**Wednesday, March 20, 2024**

The pandas library was founded by Wes McKinney. He started developing pandas in 2008

**1. What is pandas in Python?**

**Pandas is an open-source Python library which is used to clean, analyze, and manipulate datasets. Pandas is built on top of the NumPy library, i.e., its data structures Series and DataFrame are the upgraded versions of NumPy arrays.**

**Cam – dss & df**

**2. How do you access the top 6 rows and last 7 rows of a pandas DataFrame?**

**The head() method in pandas is used to access the initial rows of a DataFrame, and tail() method is used to access the last rows.**

**To access the top 6 rows: dataframe\_name.head(6)**

**To access the last 7 rows: dataframe\_name.tail(7)**

**3. Why doesn’t DataFrame.shape have parenthesis?**

**In pandas, shape is an attribute and not a method. So, you should access it without parentheses.**

**DataFrame.shape outputs a tuple with the number of rows and columns in a DataFrame.**

**4. What is the difference between Series and DataFrame?**

**DataFrame: The pandas DataFrame will be in tabular format with multiple rows and columns where each column can be of different data types.**

**Series: The Series is a one-dimensional labeled array that can store any data type, but all of its values should be of the same data type. The Series data structure is more like single column of a DataFrame.The Series data structure consumes less memory than a DataFrame. So, certain data manipulation tasks are faster on it.**

**5. What is an index in pandas?**

**The index is a series of labels that can uniquely identify each row of a DataFrame. The index can be of any datatype like integer, string, hash, etc.,**

**import pandas as pd**

**data = { 'Name': ['John', 'Matt', 'John', 'Matt'],**

**'Marks': [10, 20, 30, 15]}**

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

**print(df.index)**

**Output: RangeIndex(start=0, stop=4, step=1)**

**6. What is Multi indexing in pandas?**

**For example, you have the columns “name”, “age”, “address”, and “marks” in a DataFrame. Any of the above columns may not have unique values for all the different rows and are unfit as indexes. However, the columns “name” and “address” together may uniquely identify each row of the DataFrame. So you can set both columns as the index. Your DataFrame now has a multi-index or hierarchical index.**

**8. What is the difference between loc and iloc?**

**Both loc and the iloc methods in pandas are used to select subsets of a DataFrame.**

**The difference between the loc and iloc functions is that the loc function selects rows using row labels (e.g. tea) whereas the iloc function selects rows using their integer positions.**

**df.loc['tea']**

**df.iloc[2]**

**9. Show two different ways to create a pandas DataFrame**

**Using Python Dictionary:**

**import pandas as pd**

**data = {'Name': ['John', 'Cataline', 'Matt'],**

**'Age': [50, 45, 30],**

**'City': ['Austin', 'San Francisco', 'Boston'],**

**'Marks' : [70, 80, 95]}**

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

**Using Python Lists:**

**import pandas as pd**

**data = [['John', 25, 'Austin',70],**

**['Cataline', 30, 'San Francisco',80],**

**['Matt', 35, 'Boston',90]]**

**columns = ['Name', 'Age', 'City', 'Marks']**

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

**10. How do you get the count of all unique values of a categorical column in a DataFrame?**

**We have created a DataFrame df that contains a categorical column named ‘Sex’, and ran value\_counts() function to see the count of each unique value in that column.**

**import pandas as pd**

**data = [['John', 50, 'Male', 'Austin', 70],**

**['Cataline', 45 ,'Female', 'San Francisco', 80],**

**['Matt', 30 ,'Male','Boston', 95]]**

**# Column labels of the DataFrame**

**columns = ['Name','Age','Sex', 'City', 'Marks']**

**# Create a DataFrame df**

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

**df['Sex'].value\_counts()**

**15. How do you read Excel files to CSV using pandas?**

**First, we should use the read\_excel() function to pull in the Excel data to a variable. Then, just apply the to\_csv() function for a seamless conversion.**

**Here is the sample code:**

**import pandas as pd**

**#input your excel file path into the read\_excel() function.**

**excel\_data = pd.read\_excel("/content/sample\_data/california\_housing\_test.xlsx")**

**excel\_data.to\_csv("CSV\_data.csv", index = None, header=True)**

**16. How do you sort a DataFrame based on columns?**

**We have the sort\_values() method to sort the DataFrame based on a single column or multiple columns.**

**import pandas as pd**

**# Create a sample DataFrame**

**data = {**

**'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],**

**'Age': [24, 30, 22, 35, 29],**

**'Score': [85, 95, 80, 75, 90]**

**}**

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

**print("Original DataFrame:")**

**print(df)**

**# Sort by Age in ascending order**

**df\_sorted\_by\_age = df.sort\_values(by='Age')**

**print("\nDataFrame sorted by Age (ascending):")**

**print(df\_sorted\_by\_age)**

**# Sort by Age in descending order**

**df\_sorted\_by\_age\_desc = df.sort\_values(by='Age', ascending=False)**

**print("\nDataFrame sorted by Age (descending):")**

**print(df\_sorted\_by\_age\_desc)**

**# Sort by Age (ascending) and then by Score (descending)**

**df\_sorted\_by\_age\_score = df.sort\_values(by=['Age', 'Score'], ascending=[True, False])**

**print("\nDataFrame sorted by Age (ascending) and Score (descending):")**

**print(df\_sorted\_by\_age\_score)**

**17. Show two different ways to filter data**

**import pandas as pd**

**data = {'Name': ['John', 'Cataline', 'Matt'],**

**'Age': [50, 45, 30],**

**'City': ['Austin', 'San Francisco', 'Boston'],**

**'Marks' : [70, 80, 95]}**

**# Create a DataFrame df**

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

**new\_df = df[(df.Name == "John") | (df.Marks > 90)]**

**print (new\_df)**

**18. How do you aggregate data and apply some aggregation function like mean or sum on it?**

**The groupby function lets you aggregate data based on certain columns and perform operations on the grouped data.**

**In the following code, the data is grouped on the column ‘Name’ and the mean ‘Marks’ of each group is calculated.**

**import pandas as pd**

**data = { 'Name': ['John', 'Matt', 'John', 'Matt', 'Matt', 'Matt'],**

**'Marks': [10, 20, 30, 15, 25, 18]}**

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

**print(df.groupby('Name').mean())**

**19. How can you create a new column derived from existing columns?**

**We can use apply() method to derive a new column by performing some operations on existing columns.**

**import pandas as pd**

**# Create a DataFrame**

**data = {**

**'Name': ['John', 'Matt', 'John', 'Cateline'],**

**'math\_Marks': [18, 20, 19, 15],**

**'science\_Marks': [10, 20, 15, 12]**

**}**

**# Create a DataFrame df**

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

**df['total'] = df.apply(lambda row : row["math\_Marks"] + row["science\_Marks"], axis=1)**

**print(df)**

**20. How do you handle null or missing values in pandas?**

**You can use any of the following three methods to handle missing values in pandas:**

**dropna() – the function removes the missing rows or columns from the DataFrame.**

**fillna() – fill nulls with a specific value using this function.**

**interpolate() – this method fills the missing values with computed interpolation values. The interpolation technique can be linear, polynomial, spline, time, etc.,**

**21. Difference between fillna() and interpolate() methods**

**fillna(0): replaces all missing values with 0.**

**interpolate() fills missing values by interpolating between the existing values. In this case, linear interpolation is used.**

**For column 'B', the value is interpolated between 2 and 3 (since the first and fourth rows are NaN) resulting in 2.5.**

**import pandas as pd**

**import numpy as np**

**# Creating a sample DataFrame with missing values**

**data = {'A': [1, 2, np.nan, 4, 5],**

**'B': [np.nan, 2, 3, np.nan, 5],**

**'C': [1, np.nan, np.nan, 4, 5]}**

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

**print("Original DataFrame:")**

**print(df)**

**# Using fillna() to fill missing values with a specified value, let's say 0**

**filled\_df = df.fillna(0)**

**print("\nDataFrame after filling missing values with 0:")**

**print(filled\_df)**

**# Using interpolate() to fill missing values with interpolated values**

**interpolated\_df = df.interpolate()**

**print("\nDataFrame after interpolating missing values:")**

**print(interpolated\_df)**

**Original DataFrame:**

**A B C**

**0 1.0 NaN 1.0**

**1 2.0 2.0 NaN**

**2 NaN 3.0 NaN**

**3 4.0 NaN 4.0**

**4 5.0 5.0 5.0**

**DataFrame after filling missing values with 0:**

**A B C**

**0 1.0 0.0 1.0**

**1 2.0 2.0 0.0**

**2 0.0 3.0 0.0**

**3 4.0 0.0 4.0**

**4 5.0 5.0 5.0**

**DataFrame after interpolating missing values:**

**A B C**

**0 1.0 NaN 1.0**

**1 2.0 2.0 2.0**

**2 3.0 3.0 3.0**

**3 4.0 4.0 4.0**

**4 5.0 5.0 5.0**

**What is resampling?**

Resampling in pandas involves changing the frequency of your time series data. For instance, you can convert a daily time series to a monthly or yearly frequency,

**import pandas as pd**

**import numpy as np**

**# Creating a sample DataFrame with a daily time series**

**date\_range = pd.date\_range(start='2024-01-01', end='2024-01-10', freq='D')**

**data = {'value': np.random.randint(0, 100, size=len(date\_range))}**

**df = pd.DataFrame(data, index=date\_range)**

**print("Original DataFrame:")**

**print(df)**

**# Resampling to monthly frequency, taking the mean of each month**

**monthly\_resampled = df.resample('ME').mean()**

**print("\nResampled DataFrame with monthly frequency:")**

**print(monthly\_resampled)**

**# Resampling to yearly frequency, summing up the values for each year**

**yearly\_resampled = df.resample('YE').sum()**

**print("\nResampled DataFrame with yearly frequency:")**

**print(yearly\_resampled)**

**Original DataFrame:**

**value**

**2024-01-01 45**

**2024-01-02 82**

**2024-01-03 30**

**2024-01-04 69**

**2024-01-05 20**

**2024-01-06 96**

**2024-01-07 92**

**2024-01-08 56**

**2024-01-09 24**

**2024-01-10 15**

**Resampled DataFrame with monthly frequency:**

**value**

**2024-01-31 51.888889**

**Resampled DataFrame with yearly frequency:**

**value**

**2024-12-31 529**

**23. How do you perform one-hot encoding using pandas?**

**We perform one hot encoding to convert categorical values to numeric ones so that can be fed to the machine learning algorithm.**

**import pandas as pd**

**data = {'Name': ['John', 'Cateline', 'Matt', 'Oliver'],**

**'ID': [1, 22, 23, 36]}**

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

**#one hot encoding**

**new\_df = pd.get\_dummies(df.Name)**

**print(df)**

**print("==============")**

**print(new\_df)**

**24. How do you create a line plot in pandas?**

**To draw a line plot, we have a plot function in pandas.**

**import pandas as pd**

**data = {'units': [1, 2, 3, 4, 5],**

**'price': [7, 12, 8, 13, 16]}**

**# Create a DataFrame df**

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

**df.plot(x='units', y='price')**

**25. What is the pandas method to get the statistical summary of all the columns in a DataFrame?**

**df.describe()**

**This method returns stats like mean, percentile values, min, max, etc., of each column in the DataFrame.**

**Q3. List Key Features of Pandas.**

**Fast and efficient data manipulation and analysis**

**Provides time-series functionality.**

**Easy missing data handling**

**Faster data merging and joining**

**Flexible reshaping and pivoting of data sets.**

**Powerful group by functionality**

**Integrates with NumPy**

**Q5. What are the Different Ways to Create a Series?**

**Creating an Empty Series by just calling the pandas.Series() constructor.**

**import pandas as pd**

**print(pd.Series())**

**Output:**

**Series([], dtype: float64)**

**by using array:**

**import pandas as pd**

**import numpy as np**

**data = np.array(['g', 'e', 'e', 'k', 's'])**

**# convert array to Series**

**print(pd.Series(data))**

**Q13. How to Rename a Column in a DataFrame?**

**df.rename(columns={'A': 'New\_A'}, inplace=True)**