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

**Answer :**

You can load a CSV (Comma-Separated Values) file into a Pandas DataFrame using the **pandas.read\_csv() function.**

**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, you can use the **dtypes** attribute or the **dtype method** on the specific column.

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

**Answer :**

You can select rows from aPandas DataFrame based on a condition by using **boolean indexing.** Boolean indexing allows you to filter rows that meet a specific condition or set of conditions.

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

**Answer :**

You can rename columns in a Pandas DataFrame using the **rename() method** or by directly assigning new column names to the columns attribute.

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

**Answer :**

You can drop columns in a Pandas DataFrame using the **drop() method**

**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 or the value\_counts() method.

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

**Answer :**

You can find the number of missing (NaN) values in each column of a Pandas DataFrame using the **isna()** o**r isnull() method,** followed by the **sum() method**.

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

**Answer :**

You can fill missing values in a Pandas DataFrame with a specific value using the **fillna() method.**

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

**Answer :**

You can concatenate two Pandas DataFrames vertically or horizontally using the **pd.concat() function.**

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

You can merge two Pandas DataFrames on a specific column using the **pd.merge() function.** This function allows you to combine DataFrames by specifying a common column or key on which to merge.

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

**Answer :**

You can group data in a Pandas DataFrame by a specific column and apply an aggregation function using the **groupby() method** along with aggregation functions li**ke sum(), mean(), count(), max(), min()**, and custom aggregation functions.

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

**Answer :**

You can pivot a Pandas DataFrame using the **pivot() method** or the **pivot\_table() method.** Pivoting reshapes the DataFrame by converting columns into rows or vice versa, allowing you to transform the data's structure for analysis.

**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** or the **pd.to\_numeric() function** (for numeric conversions) or other specific conversion functions depending on the desired data type.

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

**Answer :**

You can sort a Pandas DataFrame by a specific column using the **sort\_values() method.** This method allows you to specify the column by which you want to sort the DataFrame.

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

**Answer :**

You can create a copy of a Pandas DataFrame using the **copy() method** or by directly assigning the DataFrame to a new variable.

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

**Answer :**

You can filter rows of a Pandas DataFrame by multiple conditions using **logical operators** like & (and) and | (or) to combine the conditions.

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

**Answer :**

You can calculate the mean (average) of a column in a Pandas DataFrame using the **mean() method.**

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

**Answer :**

You can calculate the standard deviation of a column in a Pandas DataFrame using the **std() method.**

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

**Answer :**

You can calculate the correlation between two columns in a Pandas DataFrame using the **corr() method.** This method computes the Pearson correlation coefficient, which measures the linear correlation between two numeric variables.

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

**Answer :**

You can select specific columns in a Pandas DataFrame using their labels by using square **brackets []** or the **loc[] method.**

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

**Answer :**

You can select specific rows in a Pandas DataFrame using their indexes by using the **iloc[] method** or the **loc[] method.**

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

**Answer :**

To sort a DataFrame by a specific column, you can use the **sort\_values() method.** This method allows you to specify the column by which you want to sort the DataFrame.

**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 DataFrame based on the values of another column, you can simply assign a new column name to the DataFrame and specify the desired values based on the existing column(s).

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

**Answer :**

You can remove duplicates from a Pandas DataFrame using the **drop\_duplicates() method** or by using boolean indexing to select unique rows.

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

**Answer :**

In Pandas, both .loc and .iloc are used for selecting data from a DataFrame, but they have different methods of indexing and selecting data

**.loc (label-based selection):**

* .loc is primarily used for selecting data by labels, which means you specify row and column labels to access data.
* It uses the format df.loc[row\_label, column\_label] to select data.
* You can select data by label or a boolean condition based on labels.
* It is inclusive of both the start and stop indices when using slices.
* You can use labels or label-based boolean conditions for both rows and columns.

**Example:** df.loc['row\_label', 'column\_label']

**.iloc (integer-based selection):**

* .iloc is used for selecting data by integer-based positions, which means you specify row and column indices (integers) to access data.
* It uses the format df.iloc[row\_index, column\_index] to select data.
* You can select data by integer positions or a boolean condition based on integer positions.
* It is exclusive of the stop index when using slices (similar to Python's regular slicing).
* You can only use integers or integer-based boolean conditions for both rows and columns.

**Example:** df.iloc[1, 2]