**Pandas-2**

Mark as done

Groupby

 method is used with aggregation functions such as:

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

.

**Groupby**

Groupby

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

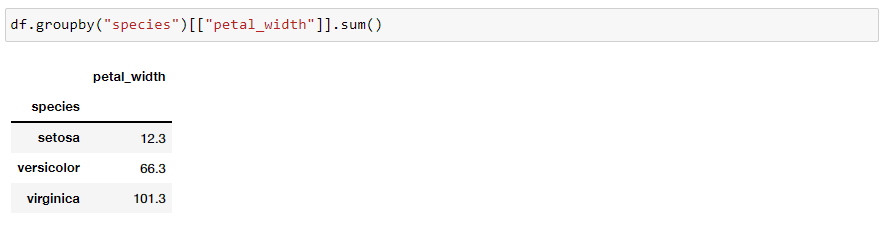
* **Splitting** the Object
* **Applying** a function
* **Combining** the results



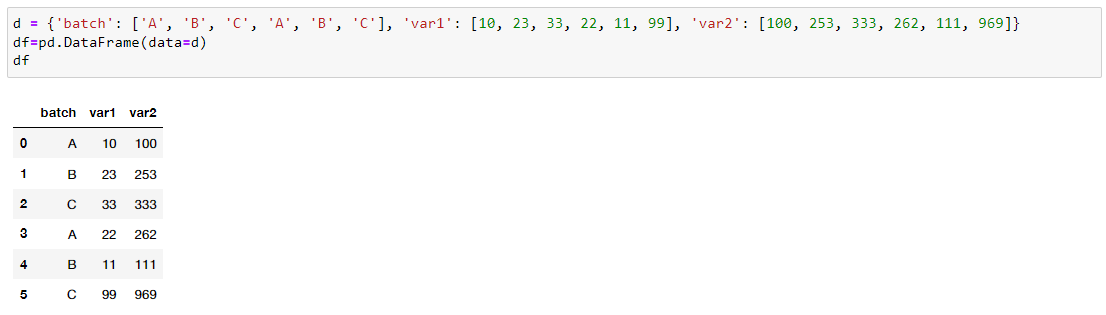
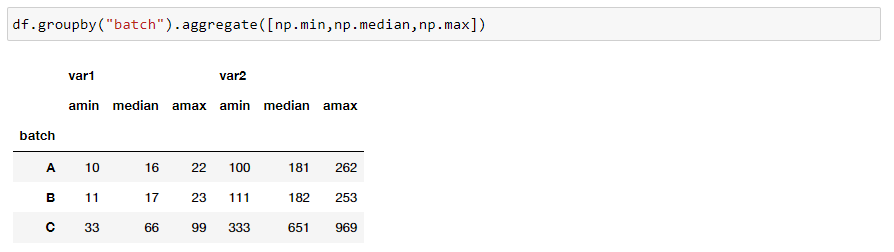
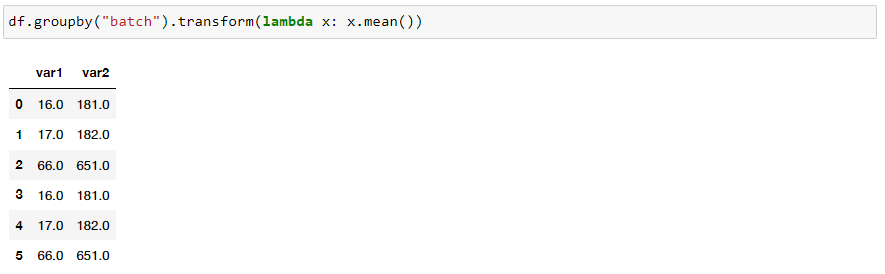
Groupby

 method is used with aggregation functions such as:

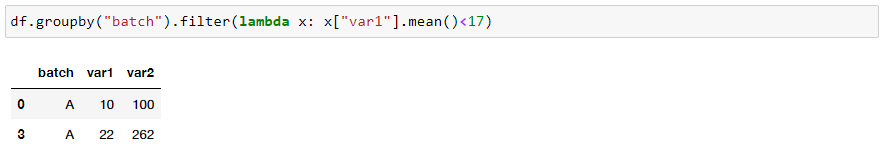
* mea,
* standard deviation,
* max and min,
* count.
* sum

. 

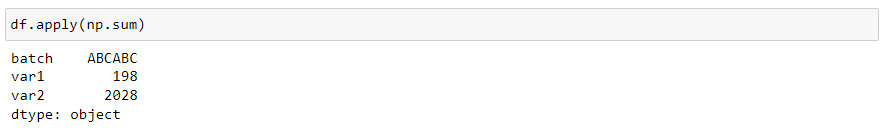
[When should I use a "groupby" in pandas?](https://www.youtube.com/watch?v=qy0fDqoMJx8)

While applying a function one of the following operations is used. **In the following examples new df below 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.



* **Apply :**Only allows a function. Produce aggregated results.



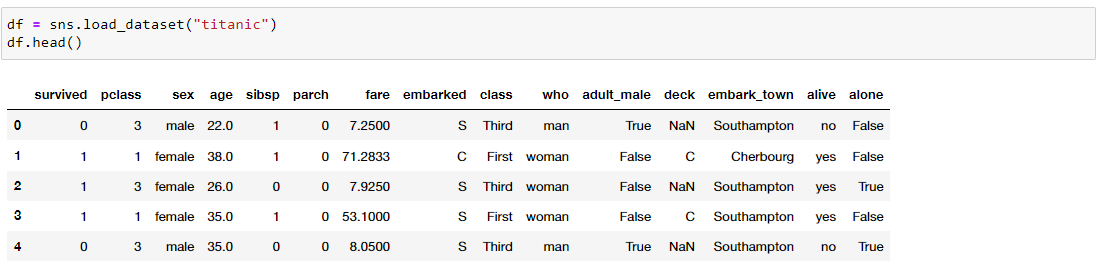
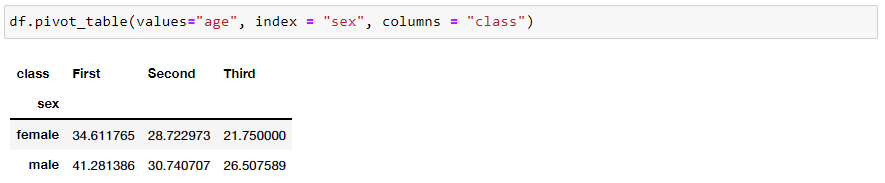
[Advanced Use of groupby(), aggregate, filter, transform, apply - Beginner Python Pandas Tutorial #5](https://www.youtube.com/watch?v=DUgd48QYmfI)

**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.

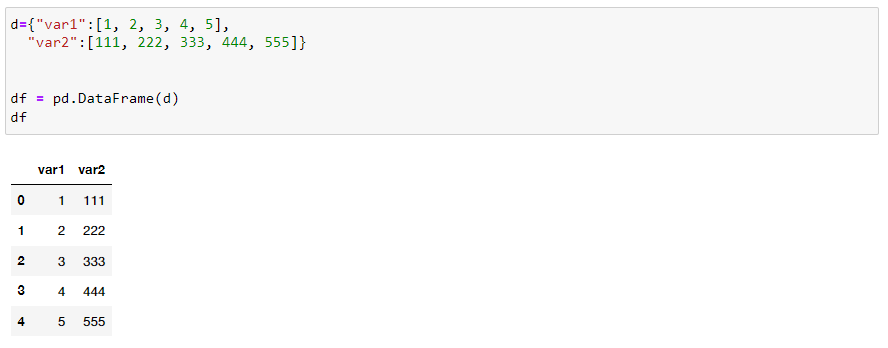
* **Stack/Unstack**

When you stack a DataFrame, the innermost column index becomes the innermost row index. Unstacking is the inverse operation.  
  
[Python Pandas Tutorial 10. Pivot table](https://www.youtube.com/watch?v=xPPs59pn6qU)

* **Stack/Unstack**

When you stack a DataFrame, the innermost column index becomes the innermost row index. Unstacking is the inverse operation.  
  
  
  
  
  
  
  
[Python Pandas Tutorial 12. Stack Unstack](https://www.youtube.com/watch?v=BUOy4RUUepg)

**Useful Methods**

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


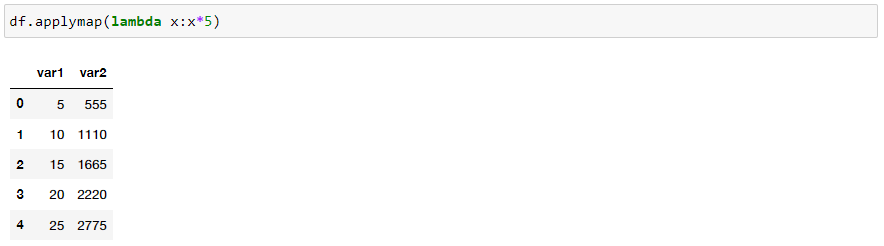
<em><strong>apply()</strong></em>

*:* Apply a function along an axis of the DataFrame. Series in given axis are passed to the function.

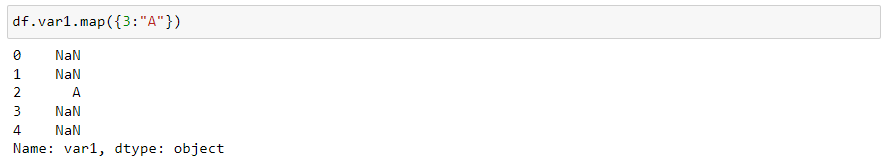
* Row or Column Wise Function Application: **apply()**



* Element wise Function Application: **applymap()**



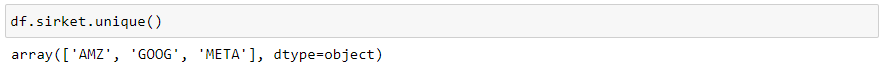
* **Function application on Series : map()**



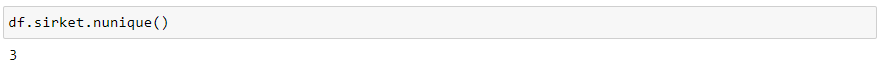
* **Dealing with unique values in a column using;**

****

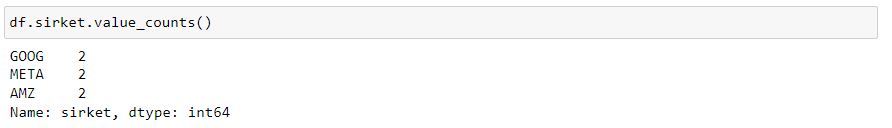
* **unique() :Compute array of unique values in a Series, returned in the order observed**

****

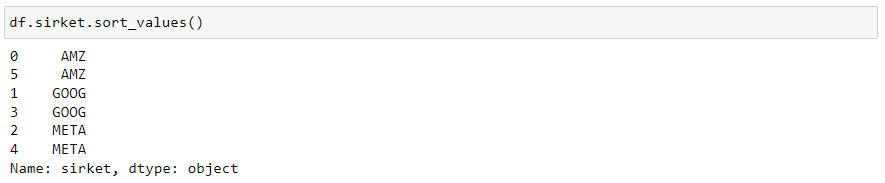
* **nunique(): gives the number of unique values**

****

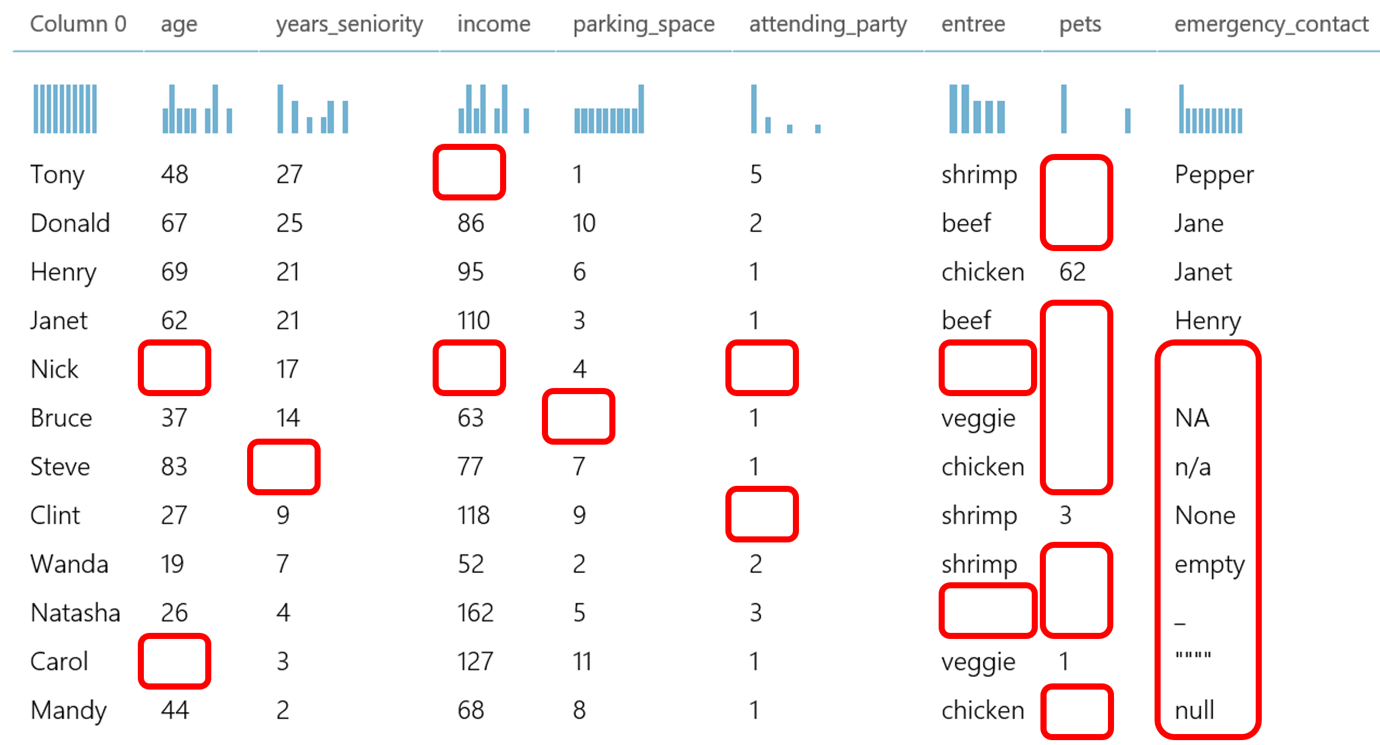
* **value\_counts() :Return a Series containing unique values as its index and frequencies as its values, ordered count in descending order**

****

* **sort\_values()method Sorting a DataFrame by a column. It accepts a 'by' argument with column name.**

  
  
  
[How do I apply a function to a pandas Series or DataFrame?](https://www.youtube.com/watch?v=P_q0tkYqvSk)

**Missing Data**

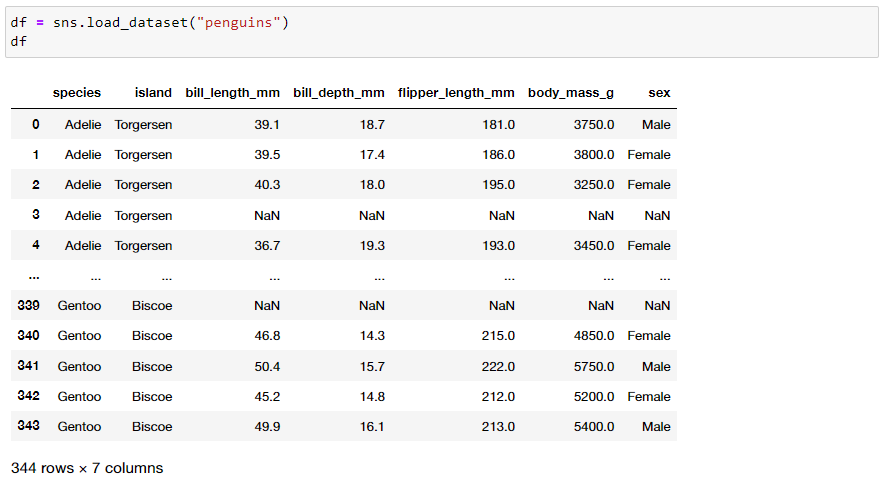


In real-world circumstances, missing data is always an issue. Machine learning and data mining confront significant challenges in terms of model prediction accuracy due to poor data quality caused by missing values. Missing value treatment is a primary focus in these domains to improve the accuracy and validity of their models.

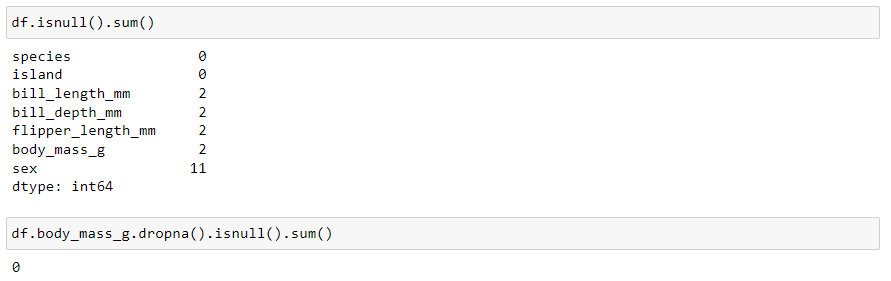
Missing data occurs because of variety of reasons, including manual data entry techniques, equipment faults, and wrong measurements. In a dataset **NaN** means **Not a Number.**Pandas provides the isnull() and notnull() functions to detect missing data.

In this lesson you will learn:

* How to handle missing values in a DataFrame,



* Dropna method drops the missing values along with the **axis** argument,



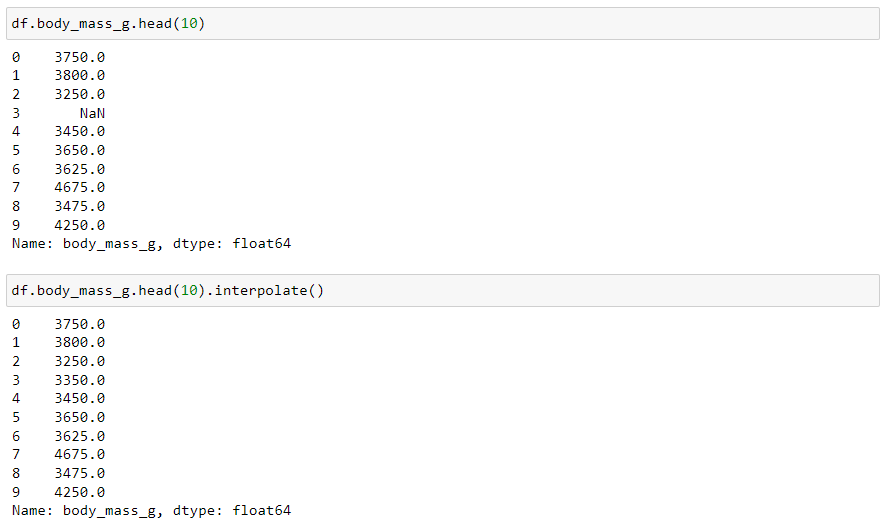
* Fillna  method can “fill in” NA values with non-null data



* replacemethod similar to fillna “fill in” NA values with non-null values.

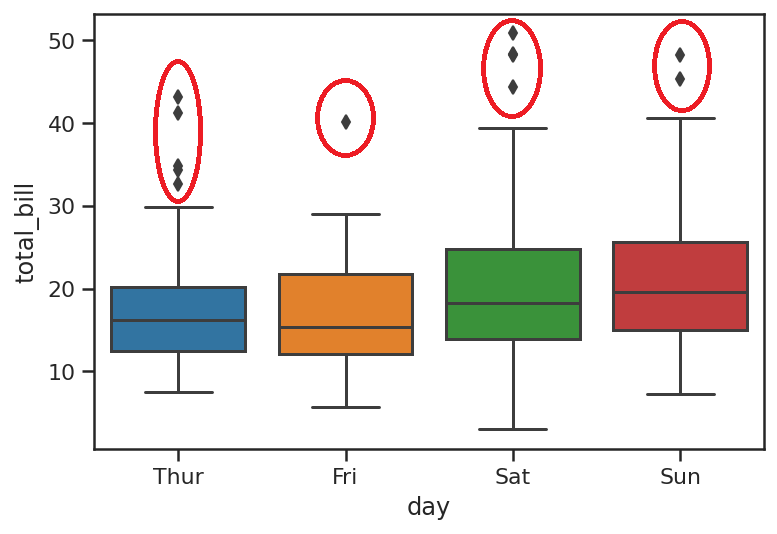


* interpolatemethod uses various interpolation technique to fill the missing values.



[Handling Missing Values in Pandas Dataframe | GeeksforGeeks](https://www.youtube.com/watch?v=uDr67HBIPz8)

**Outliers**



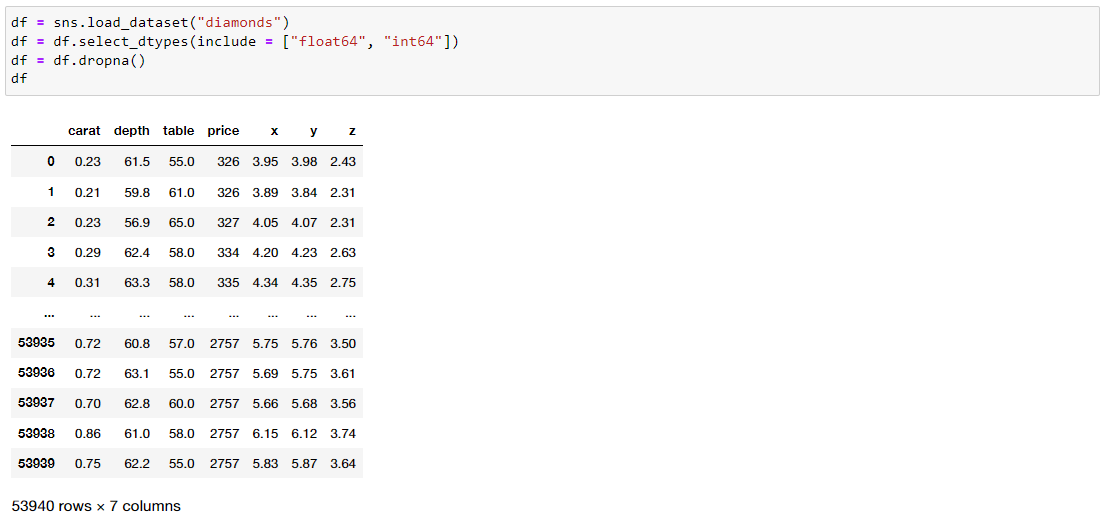
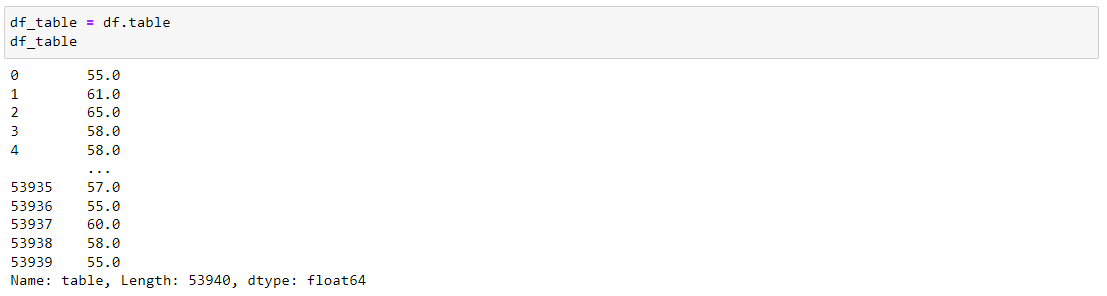
During EDA phase some preventions are to be applied to data so that in the future Machine Learning modelling process will not be affected. Alongside with missing data handling next operation should be handling outliers in our data.

*An****outlier****is an observation point that is distant from other observations*according to the Wikipedia.

There can be many different reasons which result some values in our dataset to become an outlier like data collection mistakes or variance in the dataset. The image above shows the outlier values in a dataframes table column. Black dots on both side of the boxplot indicates the outlier values.

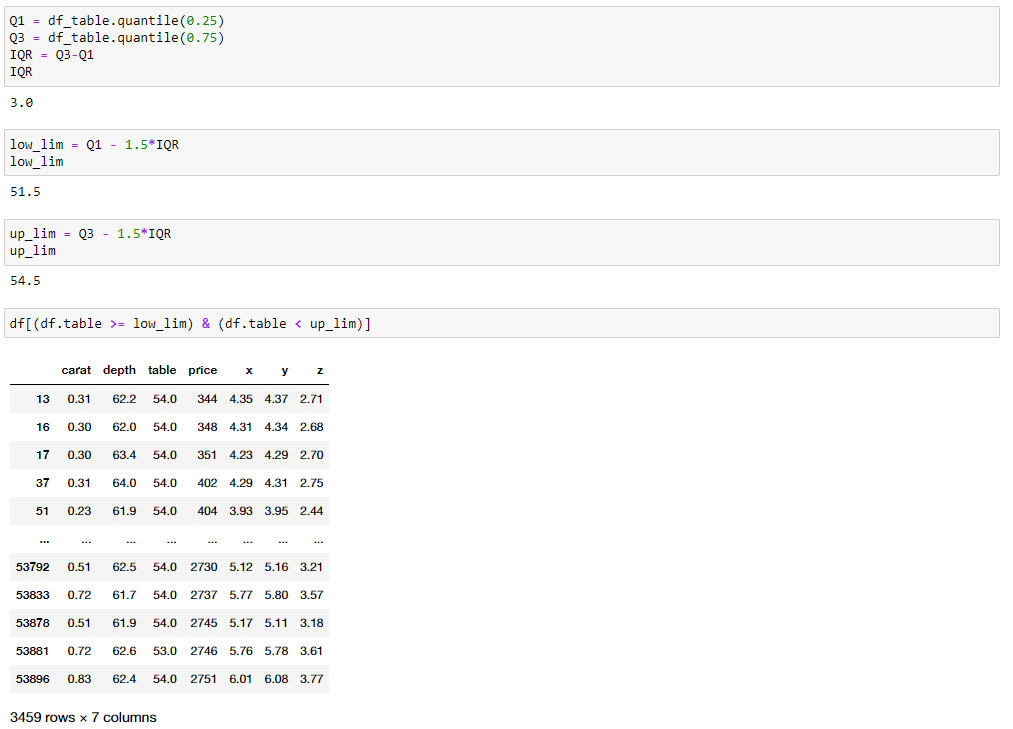
To be able to find the outliers IQR method is used. From Wikipedia;

*The****interquartile range****(****IQR****), also called the****midspread****or****middle 50%****, or technically****H-spread****, is a measure of statistical dispersion, being equal to the difference between 75th and 25th percentiles, or between upper and lower quartiles, IQR = Q3 − Q1.*

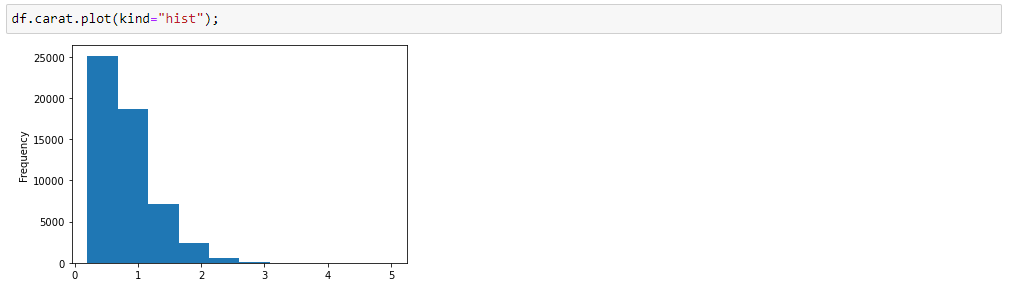
Mainly there are two way to get rid of these outliers.

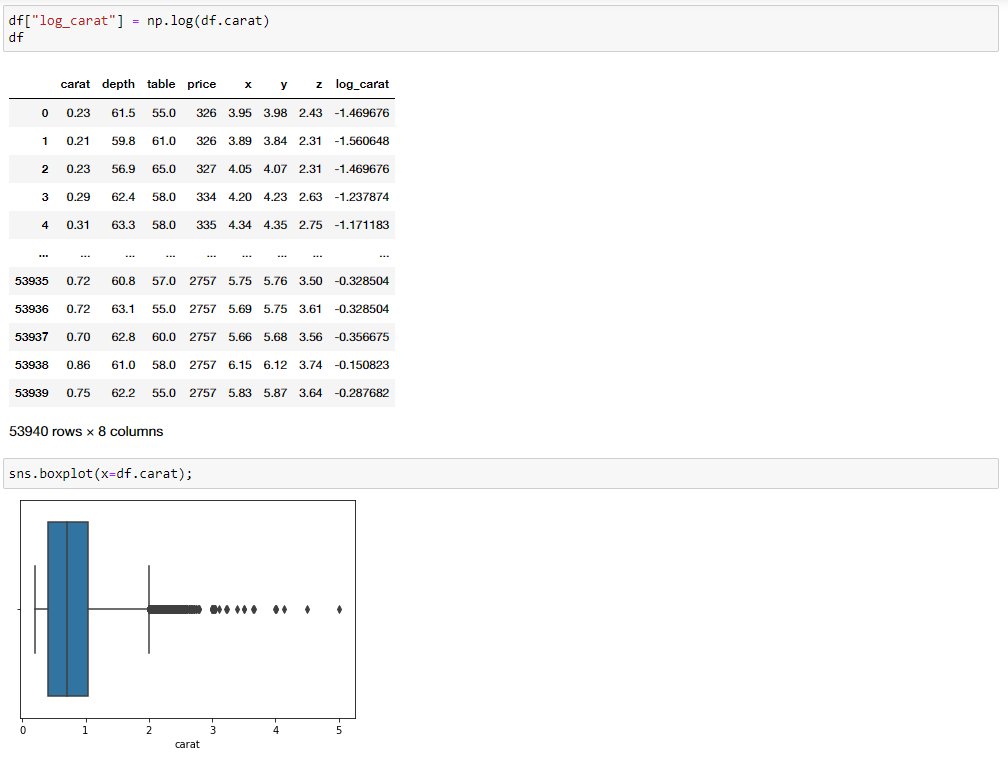
1. **Removing** the outliers with **IQR method**

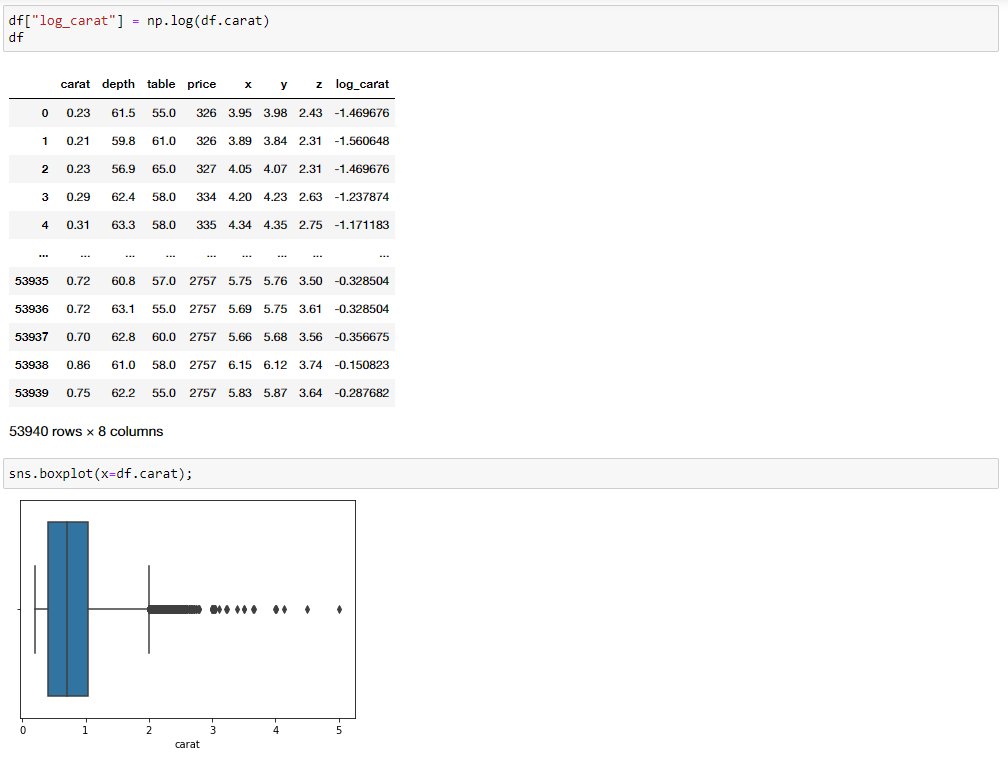
****

**2. Transforming them into non-outlier values with Winsorize method**

****

**Log transformation** method  








In this lesson you will learn:

* How to handle outliers in a DataFrame,
* log transformation

is used on skewed data to reduce the skewness to make it normally distributed.

* winsorize(<em>from scipy.stats.mstats import winsorize</em>)The (limits[0])th lowest values are set to the (limits[0])th percentile, and the (limits[1])th highest values are set to the (1 - limits[1])th percentile

[Outlier detection and removal using IQR | Feature engineering tutorial python # 4](https://www.youtube.com/watch?v=A3gClkblXK8)

**Combining DataFrames (Merging-Joining and Concatenating)**

In this lesson, you will learn :

* Merge;

merges DataFrame or named Series objects with a database-style join like SQL.  It is used to combine two or more dataframes on the basis of values of common columns.

left = pd.DataFrame({'key': ['K0', 'K1', 'K2'],  'A': ['A0', 'A1', 'A2'], 'B': ['B0', 'B1', 'B2']})

right = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'], 'C': ['C0', 'C1', 'C2', 'C3'], 'D': ['D0', 'D1', 'D2', 'D3']})

* Join;

joins columns with *other* DataFrame either on index or on a key column. It is used to merge 2 dataframes on the basis of the index; instead of using merge(left\_index=True) we can use join().

left = pd.DataFrame({'A': ['A0', 'A1', 'A2'], 'B': ['B0', 'B1', 'B2']}, index = ['K0', 'K1', 'K2'])

right = pd.DataFrame({'C': ['C0', 'C2', 'C3'], 'D': ['D0', 'D2', 'D3']},index = ['K0', 'K2', 'K3'])

* Concat;

 Concatenate pandas objects along a particular axis. It is a kind of appending dataframes one over another or one next to another.

df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],  'B': ['B0', 'B1', 'B2', 'B3'], 'C': ['C0', 'C1', 'C2', 'C3'], 'D': ['D0', 'D1', 'D2', 'D3']})

df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],  'B': ['B4', 'B5', 'B6', 'B7'], 'C': ['C4', 'C5', 'C6', 'C7'], 'D': ['D4', 'D5', 'D6', 'D7']})

df3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'], 'B': ['B8', 'B9', 'B10', 'B11'], 'C': ['C8', 'C9', 'C10', 'C11'], 'D': ['D8', 'D9', 'D10', 'D11']})

In Pandas merge() and join() are used for a horizontal combination,  whereas concat() and append() are used for vertical combination.

**Text and Time Methods**

Many pieces of data are in text form rather than pure numbers. This means that string must be cleaned and preprocessed before it can be examined, processed by algorithms, or displayed to the public. Fortunately, pandas library has a section dedicated to string processing that make it simple to work with string data.

Series and Index are equipped with a set of string processing methods that make it easy to operate on each element of the array. Perhaps most importantly, these methods exclude missing/NA values automatically. These are accessed via the

str

 attribute.

* lower()

: Converts strings to lower case

* upper()

 : Converts strings to upper case.

* islower() :

 Checks whether all characters in each string in lower case or not. Returns Boolean

* isupper() :

 Checks whether all characters in each string in upper case or not. Returns Boolean

* isdigit()

 : Check whether all characters in each string are digits.

* isnumeric()

: Checks whether all characters in each string are numeric. Returns Boolean.

* replace():

 Replaces the value **a** with the value **b**

* **contains():**

**Returns a Boolean value True for each element if the substring contains in the element, else False.**



**split()**

**: Splits each string with the given pattern**

* **strip():**

**Helps strip whitespace(including newline) from each string**

* **find() :**

**Returns the first position of the first occurrence of the pattern**

* **findall() :**

**Returns a list of all occurrence of the pattern.**

**Time Methods**

Time-series data is currently used in a wide range of industries for time series forecasting, seasonality analysis,trend detection, and making critical business and research choices. As a result, it is critical for a data scientist or data analyst to accurately understand time series data.

* datetime

 module contains the core objects for working with time series data in Python.

* to\_datetime()

 method parses many different kinds of date representations returning a

Timestamp

 object.

* strftime

 - convert object to a string according to a given format

* strptime

 - parse a string into a

datetime

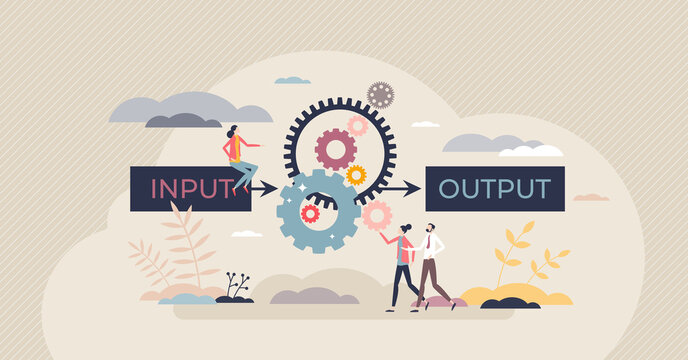
 object given a corresponding format

* timedelta

-  gives time difference

**Get\_Dummies Method**Since Machine Learning models does not work with categorical/non-numerical data we have to convert categorical data to numerical data before feeding it into a model. Get\_Dummies method basically converts categorical variable into dummy/indicator variables.

**Data Input and Output**



**👨‍🏫 Note:**

* **This part of Pandas pre-class is complimentary since we will read almost all kinds of files throughout our course and be familiar with how to read and save various files with different extensions. So, the instructors do NOT particularly lecture this part in our class sessions.**

**In this lesson, you will learn;**

* **read\_csv() : Read a comma-separated values (csv) file into DataFrame.**

* **DataFrame.to\_csv() :Write DataFrame to a comma-separated values (csv) file.**

* **read\_excel(): Read an Excel file into a pandas DataFrame.**

* **DataFrame.to\_excel(): Write object to an Excel sheet.**

* **read\_html()**: Read HTML tables into a

list

 of

DataFrame

 objects. Most of the time you don't need to save your file as HTML so we pass that.

**SQL files**

      Reading data with read\_sql() from a relational database is not possible since loading all the data into memory at once overloads your memory. So instead of that, you can first create a database with the help of sqlalchemy library--> create\_engine method. Then you can import data from any source to this database with  to\_sql(). Finally, you can either use read\_sql() command to read the dataset from SQL database or read\_sql\_query() command to make a unique query from the database.

**💡Tips:**

* The main purpose of this lesson is to learn how to read the

csv

 files. Because we will work with

csv

 files in the next part of the course. If this lesson is too complicated, for now, you can only try to learn how to read the

csv

 files.

**Useful Pandas Tutorials**

**DataFrame and Series Basics**

In this section, you will find additional lectures that will help you reinforce what you have learned before. Throughout the video series, the instructor will use the same two data sets (Stack Overflow Annual Developer Survey-2019). You can find the link for the data sets [here](https://insights.stackoverflow.com/survey). We recommend that you download the datasets to your computer and execute the same commands simultaneously.

**Useful Pandas Tutorials**

**Indexes**

**Useful Pandas Tutorials**

**Filtering-Using Conditionals**

**Useful Pandas Tutorials**

**Updating Rows and Columns**

**Useful Pandas Tutorials**

**Add/Remove Rows and Columns**

**Useful Pandas Tutorials**

**Sorting Data**

**Useful Pandas Tutorials**

**Grouping and Aggregating**

**Useful Pandas Tutorials**

**Missing Values**

**Useful Pandas Tutorials**

**Reading & Writing Data**