**PandasAssignment**

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

2. You can check the data type of a column in a Pandas DataFrame using the **dtype** property or the **.astype()** method.

3. You can select rows from a Pandas DataFrame based on a condition using boolean indexing, for example: **df[df['column\_name'] == value]**

4. You can rename columns in a Pandas DataFrame using the **.rename()** method, for example: **df.rename(columns={'old\_name': 'new\_name'}, inplace=True)**

5. You can drop columns in a Pandas DataFrame using the **.drop()** method, for example: **df.drop(columns=['column\_name'], inplace=True)**

6. You can find the unique values in a column of a Pandas DataFrame using the **.unique()** method, for example: **df['column\_name'].unique()**

7. You can find the number of missing values in each column of a Pandas DataFrame using the **.isna()** method, for example: **df.isna().sum()**

8. You can fill missing values in a Pandas DataFrame with a specific value using the **.fillna()** method, for example: **df.fillna(value, inplace=True)**

9. You can concatenate two Pandas DataFrames using the **pandas.concat()** function, for example: **pandas.concat([df1, df2], axis=0)**

10. You can merge two Pandas DataFrames on a specific column using the **.merge()** method, for example: **pd.merge(df1, df2, on='column\_name')**

11. You can group data in a Pandas DataFrame by a specific column and apply an aggregation function using the **.groupby()** method, for example: **df.groupby('column\_name').agg({'column\_name': 'mean'})**

12. You can pivot a Pandas DataFrame using the **.pivot()** method, for example: **df.pivot(index='column\_name', columns='column\_name', values='column\_name')**

13. You can change the data type of a column in a Pandas DataFrame using the **.astype()** method, for example: **df['column\_name'] = df['column\_name'].astype(int)**

14. You can sort a Pandas DataFrame by a specific column using the **.sort\_values()** method, for example: **df.sort\_values('column\_name')**

15. You can create a copy of a Pandas DataFrame using the **.copy()** method, for example: **df2 = df1.copy()**

16. You can filter rows of a Pandas DataFrame by multiple conditions using the **&** (and) or **|** (or) operator, for example: **df[(df['column\_name1'] > value1) & (df['column\_name2'] < value2)]**

17. You can calculate the mean of a column in a Pandas DataFrame using the **.mean()** method, for example: **df['column\_name'].mean()**

18. You can calculate the standard deviation of a column in a Pandas DataFrame using the **.std()** method, for example: **df['column\_name'].std()**

19. You can calculate the correlation between two columns in a Pandas DataFrame using the **.corr()** method, for example: **df['column\_name1'].corr(df['column\_name2'])**

20. You can select specific columns in a DataFrame using their labels by indexing the DataFrame with the column names, for example: **df[['column\_name1', 'column\_name2']]**

21. You can select specific rows in a DataFrame using their indexes by indexing the DataFrame with the **.loc** or **.iloc** attribute, for example: **df.loc[index\_value]** or **df.iloc[index\_value]**

22. You can sort a DataFrame by a specific column using the **.sort\_values()** method, for example: **df.sort\_values('column\_name')**

23. You can create a new column in a DataFrame based on the values of another column using assignment, for example: **df['new\_column'] = df['column\_name'] \* 2**

24. You can remove duplicates from a DataFrame using the **.drop\_duplicates()** method, for example: **df.drop\_duplicates()**

25. The **.loc** and **.iloc** attributes in Pandas are used to select rows and columns from a DataFrame by label and by integer-location, respectively. The **.loc** attribute uses the DataFrame's index and column labels to select data, while **.iloc** uses the index and column integer-locations.