijing	Asia	1.425887e+09	1.424930e+09	1.393715e+09	1.3
92	2	IND	India	New Delhi	Asia	1.417173e+09	1.396387e+09	1.322867e+09	1.2
93	4	IDN	Indonesia	Jakarta	Asia	2.755013e+08	2.718580e+08	2.590920e+08	2.4
149	6	NGA	Nigeria	Abuja	Africa	2.185412e+08	2.083274e+08	1.839958e+08	1.6
156	5	PAK	Pakistan	Islamabad	Asia	2.358249e+08	2.271967e+08	2.109693e+08	1.9
171	9	RUS	Russia	Moscow	Europe	1.447133e+08	1.456173e+08	1.446684e+08	1.4
221	3	USA	United States	Washington, D.C.	North America	3.382899e+08	3.359420e+08	3.246078e+08	3.1

GROUP BY IN PYTHON

In [ ]:
```
df.groupby('Continent')
```

In [ ]:
```
df.groupby('Continent').describe() # calculate the mean, max, count and min of rows gr
```

In [ ]:
```
df.groupby('Continent').count() # to find count of rows
```

In [ ]:
```
df.groupby('Continent').sum()

df.groupby('Continent')['Rank'].sum() # to find sum of rank column group by the contir
```

In [ ]:
```
df2 = pd.read_csv(r"C:\Users\shashank verma\Downloads\Projects\Flavors.csv")
df2
```

In [ ]:
```
df2.groupby('Base Flavor')
```

In [ ]:
```
df2.groupby('Base Flavor').count()
```

In [ ]:
```
df2.groupby('Base Flavor').sum()
```

In [7]:
```
df2.groupby('Base Flavor').agg({'Flavor Rating': ['mean','max','count','sum']})
# this agg function will calculate the mean max count sum of column Flavor Rating data
```

Out[7]:

	Flavor Rating			
	mean	**max**	**count**	**sum**
Base Flavor				
Chocolate	8.4	8.8	3	25.2
Vanilla	5.7	10.0	6	34.2

In [ ]:
```
df2.groupby('Base Flavor').describe() # this will calc for all column
```

JOIN IN PYTHON

In [8]:
```
df3 = pd.read_csv(r"C:\Users\shashank verma\Downloads\Projects\LOTR.csv")
df4 = pd.read_csv(r"C:\Users\shashank verma\Downloads\Projects\LOTR 2.csv")
```

In [ ]:
```
df3
```

In [ ]:
```
df4
```

In [9]:
```
df3.merge(df4, how = 'inner', on = ['FellowshipID','FirstName']) # inner join two tabl
```

Out[9]:

	FellowshipID	FirstName	Skills	Age
0	1001	Frodo	Hiding	50
1	1002	Samwise	Gardening	39

In [10]:
```
df3.merge(df4, how = 'outer') # outer join two table
```

Out[10]:

	FellowshipID	FirstName	Skills	Age
0	1001	Frodo	Hiding	50.0
1	1002	Samwise	Gardening	39.0
2	1003	Gandalf	Spells	NaN
3	1004	Pippin	Fireworks	NaN
4	1006	Legolas	NaN	2931.0
5	1007	Elrond	NaN	6520.0
6	1008	Barromir	NaN	51.0

In [ ]:
```
df3.merge(df4, how = 'left') # left join two table
```

In [ ]:
```
df3.merge(df4, how = 'right') # right join two table
```

In [ ]:
```
df3.merge(df4, how = 'cross') # cross join two table
```

DATA CLEANING IN PYTHON

In [12]:
```
dff = pd.read_excel(r"C:\Users\shashank verma\Downloads\Projects\Customer Call List.xl
```

In [13]:
```
dff.set_index('CustomerID', inplace=True)
```

In [ ]:
```
dff
```

In [14]:
```
dff['Last_Name'] = dff["Last_Name"].str.strip("123._/") # removing extra character fro
```

In [ ]:
```
dff
```

In [ ]:
```
dff = dff.drop_duplicates()     # removing the duplicates rows
dff
```

In [ ]:
```
dff = dff.drop(columns = "Not_Useful_Column")  # deleting column from table
dff
```

In [ ]:
```
dff["Phone_Number"] = dff["Phone_Number"].str.replace('[^a-zA-Z0-9]', '', regex = True
dff
```

In [18]:
```
dff["Phone_Number"] = dff["Phone_Number"].apply(lambda x: str(x)) # converting specifi
```

In [ ]:
```

```

In [19]:
```
dff["Phone_Number"] = dff["Phone_Number"].apply(lambda x: x[0:3] + '-' + x[3:6] + '-'
# adding hyphen after 3rd place in row data in specific column
```

```
In [20]: dff["Phone_Number"] = dff["Phone_Number"].str.replace('nan--','')
         # Replace nan-- character with nothing(removing nan--)
```

```
In [21]: dff["Phone_Number"] = dff["Phone_Number"].str.replace('nan','')
```

```
In [22]: dff["Phone_Number"] = dff["Phone_Number"].str.replace('Na','')
```

```
In [23]: #dff['Address'].str.split(',',1, expand = True)
         dff[["Adress","State","Zip Code"]] = dff["Address"].str.split(',', expand = True)
         # splitting column and making new column from it
```

```
In [24]: dff = dff.fillna('') # removing na from
         dff = dff.replace('N/a','')
```

```
In [25]: dff['Paying Customer'] = dff['Paying Customer'].str.replace('Yes','Y')
```

```
In [ ]: dff
```

```
In [26]: dff['Paying Customer'] = dff['Paying Customer'].str.replace('No','N')

         dff["Do_Not_Contact"] = dff["Do_Not_Contact"].str.replace('Yes','Y')

         dff["Do_Not_Contact"] = dff["Do_Not_Contact"].str.replace('No','N')
         dff
```

Out[26]:

	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact	Not_U
CustomerID							
1001	Frodo	Baggins	123-545-5421	123 Shire Lane, Shire	Y	N	
1002	Abed	Nadir	123-643-9775	93 West Main Street	N	Y	
1003	Walter	White		298 Drugs Driveway	N		
1004	Dwight	Schrute	123-543-2345	980 Paper Avenue, Pennsylvania, 18503	Y	Y	
1005	Jon	Snow	876-678-3469	123 Dragons Road	Y	N	
1006	Ron	Swanson	304-762-2467	768 City Parkway	Y	Y	
1007	Jeff	Winger		1209 South Street	N	N	
1008	Sherlock	Holmes	876-678-3469	98 Clue Drive	N	N	
1009	Gandalf		--	123 Middle Earth	Y		
1010	Peter	Parker	123-545-5421	25th Main Street, New York	Y	N	
1011	Samwise	Gamgee		612 Shire Lane, Shire	Y	N	
1012	Harry	Potter		2394 Hogwarts Avenue	Y		
1013	Don	Draper	123-543-2345	2039 Main Street	Y	N	
1014	Leslie	Knope	876-678-3469	343 City Parkway	Y	N	
1015	Toby	Flenderson	304-762-2467	214 HR Avenue	N	N	
1016	Ron	Weasley	123-545-5421	2395 Hogwarts Avenue	N	N	

	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact	Not_U
CustomerID							
1017	Michael	Scott	123-643-9775	121 Paper Avenue, Pennsylvania	Y	N	
1018	Clark	Kent		3498 Super Lane	Y		
1019	Creed	Braton	--			Y	
1020	Anakin	Skywalker	876-678-3469	910 Tatooine Road,	Y	N	

In [ ]:
```
for x in dff.index:                          # here if phone number column data is
    if dff.loc[x, "Phone_Number"] == '':
        dff.drop(x, inplace=True)

dff
```

In [ ]:
```
```