**PandasAssignment**

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

Answer : To load a CSV file into a Pandas DataFrame, we use the pd.read\_csv() function provided in the Pandas library. we simply need to pass the file path of the CSV file as an argument to this function, and it will return a DataFrame containing the data from the CSV file

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

Answer : To check the data type of a column in a Pandas DataFrame, we can use the dtypes attribute of the DataFrame, which returns the data types of all columns in the form of a Series. You can then use standard Python indexing to select the specific column whose data type you want to check.

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

Answer : To select rows from a Pandas DataFrame based on a condition, you can use boolean indexing. You can create a boolean condition by applying a comparison or logical operator to one or more columns of the DataFrame, and then pass this condition within square brackets to the DataFrame to filter out rows that meet the condition.

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

Answer : To rename columns in a Pandas DataFrame, you can use the rename() method provided by the DataFrame. You need to specify a dictionary that maps the current column names to the new column names as the argument to this method, and it will update the column names accordingly.

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

Answer : To drop columns in a Pandas DataFrame, you can use the drop() method provided by the DataFrame. You simply need to specify the names of the columns you want to drop as a list or an array as the argument to this method, along with the axis parameter set to 1 to indicate that you want to drop columns and not rows.

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

Answer : To find the unique values in a column of a Pandas DataFrame, you can use the unique() method provided by the Series, which represents a single column of a DataFrame. This method returns an array containing all the unique values in the column.

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

Answer : To find the number of missing values in each column of a Pandas DataFrame, you can use the isna() method provided by the DataFrame, which returns a boolean mask indicating whether each element in the DataFrame is missing or not. You can then use the sum() method along with the axis parameter set to 0 to sum the number of missing values in each column.

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

Answer : To fill missing values in a Pandas DataFrame with a specific value, you can use the fillna() method provided by the DataFrame. You simply need to specify the value you want to fill as an argument to this method, and it will replace all the missing values in the DataFrame with that value.

Q9. How do you concatenate two Pandas DataFrames?

Answer : To concatenate two Pandas DataFrames, you can use the concat() function provided by the Pandas library. You need to pass the DataFrames you want to concatenate as a list or tuple to this function, along with the axis parameter set to 0 to indicate that you want to concatenate them vertically (i.e., along rows).

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

Answer : To merge two Pandas DataFrames on a specific column, you can use the merge() function provided by the Pandas library. You need to specify the DataFrames you want to merge, the column on which you want to merge them, and the type of merge you want to perform (e.g., inner, outer, left, or right) as arguments to this function.

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

Answer : To group data in a Pandas DataFrame by a specific column and apply an aggregation function, you can use the groupby() method provided by the DataFrame. You need to specify the column by which you want to group the data as an argument to this method, and then chain it with an aggregation function (e.g., mean(), `sum

Q12. How do you pivot a Pandas DataFrame?

Answer: Pivoting a Pandas DataFrame involves reshaping the data so that the values of one column are used as columns themselves, with the original column becoming the index, and another column providing the values. To do this, you can use the pivot() method, specifying the index, columns, and values to be used. Here's an example:

df\_pivoted = df.pivot(index='column\_name1', columns='column\_name2', values='column\_name3')

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

Answer: To change the data type of a column in a Pandas DataFrame, you can use the astype() method, specifying the column name and the desired data type as the argument. Here's an example:

df['column\_name'] = df['column\_name'].astype('new\_data\_type')

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

Answer: To sort a Pandas DataFrame by a specific column, you can use the sort\_values() method, specifying the column name to sort by as the argument. Here's an example:

df\_sorted = df.sort\_values('column\_name')

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

Answer: To create a copy of a Pandas DataFrame, you can use the copy() method, which creates a new DataFrame with the same data and column names, but as a separate object in memory. Here's an example:

df\_copy = df.copy()

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

Answer: To filter rows of a Pandas DataFrame by multiple conditions, you can use logical operators such as & (and) or | (or) to combine multiple conditions within parentheses. Here's an example:

filtered\_df = df[(df['column1'] > value1) & (df['column2'] == value2)]

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

Answer: To calculate the mean of a column in a Pandas DataFrame, you can use the mean() method on the specific column. Here's an example:

mean\_value = df['column\_name'].mean()

Q18. How do you calculate the standard deviation of a column in a Pandas DataFrame? Answer: To calculate the standard deviation of a column in a Pandas DataFrame, you can use the std() method on the specific column. Here's an example:

std\_value = df['column\_name'].std()

Q19. How do you calculate the correlation between two columns in a Pandas DataFrame? Answer: To calculate the correlation between two columns in a Pandas DataFrame, you can use the corr() method on the two columns. Here's an example:

correlation = df['column\_name1'].corr(df['column\_name2'])

Q20. How do you select specific columns in a DataFrame using their labels? Answer: To select specific columns in a Pandas DataFrame using their labels, you can pass a list of column names as the argument inside square brackets. Here's an example:

selected\_columns = df[['column\_name1', 'column\_name2', 'column\_name3']]

Q21. How do you select specific rows in a DataFrame using their indexes? Answer: To select specific rows in a Pandas DataFrame using their indexes, you can use the .iloc[] method, which allows you to select rows by their integer-based index. You can specify the row index or a range of row indexes as the argument. Here's an example:

selected\_rows = df.iloc[1:5] # selects rows with index 1, 2, 3, and 4

Q22. How do you sort a DataFrame by a specific column? Answer: To sort a Pandas DataFrame by a specific column, you can use the .sort\_values() method, specifying the column name to sort by as the argument. You can also specify the sorting order (ascending or descending) using the ascending parameter. Here's an example:

df\_sorted = df.sort\_values('column\_name', ascending=False) # sorts by 'column\_name' in descending order

Q23. How do you create a new column in a DataFrame based on the values of another column? Answer: To create a new column in a Pandas DataFrame based on the values of another column, you can simply assign a new column name and provide the desired calculation or transformation using the existing column. Here's an example:

df['new\_column'] = df['existing\_column'] \* 2 # creates a new column with values double the existing column

Q24. How do you remove duplicates from a DataFrame? Answer: To remove duplicates from a Pandas DataFrame, you can use the .drop\_duplicates() method, which removes all but the first occurrence of each duplicated value in the specified column(s) or across the entire DataFrame. Here's an example:

df = df.drop\_duplicates(subset='column\_name') # removes duplicates based on 'column\_name'

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

Answer: The .loc and .iloc are both indexing methods in Pandas, they are used for different purposes.

* .loc is label-based indexing, which allows you to select rows and columns based on their labels (i.e., column names and row indexes with meaningful values). You can use labels or boolean conditions to select specific rows and columns.
* .iloc is integer-based indexing, which allows you to select rows and columns based on their integer-based indexes (i.e., positional index regardless of the label values). You can use integer-based indexes or integer-based boolean conditions to select specific rows and columns.

In summary, .loc is used for label-based indexing, and .iloc is used for integer-based indexing in Pandas.