**Q1. How do you load a CSV file into a Pandas DataFrame?**

* **import pandas as pd  
    
  df = pd.read\_csv('data.csv')  
    
  print(df.to\_string())**

**Q2. How do you check the data type of a column in a Pandas DataFrame?**

* **To check the data type in pandas DataFrame we can use the “dtype” attribute. The attribute returns a series with the data type of each column. And the column names of the DataFrame are represented as the index of the resultant series object and the corresponding data types are returned as values of the series object**

**result = df.dtypes**

**Q3. How do you select rows from a Pandas DataFrame based on a condition?**

* **Users can select rows based on a particular column value using '>', '=', '<=', '>=', '!=' operators.**
* **import** pandas as pnd
* record\_1 = {
* 'Name\_1': ['Anuj', 'Ashu', 'Yashi', 'Mark', 'Joshua', 'John', 'Ray', 'Lilly', 'Rose', 'Rachel' ],
* 'Age\_1': [23, 24, 21, 19, 21, 24, 25, 22, 23, 22],
* 'Subjects\_1': ['DBMS', 'ADS', 'ASPM', 'BCM', 'MFCS', 'ADS', 'ASPM', 'TOC', 'Data Mining', 'OOPS'],
* 'Percentage\_1': [88, 62, 85, 71, 55, 78, 70, 66, 71, 89] }
* # Now, we are creating a dataframe
* Data\_Frame = pnd.DataFrame(record\_1, columns = ['Name\_1', 'Age\_1', 'Subject\_1', 'Percentage\_1'])
* print("Given DataFrame: \n", Data\_Frame)
* # Then we will select rows based on condition
* result\_DataFrame = Data\_Frame[Data\_Frame['Percentage\_1'] > 70]
* print('\nFollowing is the Result DataFrame: \n', result\_DataFrame)

**Q4. How do you rename columns in a Pandas DataFrame?**

* **Using rename() function**
* import pandas as pd
* # Define a dictionary containing ICC rankings
* rankings = {'test': ['India', 'South Africa', 'England',
* 'New Zealand', 'Australia'],
* 'odi': ['England', 'India', 'New Zealand',
* 'South Africa', 'Pakistan'],
* 't20': ['Pakistan', 'India', 'Australia',
* 'England', 'New Zealand']}
* # Convert the dictionary into DataFrame
* rankings\_pd = pd.DataFrame(rankings)
* # Before renaming the columns
* print(rankings\_pd)
* rankings\_pd.rename(columns = {'test':'TEST'}, inplace = True)
* # After renaming the columns
* print("\nAfter modifying first column:\n", rankings\_pd.columns)

**Q5. How do you drop columns in a Pandas DataFrame?**

data = {

    'A': ['A1', 'A2', 'A3', 'A4', 'A5'],

    'B': ['B1', 'B2', 'B3', 'B4', 'B5'],

    'C': ['C1', 'C2', 'C3', 'C4', 'C5'],

    'D': ['D1', 'D2', 'D3', 'D4', 'D5'],

    'E': ['E1', 'E2', 'E3', 'E4', 'E5']}

# Convert the dictionary into DataFrame

df = pd.DataFrame(data)

# Remove column name 'A'

df.drop(['A'], axis=1)

**Q6. How do you find the unique values in a column of a Pandas DataFrame?**

* **To get unique values from a column in a DataFrame, use the unique()To count the unique values from a column in a DataFrame, use the nunique()**
* df = pd.DataFrame({'team':['A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'],
* 'points': [25, 12, 15, 14, 19, 23, 15,25],
* 'assists': [5, 7, 7, 9, 12, 9, 9, 4],
* 'rebounds': [11, 8, 10, 6, 6, 5, 9, 12]})
* print(df['points'].unique())
* print(df['team'].nunique())
* print(df['points'].nunique())
* print(df['rebounds'].unique())

**Q7. How do you find the number of missing values in each column of a Pandas DataFrame?**

* We can use pandas **“isnull()”** function to find out all the fields which have missing values. This will return True if a field has missing values and false if the field does not have missing values.
* df = pd.DataFrame({'team':['A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'],
* 'points': [25, 12, 15, 14, 19, 23, 15,25],
* 'assists': [5, 7, 7, 9, 12, 9, 9, 4],
* 'rebounds': [11, 8, 10, 6, 6, 5, 9, 12]})
* print(df['points'].isnull())

**Q8. How do you fill missing values in a Pandas DataFrame with a specific value?**

* **we use fillna() function these function replace NaN values with some value of their own. All these function help in filling a null values in datasets of a DataFrame.**
* **import pandas as pd**
* **import numpy as np**
* **dict = {'First Score':[100, 90, np.nan, 95],**
* **'Second Score': [30, 45, 56, np.nan],**
* **'Third Score':[np.nan, 40, 80, 98]}**
* **df = pd.DataFrame(dict)**
* **df.fillna(0)**

**Q9. How do you concatenate two Pandas DataFrames?**

* **We’ll pass two dataframes to pd.concat() method in the form of a list and mention in which axis you want to concat, i.e. axis=0 to concat along rows, axis=1 to concat along columns.**
* **df1 = pd.DataFrame(np.random.randint(25, size=(4, 4)),**
* **index=["1", "2", "3", "4"],**
* **columns=["A", "B", "C", "D"])**
* **df2 = pd.DataFrame(np.random.randint(25, size=(6, 4)),**
* **index=["5", "6", "7", "8", "9", "10"],**
* **columns=["A", "B", "C", "D"])**
* **vertical\_concat = pd.concat([df1, df2], axis=0)**
* **# concatenating df3 and df4 along columns**
* **horizontal\_concat = pd.concat([df3, df4], axis=1)**
* **display(vertical\_concat, horizontal\_concat)**

**Q10. How do you merge two Pandas DataFrames on a specific column?**

* **We can merge two DataFrames on certain column using merge() function**
* **# importing modules**
* **import pandas as pd**
* **# creating a dataframe**
* **df1 = pd.DataFrame({'Name':['Raju', 'Rani', 'Geeta', 'Sita', 'Sohit'],**
* **'Marks':[80, 90, 75, 88, 59]})**
* **# creating another dataframe with different data**
* **df2 = pd.DataFrame({'Name':['Raju', 'Divya', 'Geeta', 'Sita'],**
* **'Grade':['A', 'A', 'B', 'A'],**
* **'Rank':[3, 1, 4, 2 ],**
* **'Gender':['Male', 'Female', 'Female', 'Female']})**
* **# display df1**
* **display(df1)**
* **# display df2**
* **display(df2)**

|  |
| --- |
| **df1.merge(df2[['Name', 'Grade', 'Rank']])** |

**Q11. How do you group data in a Pandas DataFrame by a specific column and apply an aggregation function?**

**We can use groupby and aggregate function to group the column and aggregate them and find mean,min,max values of a particular column.**

**data = {"Team": ["Red Sox", "Red Sox", "Red Sox", "Red Sox", "Red Sox", "Red Sox", "Yankees", "Yankees", "Yankees", "Yankees", "Yankees", "Yankees"],**

**"Pos": ["Pitcher", "Pitcher", "Pitcher", "Not Pitcher", "Not Pitcher", "Not Pitcher", "Pitcher", "Pitcher", "Pitcher", "Not Pitcher", "Not Pitcher", "Not Pitcher"],**

**"Age": [24, 28, 40, 22, 29, 33, 31, 26, 21, 36, 25, 31]}**

**df = pd.DataFrame(data)**

**print(df)**

**grouped\_single = df.groupby('Team').agg({'Age': ['mean', 'min', 'max']}**

**print(grouped\_single)**

**Q12. How do you pivot a Pandas DataFrame?**

**import pandas as pd**

**# creating a dataframe**

**df = pd.DataFrame({'A': ['John', 'Boby', 'Mina'],**

**'B': ['Masters', 'Graduate', 'Graduate'],**

**'C': [27, 23, 21]})**

**Df**

|  |
| --- |
| **df.pivot('A', 'B', 'C')** |

**Q13. How do you change the data type of a column in a Pandas DataFrame?**

**df = pd.DataFrame({'team':['A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'],**

**'points': [25, 12, 15, 14, 19, 23, 15,25],**

**'assists': [5, 7, 7, 9, 12, 9, 9, 4],**

**'rebounds': [11, 8, 10, 6, 6, 5, 9, 12]})**

**print(df['points'].isnull())**

**print(df.dtypes)**

**df=df.astype(str)**

**print(df.dtypes)**

**Q14. How do you sort a Pandas DataFrame by a specific column?**

**We can sort a Pandas Datafrsme using the sort\_values function**

**age\_list = [['Brazil', 1952, 8425333, 'Asia'],**

**['Australia', 1957, 9712569, 'Oceania'],**

**['Russia', 1962, 76039390, 'Americas'],**

**['China', 1957, 637408000, 'Asia'],**

**['Zimbabwe', 1957, 44310863, 'Europe'],**

**['India', 1952, 3.72e+08, 'Asia'],**

**['United States', 1957, 171984000, 'Americas']]**

**df = pd.DataFrame(age\_list, columns=['Country', 'Year',**

**'Population', 'Continent'])**

**print(df.sort\_values(by=['Country'],ascending=False))**

**Q15. How do you create a copy of a Pandas DataFrame?**

**Using copy() function we can create a copy of DataFrame**

ge\_list = [['Brazil', 1952, 8425333, 'Asia'],

            ['Australia', 1957, 9712569, 'Oceania'],

            ['Russia', 1962, 76039390, 'Americas'],

            ['China', 1957, 637408000, 'Asia'],

            ['Zimbabwe', 1957, 44310863, 'Europe'],

            ['India', 1952, 3.72e+08, 'Asia'],

            ['United States', 1957, 171984000, 'Americas']]

df=pd.DataFrame(age\_list)

copy=df.copy(deep=True)

print(copy)

Q16. How do you filter rows of a Pandas DataFrame by multiple conditions?

import pandas as pd

# assign data

dataFrame = pd.DataFrame({'Name': [' RACHEL  ', ' MONICA  ', ' PHOEBE  ',

                                   '  ROSS    ', 'CHANDLER', ' JOEY    '],

                          'Age': [30, 35, 37, 33, 34, 30],

                          'Salary': [100000, 93000, 88000, 120000, 94000, 95000],

                          'JOB': ['DESIGNER', 'CHEF', 'MASUS', 'PALENTOLOGY',

                                  'IT', 'ARTIST']})

# filter dataframe

display(dataFrame.loc[(dataFrame['Salary']>=100000) & (dataFrame['Age']< 40) & (dataFrame['JOB'].str.startswith('D')),

                    ['Name','JOB']])

**Q17. How do you calculate the mean of a column in a Pandas DataFrame?**

**Pandas dataframe.mean() function return the mean of the values for the requested axis. If the method is applied on a pandas series object, then the method returns a scalar value which is the mean value of all the observations in the dataframe. If the method is applied on a pandas dataframe object, then the method returns a pandas series object which contains the mean of the values over the specified axis.**

**# importing pandas as pd**

**import pandas as pd**

**df = pd.DataFrame({"A":[12, 4, 5, 44, 1],**

**"B":[5, 2, 54, 3, 2],**

**"C":[20, 16, 7, 3, 8],**

**"D":[14, 3, 17, 2, 6]})**

**Df**

**df.mean(axis = 0)**

**df.mean(axis = 1, skipna = True)**

**Q18. How do you calculate the standard deviation of a column in a Pandas DataFrame?**

**Pandas dataframe.std() function return sample standard deviation over requested axis. By default the standard deviations are normalized by N-1.**

|  |
| --- |
| **# finding STD**  **df.std(axis = 0, skipna = True)** |

**Q19. How do you calculate the correlation between two columns in a Pandas DataFrame?**

**Correlation is used to summarize the strength and direction of the linear association between two quantitative variables. It is denoted by r and values between -1 and +1. A positive value for r indicates a positive association, and a negative value for r indicates a negative association.**

**# import pandas module**

**import pandas as pd**

**# create dataframe with 3 columns**

**data = pd.DataFrame({**

**"column1": [12, 23, 45, 67],**

**"column2": [67, 54, 32, 1],**

**"column3": [34, 23, 56, 23]**

**}**

**)**

**# display dataframe**

**print(data)**

**# correlation between column 1 and column2**

**print(data['column1'].corr(data['column2']))**

**# correlation between column 2 and column3**

**print(data['column2'].corr(data['column3']))**

**# correlation between column 1 and column3**

**print(data['column1'].corr(data['column3']))**

**Q20. How do you select specific columns in a DataFrame using their labels?**

**The [ ] is used to select a column by mentioning the respective column name.**

age\_list = [['Brazil', 1952, 8425333, 'Asia'],

            ['Australia', 1957, 9712569, 'Oceania'],

            ['Russia', 1962, 76039390, 'Americas'],

            ['China', 1957, 637408000, 'Asia'],

            ['Zimbabwe', 1957, 44310863, 'Europe'],

            ['India', 1952, 3.72e+08, 'Asia'],

            ['United States', 1957, 171984000, 'Americas']]

df = pd.DataFrame(age\_list, columns=['Country', 'Year',

                                    'Population', 'Continent'])

result=df[['Year','Country']]

print(result)

**Q21. How do you select specific rows in a DataFrame using their indexes?**

age\_list = [['Brazil', 1952, 8425333, 'Asia'],

            ['Australia', 1957, 9712569, 'Oceania'],

            ['Russia', 1962, 76039390, 'Americas'],

            ['China', 1957, 637408000, 'Asia'],

            ['Zimbabwe', 1957, 44310863, 'Europe'],

            ['India', 1952, 3.72e+08, 'Asia'],

            ['United States', 1957, 171984000, 'Americas']]

df = pd.DataFrame(age\_list, columns=['Country', 'Year',

                                    'Population', 'Continent'])

df.set\_index("Country",inplace=True)

res=df.loc["Zimbabwe"]

print(res)

**Q22. How do you sort a DataFrame by a specific column?**

**We can sort a Pandas Datafrsme using the sort\_values function**

**age\_list = [['Brazil', 1952, 8425333, 'Asia'],**

**['Australia', 1957, 9712569, 'Oceania'],**

**['Russia', 1962, 76039390, 'Americas'],**

**['China', 1957, 637408000, 'Asia'],**

**['Zimbabwe', 1957, 44310863, 'Europe'],**

**['India', 1952, 3.72e+08, 'Asia'],**

**['United States', 1957, 171984000, 'Americas']]**

**df = pd.DataFrame(age\_list, columns=['Country', 'Year',**

**'Population', 'Continent'])**

**print(df.sort\_values(by=['Country'],ascending=False))**

**Q23. How do you create a new column in a DataFrame based on the values of another column?**

**We can use DataFrame.apply() function to create a new Column in a DataFrame based on the values of another column**

**# importing pandas as pd**

**import pandas as pd**

**# Creating the DataFrame**

**df = pd.DataFrame({'Date':['10/2/2011', '11/2/2011', '12/2/2011', '13/2/2011'],**

**'Event':['Music', 'Poetry', 'Theatre', 'Comedy'],**

**'Cost':[10000, 5000, 15000, 2000]})**

**# Print the dataframe**

**print(df)**

**# using apply function to create a new column**

**df['Discounted\_Price'] = df.apply(lambda row: row.Cost -(row.Cost \* 0.1), axis = 1)**

**# Print the DataFrame after addition**

**# of new column**

**print(df)**

**Q24. How do you remove duplicates from a DataFrame?**

**Pandas drop\_duplicates() method helps in removing duplicates from the**[**Pandas Dataframe**](https://www.geeksforgeeks.org/python-pandas-dataframe/)**In**[**Python**](https://www.geeksforgeeks.org/python-programming-language/)**.**

**import pandas as pd**

**data = {**

**"A": ["TeamA", "TeamB", "TeamB", "TeamC", "TeamA"],**

**"B": [50, 40, 40, 30, 50],**

**"C": [True, False, False, False, True]**

**}**

**df = pd.DataFrame(data)**

**display(df.drop\_duplicates())**

**Q25. What is the difference between .loc and .iloc in Pandas?**

* **loc[] is used to select rows and columns by Names/Labels**
* **iloc[] is used to select rows and columns by Integer Index/Position. zero based index position.**