**Pandas Assignment**

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

**Ans:**

* A simple way to store big data sets is to use CSV files (comma separated files).
* CSV files contains plain text and is a well know format that can be read by everyone including Pandas.
* **Example:**

**Load the CSV into a 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?**

**Ans:**

* 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.
* If any column has mixed data types are stored then the data type of the entire column is indicated as object dtype.

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

**Ans:**

Users can select rows based on a particular column value using '>', '=', '<=', '>=', '!=' operators.

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

**Ans:**

**Method 1: Using rename() function**

* One way of renaming the columns in a Pandas Dataframe is by using the rename() function.
* This method is quite useful when we need to rename some selected columns because we need to specify information only for the columns which are to be renamed.

**Method 2: By assigning a list of new column names**

* The columns can also be renamed by directly assigning a list containing the new names to the columns attribute of the Dataframe object for which we want to rename the columns.
* The disadvantage of this method is that we need to provide new names for all the columns even if want to rename only some of the columns.

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

**Ans:**

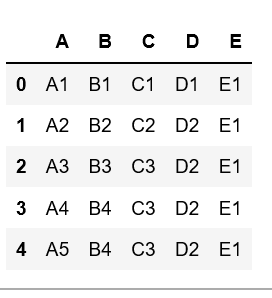
* The drop() method removes the specified row or column.
* By specifying the column axis (axis='columns'), the drop() method removes the specified column.
* By specifying the row axis (axis='index'), the drop() method removes the specified row.
* **Syntax:**

dataframe.drop(labels, axis, index, columns, level, inplace., errors)

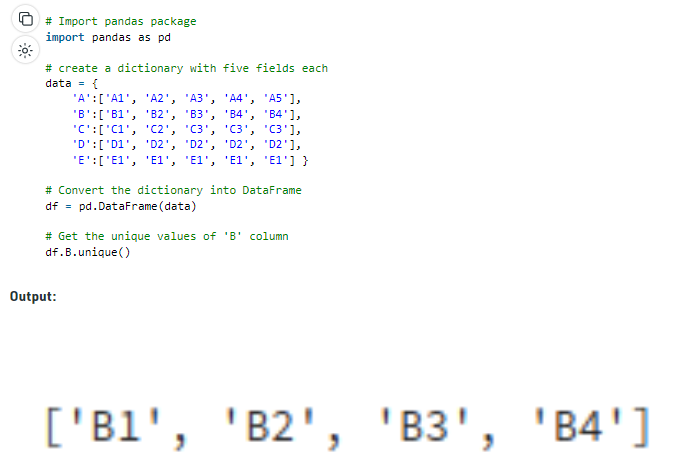
**Q6. How do you find the unique values in a column of a Pandas DataFrame?**

**Ans:**

* Let’s discuss how to get unique values from a column in Pandas DataFrame.
* Create a simple dataframe with dictionary of lists, say columns name are A, B, C, D, E with duplicate elements.



* Now, let’s get the unique values of a column in this dataframe.
* **Example #1:** Get the unique values of ‘B’ column



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

**Ans:**

Count NaN values under an entire DataFrame: *df.isna().sum().sum()*

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

**Ans:**

* The fillna() method replaces the NULL values with a specified value.
* The fillna() method returns a new DataFrame object unless the inplace parameter is set to True, in that case the fillna() method does the replacing in the original DataFrame instead.

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

**Ans:**

* The concat() function in pandas is used to append either columns or rows from one DataFrame to another.
* The concat() function does all the heavy lifting of performing concatenation operations along an axis while performing optional set logic (union or intersection) of the indexes (if any) on the other axes.
* **Example:**

*df1 = pd.DataFrame({'id': ['A01', 'A02', 'A03', 'A04'],*

*'Name': ['ABC', 'PQR', 'DEF', 'GHI']})*

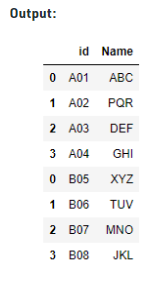
*df2 = pd.DataFrame({'id': ['B05', 'B06', 'B07', 'B08'],*

*'Name': ['XYZ', 'TUV', 'MNO', 'JKL']})*

*frames = [df1, df2]*

*result = pd.concat(frames)*

*display(result)*



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

**Ans:**

We can merge two Pandas DataFrames on certain columns using the merge function by simply specifying the certain columns for merge.

***Syntax:***

*DataFrame.merge(right, how=’inner’, on=None, left\_on=None, right\_on=None, left\_index=False, right\_index=False, sort=False, copy=True, indicator=False, validate=None)*

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

**Ans:**

* Python is a great language for doing data analysis, primarily because of the fantastic ecosystem of data-centric Python packages.
* *Pandas*is one of those packages and makes importing and analyzing data much easier.
* Dataframe.aggregate() function is used to apply some aggregation across one or more column.
* Aggregate using callable, string, dict, or list of string/callables. Most frequently used aggregations are:
* **sum:** Return the sum of the values for the requested axis  
  **min:** Return the minimum of the values for the requested axis  
  **max:** Return the maximum of the values for the requested axis
* ***Syntax:****DataFrame.aggregate(func, axis=0, \*args, \*\*kwargs)*

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

**Ans:**

* The pivot() function is used to reshaped a given DataFrame organized by given index / column values.
* This function does not support data aggregation, multiple values will result in a MultiIndex in the columns.
* **Syntax:**

DataFrame.pivot(self, index=None, columns=None, values=None)

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

**Ans:**

***Change column type into string object using DataFrame.astype():***

* DataFrame.astype() method is used to cast pandas object to a specified dtype.
* This function also provides the capability to convert any suitable existing column to a categorical type.

***Change column type in pandas using dictionary and DataFrame.astype():***

* We can pass any Python, Numpy, or Pandas datatype to change all columns of a Dataframe to that type.
* We can pass a dictionary having column names as keys and datatype as values to change the type of selected columns.

***Change column type in pandas using DataFrame.apply()***

* We can pass pandas.to\_numeric, pandas.to\_datetime, and pandas.to\_timedelta as arguments to apply the apply() function to change the data type of one or more columns to numeric, DateTime, and time delta respectively.

***Change column type in pandas using DataFrame.infer\_objects()***

* This DataFrame.infer\_objects() method attempts soft-conversion by inferring the data type of ‘object’-type columns. Non-object and unconvertible columns are left unchanged.

***Change column type in pandas using convert\_dtypes()***

* A new DataFrame with each column’s data type changed to the best one is returned by the convert dtypes() method.

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

**Ans:**

* To sort the DataFrame based on the values in a single column, you’ll use .sort\_values().
* By default, this will return a new DataFrame sorted in ascending order. It does not modify the original DataFrame.

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

**Ans:**

* The copy() method returns a copy of the DataFrame.
* By default, the copy is a "deep copy" meaning that any changes made in the original DataFrame will NOT be reflected in the copy.
* **Syntax:**

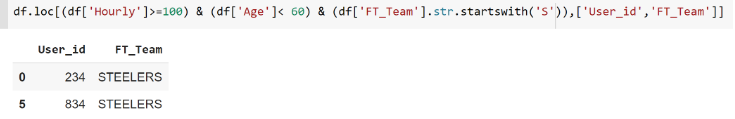
dataframe.copy(deep)

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

**Ans:**

**Using Loc to Filter With Multiple Conditions**

* The loc function in pandas can be used to access groups of rows or columns by label.
* Add each condition you want to be included in the filtered result and concatenate them with the & operator.
* You'll see our code sample will return a pd.dataframe of our filtered rows.
* Don't forget to include "import pandas as pd" at the top!



**Using The Numpy Function np.where To Filter With Multiple Conditions**

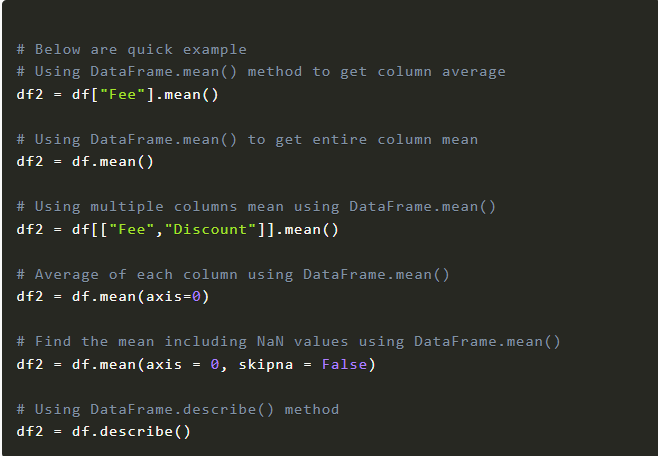
* The Numpy python library interacts great with dataframes, especially when dealing with indexing.
* Let's use the numpy.where function with a few conditions to produce the same result we wanted above.



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

**Ans:**

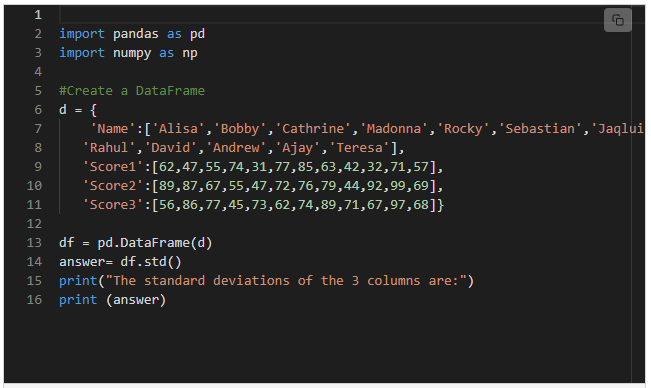
* To get column average or mean from pandas DataFrame use either mean() and describe() method.
* The DataFrame.mean() method is used to return the mean of the values for the requested axis.
* If you apply this method on a series object, then it returns a scalar value, which is the mean value of all the observations in the pandas DataFrame.



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

**Ans:**

* Standard deviation is the measure of how spread out numbers are. Pandas is a library in Python that is used to calculate the ​standard deviation.
* Standard deviation is calculated using the function.std(). However, the Pandas library creates the Dataframe object and then the function.std() is applied on that Dataframe.
* The following code calculates the standard deviation of three columns (i.e., *Score1*, *Score2*, and *Score3*).



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

**Ans:**

* 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.
* By using [corr()](https://www.geeksforgeeks.org/python-pandas-dataframe-corr/) function we can get the correlation between two columns in the dataframe.
* **Syntax:**

*dataframe[‘first\_column’].corr(dataframe[‘second\_column’])*

*where,*

*dataframe is the input dataframe*

*first\_column is correlated with second\_column of the dataframe*

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

**Ans:**

**Selecting columns based on their name**

This is the most basic way to select a single column from a dataframe, just put the string name of the column in brackets. Returns a pandas series.

df['hue']

Passing a list in the brackets lets you select multiple columns at the same time.

df[['alcohol','hue']]

Selecting a subset of columns found in a list

Similar to the previous example, but here you can search over all the columns in the dataframe.

df[df.columns[df.columns.isin(['alcohol','hue','NON-EXISTANT COLUMN'])]]

**Selecting a subset of columns based on difference of columns**

Let’s say you know what columns you *don’t*want in the dataframe. Pass those as a list to the *difference* method and you’ll get back everything except them.

df[df.columns.difference([‘alcohol’,’hue’])]

**Selecting a subset of columns that is not in a list**

Return a data frame that has columns that are not in a list that you want to search over.

df[df.columns[~df.columns.isin(['alcohol','hue'])]]

**Selecting columns based on their data type**

Data types include ‘float64’ and ‘object’ and are inferred from the columns passed to the *dtypes* method. By matching on columns that are the same data type, you’ll get a series of True/False. Use the *values* method to get just the True/False values and not the index.

df.loc[:,(df.dtypes=='float64').values]

**Selecting columns based on their column name containing a substring**

If you have tons of columns in a data frame and their column names all have a similar substring that you are interested in, you can return the columns who’s names contain a substring. Here we want everything that has the “al” substring in it.

df.loc[:,['al' in i for i in df.columns]]

**Selecting columns based on their column name containing a string wildcard**

You could have hundreds of columns, so it might make sense to find columns that match a pattern. Searching for column names that match a wildcard can be done with the “search” function from the *re* package (see the link in the reference section for more details on using the regular expression package).

df.loc[:,[True if re.search('flava+',column) else False for column in df.columns]]

**Selecting columns based on how their column name starts**

If you want to select columns with names that start with a certain string, you can use the *startswith* method and pass it in the columns spot for the data frame location.

df.loc[:,df.columns.str.startswith('al')]

**Selecting columns based on how their column name ends**

Same as the last example, but finds columns with names that end a certain way.

df.loc[:,df.columns.str.endswith('oids')]

**Selecting columns if all rows meet a condition**

You can pick columns if the rows meet a condition. Here, if all the the values in a column is greater than 14, we return the column from the data frame.

df.loc[:,[(df[col] > 14).all() for col in df.columns]]

**Selecting columns if any row of a column meets a condition**

Here, if any of the the values in a column is greater than 14, we return the column from the data frame.

df.loc[:,[(df[col] > 14).any() for col in df.columns]]

**Selecting columns if the average of rows in a column meet a condition**

Here, if the mean of all the values in a column meet a condition, return the column.

df.loc[:,[(df[col].mean() > 7) for col in df.columns]]

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

**Ans:**

You can select a single row from pandas DataFrame by integer index using df.iloc[n]. Replace n with a position you wanted to select.

# Select Row by Integer Index

print(df.iloc[2])

# Outputs

#Courses Hadoop

#Fee 26000

#Duration 35days

#Discount 1500

#Name: r3, dtype: object

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

**Ans:**

* Pandas sort\_values() method sorts a data frame in Ascending or Descending order of passed Column.
* It’s different than the sorted Python function since it cannot sort a data frame and particular column cannot be selected.
* ***Syntax:***

*DataFrame.sort\_values(by,axis=0,ascending=True,inplace=False,kind=’quicksort’, na\_position=’last’)*

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

**Ans:**

Given a Dataframe containing data about an event, we would like to create a new column called *‘Discounted\_Price’*, which is calculated after applying a discount of 10% on the Ticket price.

**Example:** We can use DataFrame.apply() function to achieve this task.

*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(df)*



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

**Ans:**

* The drop\_duplicates() method removes duplicate rows.
* Use the subset parameter if only some specified columns should be considered when looking for duplicates.
* **Syntax**

*dataframe*.drop\_duplicates(subset, keep, inplace, ignore\_index)

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

