PANDAS

Diagram

Description automatically generated with medium confidence

In a series, the axis labels are called index.

Series can only contain a single list with index, whereas the DataFrames can be made of more than one series.

* **DataFrame**(): Two-dimensional, size-mutable, potentially heterogeneous tabular data. Data structure also contains labeled axes (rows and columns). The primary pandas data structure
* .**Series**(): One-dimensional ndarray with axis labels (including time series).
* **.sort\_index :**Sort Series by index labels
* **.sort\_values :**Sort a Series in ascending or descending order by the values
* **.isin :**Return a boolean Series showing whether each element in the Series matches an element in the passed sequence of *values* exactly
* **.keys :**return the index labels of the given series object
* **.values :**Return Series as ndarray or ndarray-like depending on the dtype
* **.items :**This method returns an iterable tuple (index, value)
* .**read\_csv**(): Read a comma-separated values (csv) file into DataFrame
* .**head**(): This function returns the first *n* rows for the object based on position, default n=5
* .**tail**(): This function returns last *n* rows from the object based on position, default n=5.
* **.sample :**Return a random sample of items from an axis of object
* .**shape**: Return a tuple representing the dimensionality of the DataFrame.
* **df.index :** The basic object storing axis labels for all pandas objects.
* **df.columns :** The column labels of the DataFrame.
* **df.shape :** Return a tuple representing the dimensionality of the DataFrame.
* **df.size :** Return an int representing the number of elements in this object.
* **df.ndim :** Return an int representing the number of axes / array dimensions.
* **df.reset\_index :** Reset the index of the DataFrame, and use the default one instead. If the DataFrame has a MultiIndex, this method can remove one or more levels.
* **df.set\_index():** Set the DataFrame index using existing columns.
* **df["col"]:**You can pass a list of columns to [] to select columns in that order. If a column is not contained in the DataFrame, an exception will be raised.
* **df.col:**You may access a column on a dataframe directly as an attribute:
* **df.iloc[]:**Purely integer-location based indexing for selection by position.
* **df.loc[]:**Access a group of rows and columns by label(s) or a boolean array
* **df.xs :** This method takes a *key* argument to select data at a particular level of a MultiIndex.
* **Conditional Indexing:**The condition inside the selection brackets df[ df["Age"]>35 ] checks for which rows the Age column has a value larger than 35:

Data Frame Basics-3 (Properties)

In this lesson, you will continue with DataFrame column operations in Pandas and you will learn:

* df.info(): This method prints information about a DataFrame including the index dtype and columns, non-null values and memory usage
* df.describe(): Generate descriptive statistics which include those that summarize the central tendency, dispersion and shape of a dataset’s distribution, excluding NaN values
* df.value\_counts(): Return a Series containing counts of unique values.
* df.mean(): Return the mean of the values over the requested axis.
* df.sum(): Return the sum of the values over the requested axis.
* df.unique(): Hash table-based unique. Uniques are returned in order of appearance. This does NOT sort
* df.isnull(): Detect missing values. Return a boolean same-sized object indicating if the values are NA.
* len(df): Returns the length (number of characters) in a string. Returns the number of entries for dictionaries, lists or tuples.
* df.shape: Return a tuple representing the dimensionality of the DataFrame.
* df.drop():Remove rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names

df = pd.DataFrame(np.arange(12).reshape(3, 4), columns=['A', 'B', 'C', 'D'])

df

A B C D

0 0 1 2 3

1 4 5 6 7

2 8 9 10 11

df.drop(['B', 'C'], axis=1)

A D

0 0 3

1 4 7

2 8 11

### Groupby

Groupby operation involves some of the following operations on the original object.

* **Splitting** the Object
* **Applying** a function
* **Combining** the results

Groupby method is used with aggregation functions such as:

* mean,
* standard deviation,
* max and min,
* count.
* sum.

While applying a function one of the following operations is used.

* **Aggregation** − Computes a summary statistic. Apply multiple functions to a column or many columns. At the end a different dataframe whose length is the number of unique values of the groupby keys is extracted.
* **Transformation** − Performs some group-specific operation. Broadcast results of sub dataframes to original dataframe. It will always return a series with the same length to the original dataframe
* **Filtering** − Discards the data with some condition. It applies a filter on the results obtained from those sub dataframes. The filtered results will then be broadcast to any matching conditions in the original dataframe. In this situation, you'll obtain a condensed version of the full dataframe.

**Pivot Table & Stack**

Relationship between features can be retrieved via bivariate analysis . In order to extract their relationship with numeric and other categorical features, categorical features uses groupby and apply functions. Pivot tables and Stack/Unstack functions are also extremely useful in this context.

**Pivot Table**

     Create a spreadsheet-style pivot table as a DataFrame. It accepts three arguments; index, columns, and values. Dataframe's categorical features can be passed in the index and columns. New table's cell values are taken from a column specified by the values parameter.

**Stack/Unstack**

When you stack a DataFrame, the innermost column index becomes the innermost row index. Unstacking is the inverse operation.

### Useful Methods

In this lesson, you will learn apply function and some dataframe measurement methods :

* ***apply()*** : Apply a function along an axis of the DataFrame. Series in given axis are passed to the function.
* Row or Column Wise Function Application: **apply()**
* Element wise Function Application: **applymap()**
* Function application on Series :**map()**
* Dealing with unique values in a column using;
* unique() :Compute array of unique values in a Series, returned in the order observed
* nunique(): gives the number of unique values
* value\_counts() :Return a Series containing unique values as its index and frequencies as its values, ordered count in descending order
* sort\_values() method Sorting a DataFrame by a column. It accepts a 'by' argument with column name.