Analytics using Python

Learning outcomes

1. You will learn Python , a useful language

2. Use programming for problem solving

Great Lakes Institute of Management

A guide to learn python for analytics

P. V. Subramanian

**A workbook on Analytics using Python**

**P. V. SUBRAMANIAN**

**Visiting faculty – Great Lakes Institute of Management**

*Copyright @ 2019 Great Lakes Institute of Management : All contents in this book, including (but not limited to) its contents are proprietary to Great Lakes Institute of Management, Chennai. Any effort at re-using, modifying and circulating both through Digital or Physical means, without the express written permission of the author, will amount to infringement of copyrights, and will be liable for suitable action.*

**Chapter 4. Data Manipulation using Pandas**

**Table of contents**

Table of Contents

[**1.** **Plotting using pandas series and data frames** 6](#_Toc8222984)

[**2.** **Data Manipulation** 7](#_Toc8222985)

[**2.1** Creation of new variable 8](#_Toc8222988)

[2.1.1 A column derived from the existing column(s) to get useful insights 9](#_Toc8222989)

[2.1.2 Dummy variables 12](#_Toc8222990)

[2.1.3 Combining variables & levels to reduce the dimensionality 13](#_Toc8222991)

[**2.2** **Aggregate** 15](#_Toc8222992)

[2.2.1 Groupby operations 15](#_Toc8222993)

[2.2.2 Hierarchical indexing 16](#_Toc8222994)

[2.2.3 Pivot table 20](#_Toc8222995)

[2.2.4 Crosstab 22](#_Toc8222996)

[**2.3** **Sort** 24](#_Toc8222997)

[**2.4** **Merge** 30](#_Toc8222998)

[2.4.1 Merge two data frames when key names are same 31](#_Toc8222999)

[2.4.2 Merge two data frames when key names are different 32](#_Toc8223000)

[2.4.3 Merge function arguments 33](#_Toc8223001)

[2.4.4 Merging on index 34](#_Toc8223002)

[2.4.5 Join 35](#_Toc8223003)

[**2.5** **Append** 36](#_Toc8223004)

[2.5.1 a. concat function 36](#_Toc8223005)

[2.5.2 1.5.2 append functionfunction 40](#_Toc8223006)

[3. **Case study** 44](#_Toc8223007)

[**3.1** Titanic Data set 44](#_Toc8223009)

[3.1.1 Load required libraries 44](#_Toc8223010)

[3.1.2 Get the version number of each of the main libraries 45](#_Toc8223011)

[3.1.3 note the start time for this EDA exercise 45](#_Toc8223012)

[3.1.4 load titanic dataset 45](#_Toc8223013)

[3.1.5 Missing values treatment 47](#_Toc8223014)

[3.1.6 Create New column - adult 47](#_Toc8223015)

[3.1.7 Factor plots 49](#_Toc8223016)

[3.1.8 Create New column – travel companion 51](#_Toc8223017)

[3.1.9 Compute the elapsed time for this exercise 54](#_Toc8223018)

[4. Excercises 54](#_Toc8223019)

[5. appendix 55](#_Toc8223020)

# **Plotting using pandas series and data frames**

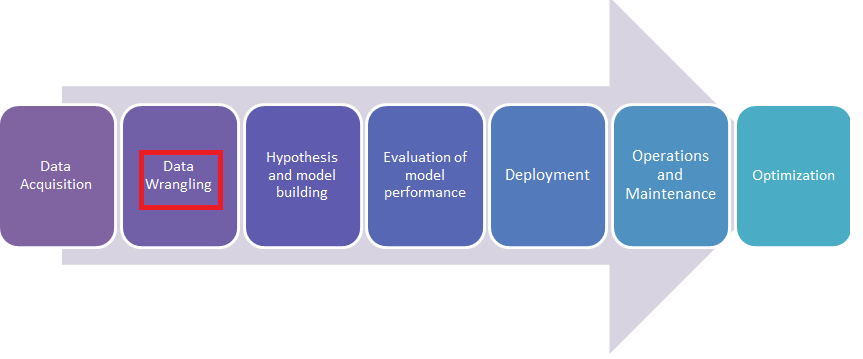
1. *The plot() method on Pandas Series and Data Frame is a simple wrapper around matplotlib.pyplot.plot().*
2. *On DataFrame, plot() is a convenience to plot all the columns with labels.*
3. *The following methods are provided as the kind keyword argument to plot(), and include in addition to the default, line plot:*
4. *bar or barh for bar plots*
5. *hist for histogram*
6. *box for boxplot*
7. *kde or density for density plots*
8. *area for area plots*
9. *scatter for scatter plots*
10. *hexbin for hexagonal bin plots*
11. *pie for pie plots*

*Refer:* [*https://pandas.pydata.org/pandas-docs/stable/user\_guide/cookbook.html#cookbook-plotting*](https://pandas.pydata.org/pandas-docs/stable/user_guide/cookbook.html#cookbook-plotting)

# **Data Manipulation**

* In data science data comes with collection and science starts with manipulation.

Look at stages of Data Science projects.

****

* Data manipulation is the process of changing data to make it easier to read or process or be more organized
* Data wrangling or data munging is the process of changing the raw data to make it clean enough to be used in analytical models.

**Refer:**

* **https://www.computerhope.com/jargon/d/datamani.htm**
* **https://www.springboard.com/blog/data-wrangling/**

**We broadly do data manipulation using the following strategies:**



## Creation of new variable

The following are the ways of creating new variables:

1. A column derived from the existing column(s) to get useful insights
2. Dummy variable (binary variable) creation to use certain models which expect numerical input
3. Combining variables to reduce the dimensionality
4. Reduce the levels of categorical variables to facilitate better view of the insights

*We shall restructure data sets to convert them from one format to another (i.e.., transformation) in order to get some meaningful insights.*

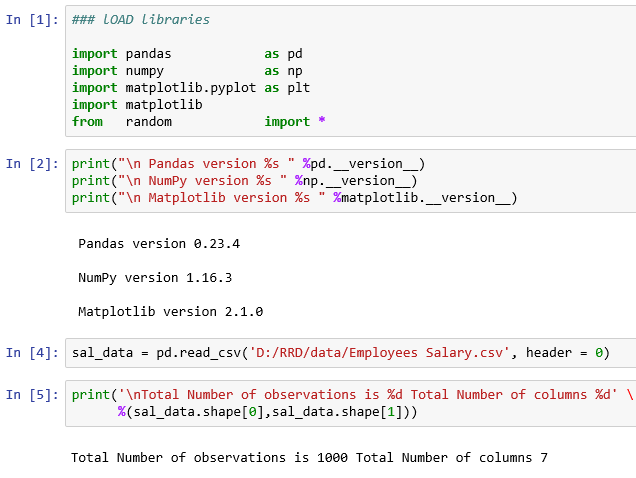
A Case study: A company employees’ salary

We shall use a subset of this information that contains the following information:

1. Employee ID: A unique number to identify the employee within the company
2. Age: Age of the employee
3. EduYears: Years of formal education
4. Salary: Basic salary per annum
5. Experience: Years of experience in the same company

### A column derived from the existing column(s) to get useful insights

*Load the required packages and the data set:*

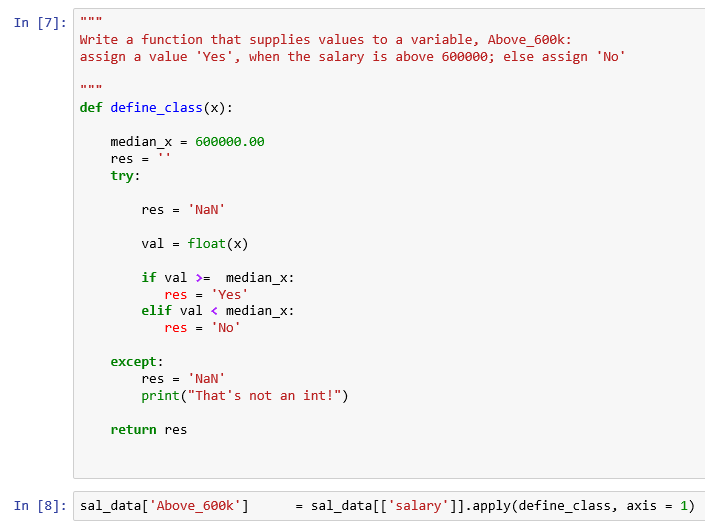


***We shall create variables as explained below:***

* *Create a variable, Above\_600k to indicate employees drawing more than Rs 600000 per annum.*
* *Discretize or separate into bins eduYears to facilitate quick analysis.*

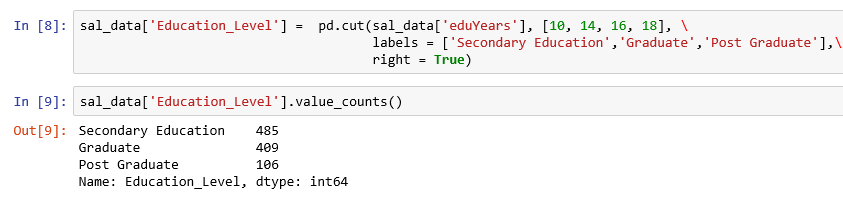
1. ***Create a new variable***

*Write a function to create the variable, Above 600k as follows:*



1. **Create custom bins for eduYears**

* *We use cut function of pandas, when you want to segment data values into bins.*
* *Here we use cut to convert Years of education, eduYears to groups of pre-specified array of bins. We divide these into bins of 10 to 14, 15 to 16, 17 to 18 and label them as [’Secondary Educa- tion’,’Graduate’,’Post Graduate’].*
* *Pandas cut computes these bins and return a Categorical object.*
* *We specify right = True to means that the bin includes the right most edge is included, and it is indicated by ] bracket.*

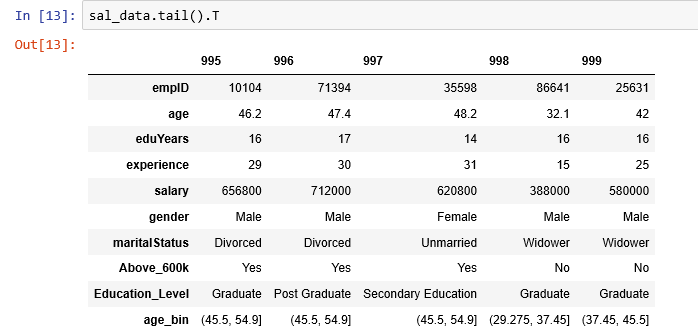


1. **Create quanitile bins for eduYears**

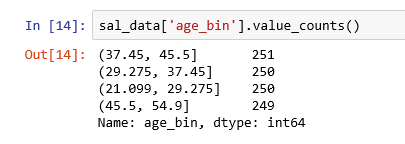
* *The function, qcut bins the data based on sample quantiles, which will yield roughly equal size bins.*
* *Let us use age to create quantile bins.*



*Examine the last five rows of the data frame, sal\_data.*



Let us get the count of each bin for the variable, age.

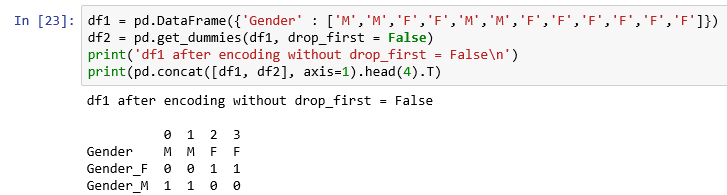


*We have created the categorical variables, Above\_600k, Education\_Level and age\_bin.*

### Dummy variables

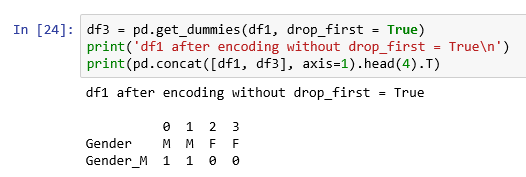
*Dummy variable (binary variable) creation to use certain models which expect numerical input*

* *There is another type of transformation in statistical modelling is to convert a categorical variable into dummy or indicator variable.*
* *If a categorical variable column has k distinct levels, you would get a matrix with k columns containing all 1s and 0s.*
* *Pandas has get\_dummies() for doing this. During dummy encoding, one level of the categorical feature becomes the reference group. By using the parameter drop\_first, we can ensure that a categorical variable of K categories, or levels, enters a model as a sequence of K-1 dummy variables.*



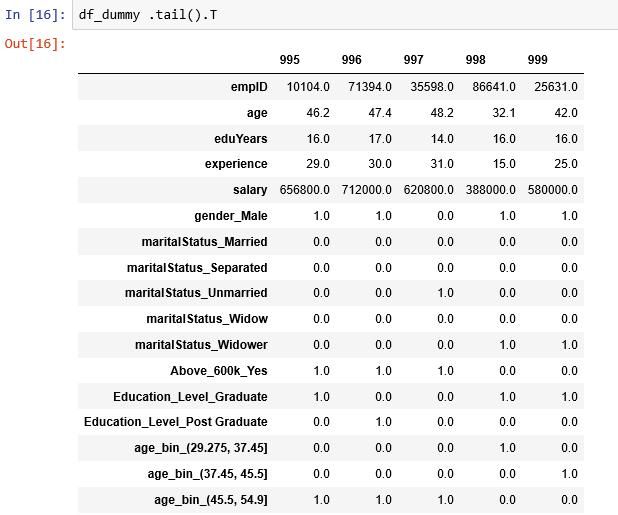
We observe that there is redundancy in representing the variable, Gender by both Gender\_F and Gender\_M.

Look at the first value for Gender. It is M (for Male). It can be represented as a numerical value 1 for the variable, Gender\_M or value 0 for the variable, Gender\_F. We don’t need both. We can drop one of them by using drop\_first = True as shown below:





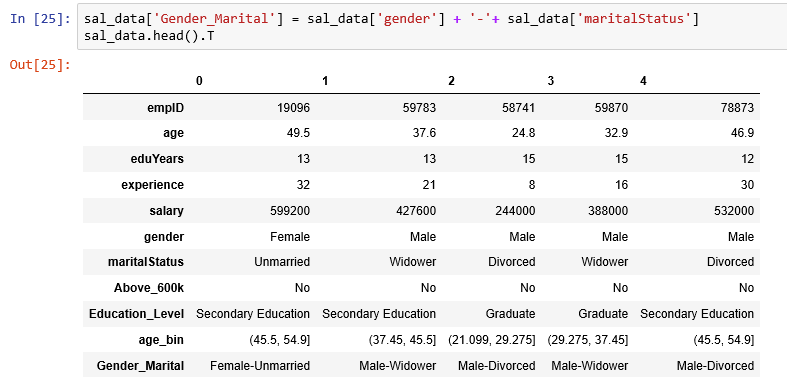
*Examine the last five rows, now.*



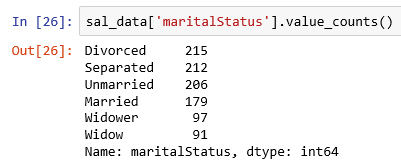
### Combining variables & levels to reduce the dimensionality

*Sometimes, it is useful to combine two variables such as caste and sub-caste.*

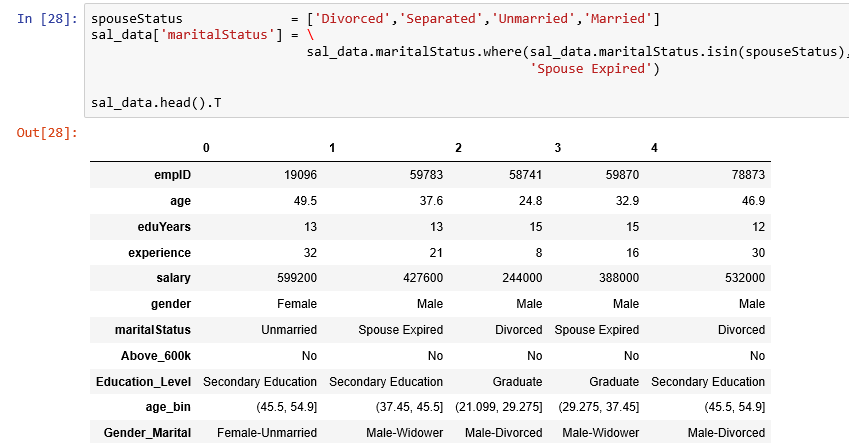
*Let us combine two variables, Gender (Male & Female) and Marital Status (Unmarried, Married , Widow/ Widower, Separated).*

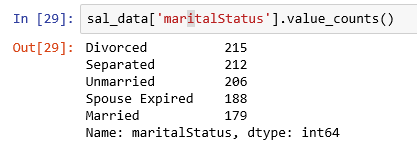


*Let us get the levels of maritalStatus.*



*Let us combine levels Widow and Widower as Spouse Expired and thus reducing one level.*





*Now we have five levels instead of six – levels,* ***not a great achievement but understanding this technique is important when you want to reduce a few thousand levels to a few levels****.*

## **Aggregate**

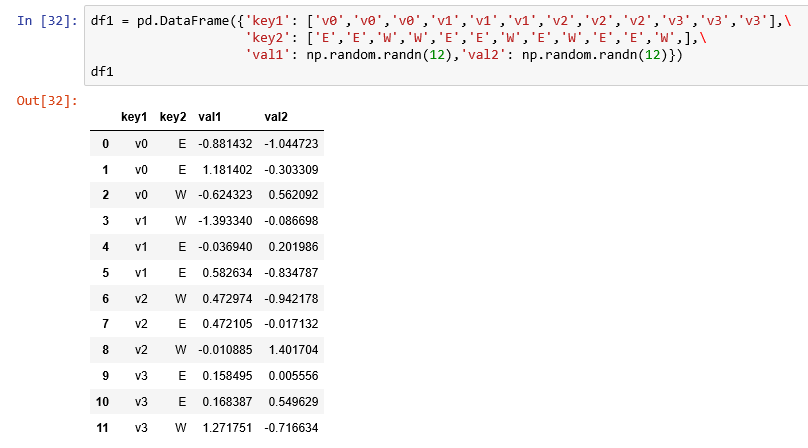
### Groupby operations

*Group operations is deﬁned by Split, Apply & Combine - Hadley Wickham, author of many R books:*

1. Data (contained in data frame or series or otherwise) is split into groups based on one or more keys you provide. The splitting is performed on a particular axis (row or column) of an object.
2. A function is applied to each group, producing a new value.
3. The results of all those function applications are combined into a result object.

*Each grouping key can take many forms, and the keys do not have to be all the same type. Ensure*

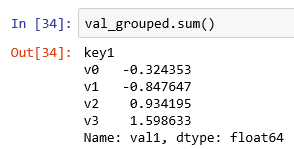
* *A list or array of values that is the same length as the axis being grouped.*
* *A value indicating a column name in a Data Frame.*
* *A dict or series giving a correspondence between the values on the axis being grouped and the group names.*
* *A function to be invoked on the axis index or the individual labels in the index.*



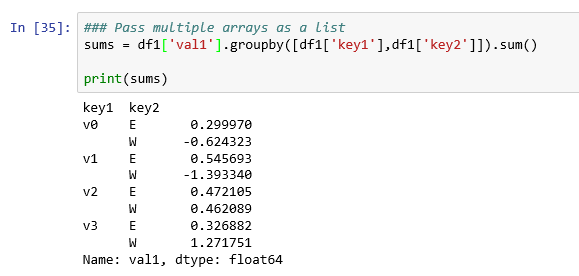
*Compute the sum of the val1 column using the labels from key1.*



*val\_grouped variable, is a groupby object. To compute group-wise sum we have called the GroupBy’s sum method.*

**

*We can group by more than one variable as shown below:*

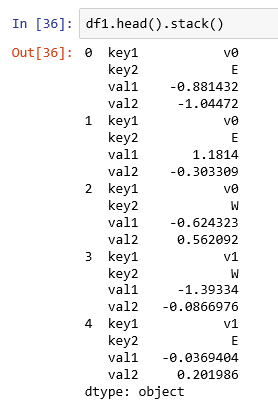


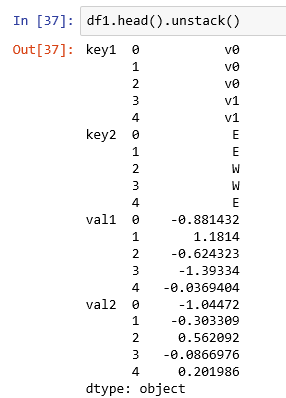
### Hierarchical indexing

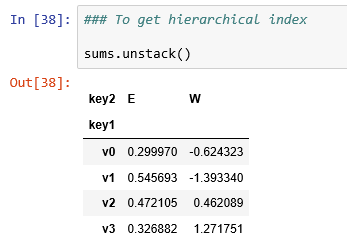
***Hierarchical indexing provides a consistent way to rearrange data in a data frame.***

*There are two main actions:*

1. *stack: results in rotation or pivoting from the columns in the data to the rows.*
2. *unstack: results in pivoting from the rows into columns.*

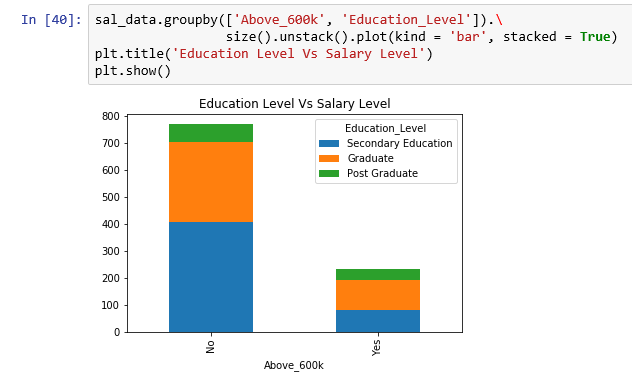






*Here we grouped the data using two keys, and the resulting Series now has a hierarchical index consisting of the unique pairs of the keys observed.*

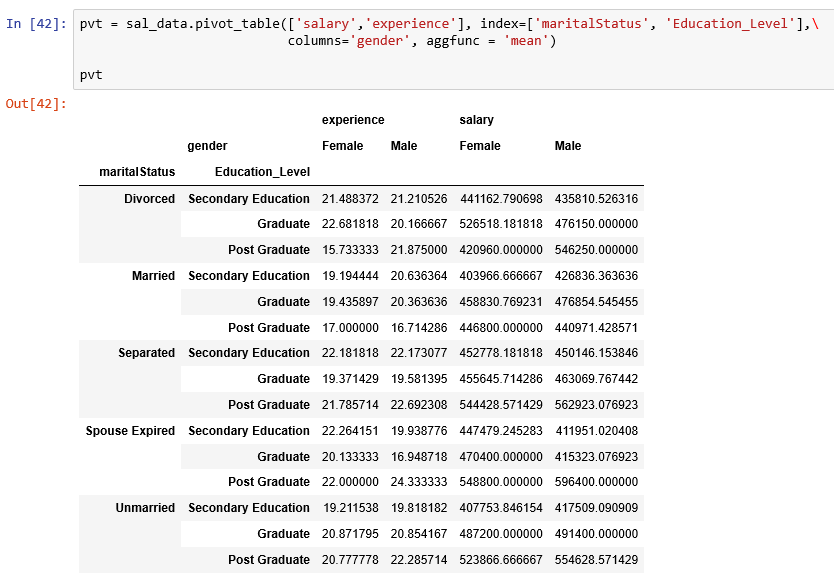
***Visualization***

******

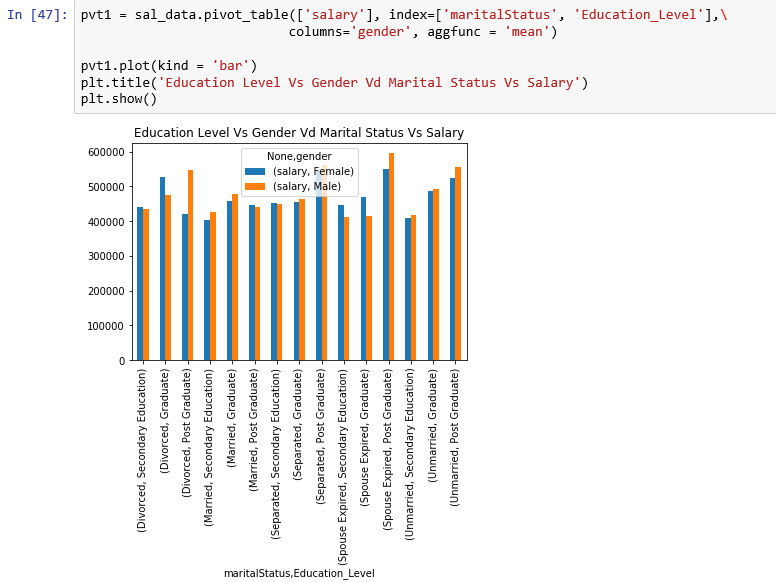
### Pivot table

* A pivot table is a data summarization tool frequently used in spreadsheet programs.
* It aggregates a table of data by one or more keys, arranging the data in a rectangle with some of the group keys along the rows and some along the columns.
* Function, groupby() and reshape operations of pandas make pivot tables possible.
* The function pivot\_table can provide margin totals (subtotals) in addition to providing an interface to groupby.

Use sal\_data. Compute a table of group means of salary and experience arranged by marital- Status. Put gender in the table columns and maritalStatus in the rows.



***Visualization***

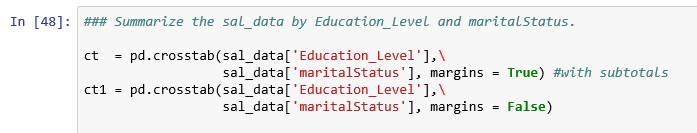


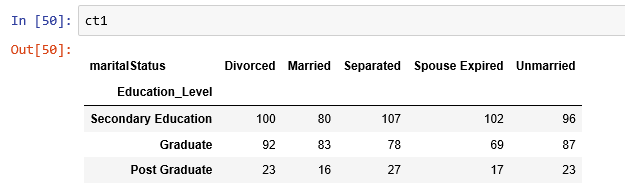
**Pivot table options**

| **Name** | **Description** |
| --- | --- |
| values | Column name or names to aggregate; by default aggregates all numeric columns |
| index | Column name or other group keys to group on the rows of the resulting pivot table |
| columns | Column names or other group keys to group on the columns of the resulting pivot table |
| aggfunc | Aggregation function or list of functions ('mean' by default); can be any function such as sum or count valid in a groupby context |
| fill\_value | Replace missing values in result table |
| dropna | If true, exclude columns whose entries are all NA |
| margins | Add row / column subtotals and grand total (False by default) |

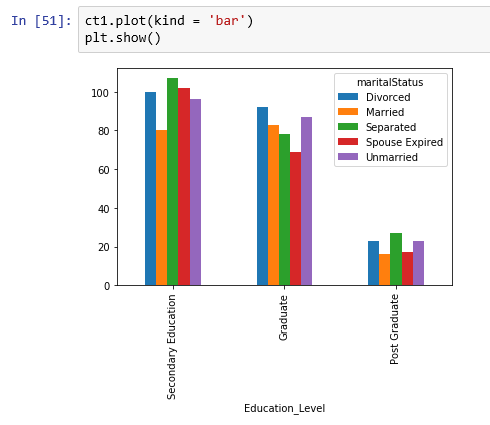
### Crosstab

A cross-tabulation (or crosstab) computes group frequencies. It is a special case of a pivot table.





**Visualization**



*Refer* [*http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.crosstab.html*](http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.crosstab.html)

## **Sort**

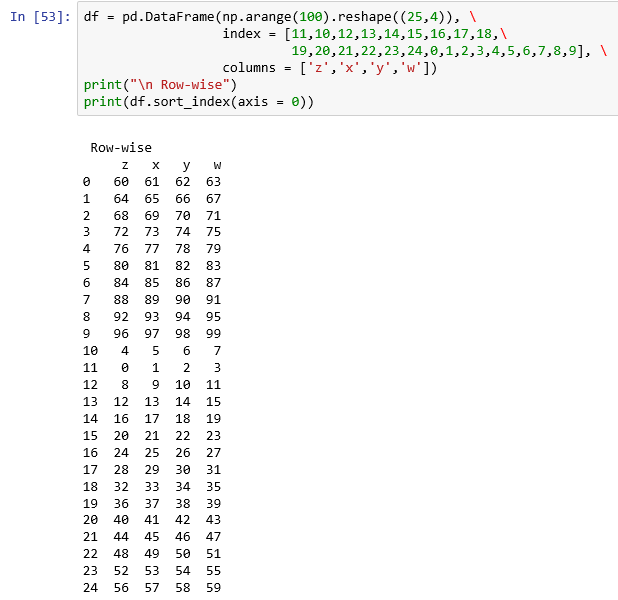
Often it is needed to rearrange the sequence of the rows of a data frame by sorting.

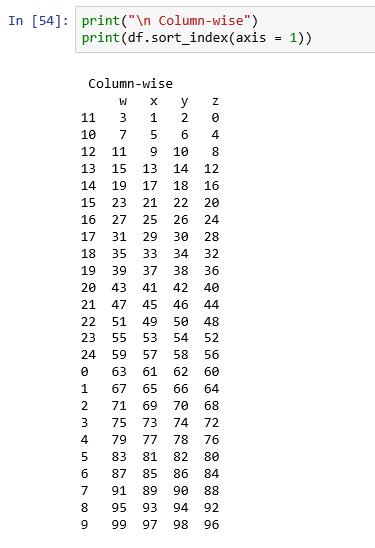
Types of sorting available in Pandas:

1. By label
2. By actual value

**By label**

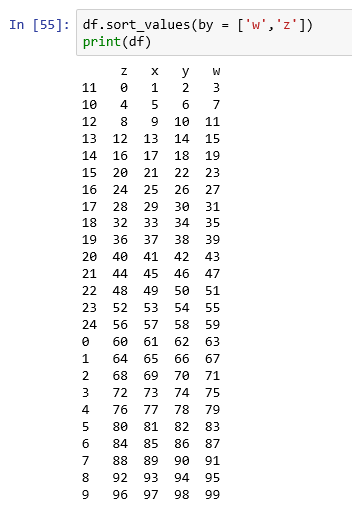
*To sort lexicographically by row or column index, use the sort\_index method, which returns a new, sorted object.*





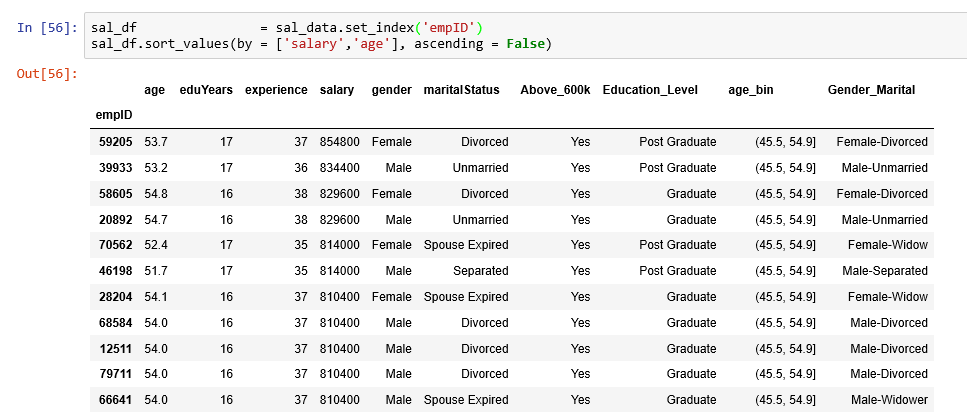
**By actual value**

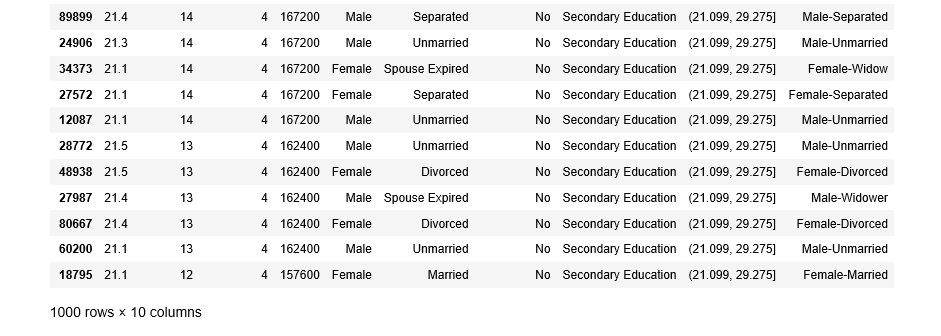
*You can sort using one or more columns as the sort keys when sorting a data frame by using sort\_values().*



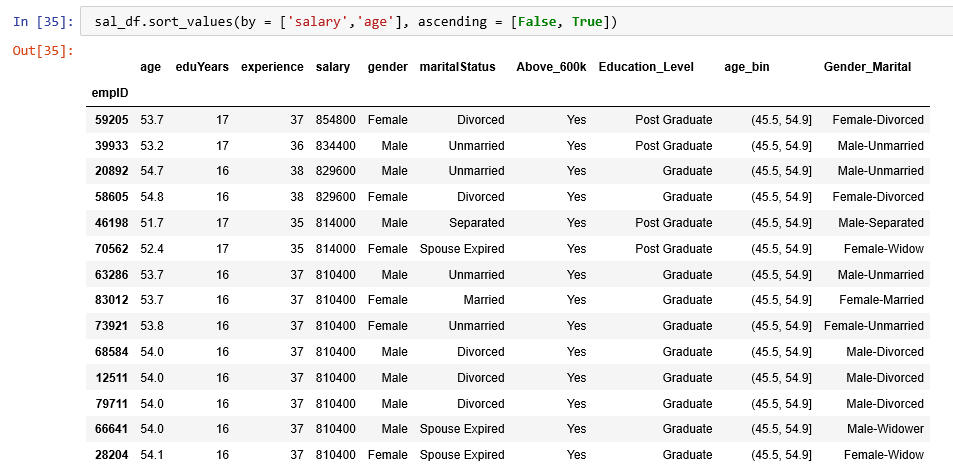
*The following code sorts the sal\_data, pandas dataframe by descending values of the column salary & age.*

*Set the index for sal\_data, a DataFrame using empID, existing column(s).*

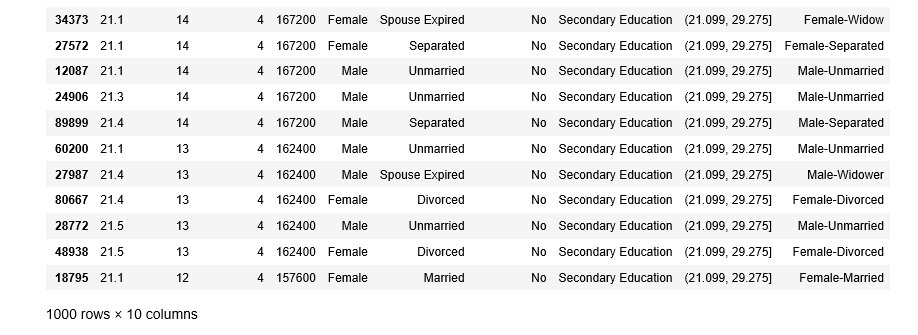


…  


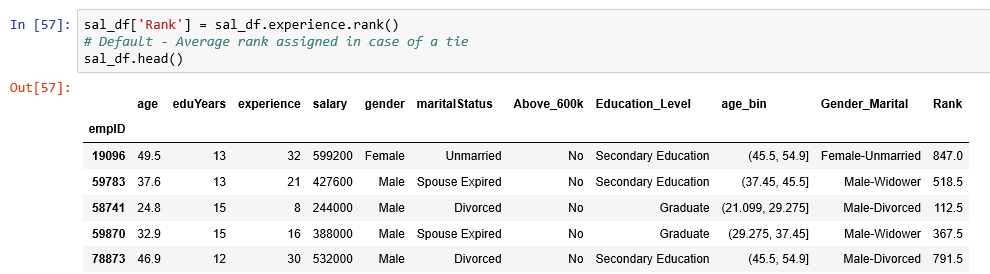
*The following code sorts the sal\_data, pandas dataframe by descending values of the column salary & ascending order of age.*

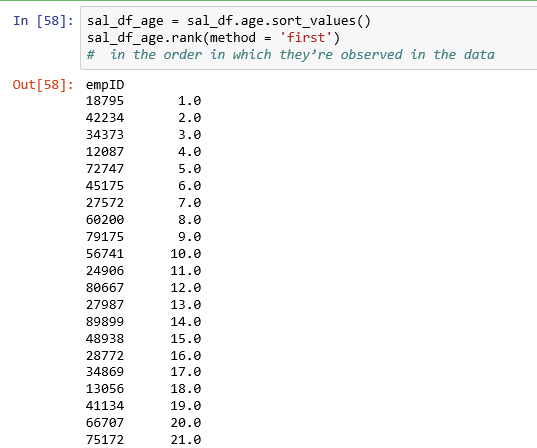


…

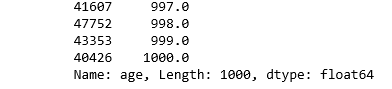


*Ranking assigns ranks from one through the number of valid data points in an array. In case of a tie, rank breaks the tie by assigning each group the mean rank.*

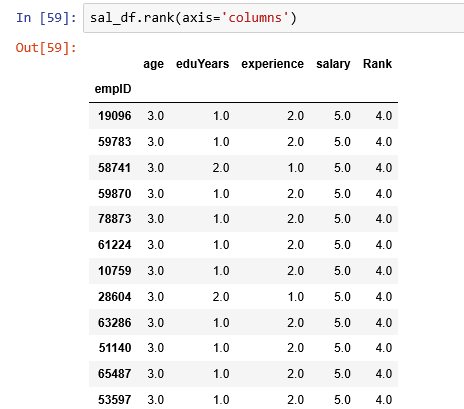




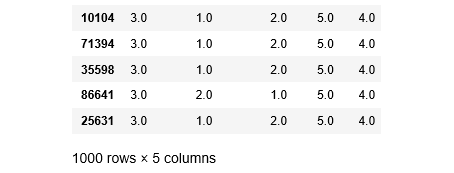
… ….



**Compute ranks over all numeric columns**

****

**… …**

****

**Tie breaking methods with rank**

| **Method** | **Description** |
| --- | --- |
| average | Default Assigns the average rank to each entry in the equal group. |
| min | Use the minimum rank for the whole group |
| max | Use the maximum rank for the whole group |
| first | Assign ranks in the order the values appear in the data |
| dense | Similar to min method; but ranks always increase by 1 in between groups rather than the number of equal elements in a group |

## **Merge**

*Pandas merge connects rows in DataFrames based on one or more keys and implements database join operations.*

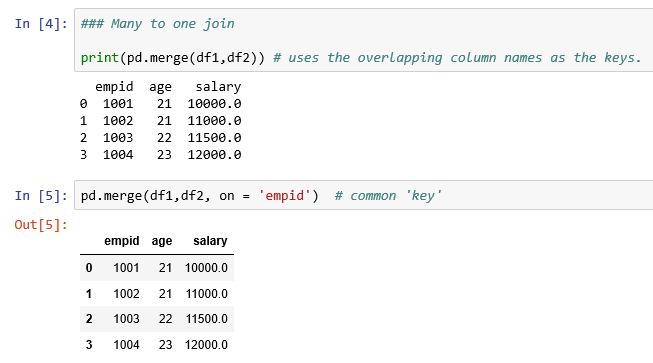
* *Pandas concat() concatenates or stacks together objects along an axis.*
* *Merge or join operations combine datasets by linking rows using one or more keys.*
* *Different join type with how argument*

|  |  |
| --- | --- |
| **Option** | **Behavior** |
| Inner\* | Use only the key combinations observed in both tables left Use all key combinations found in the left table |
| right | Use all key combinations found in the right table |
| output | Use all key combinations observed in both tables together |

\**Inner join where the keys in the result are the intersection or the common set found in both tables.*



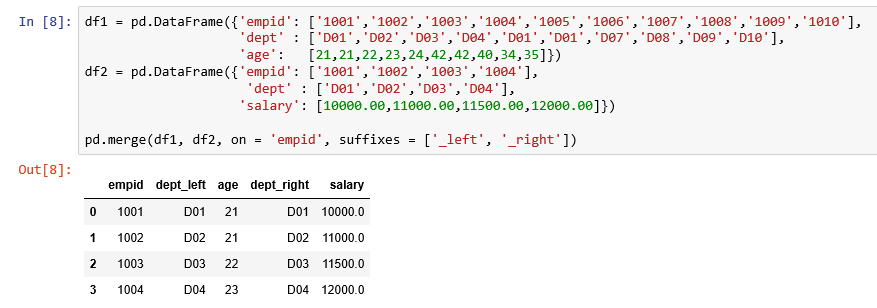
### Merge two data frames when key names are same



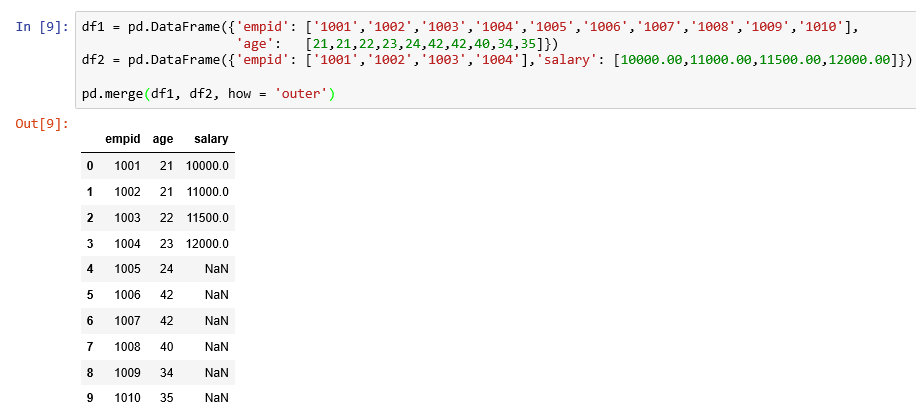
### Merge two data frames when key names are different



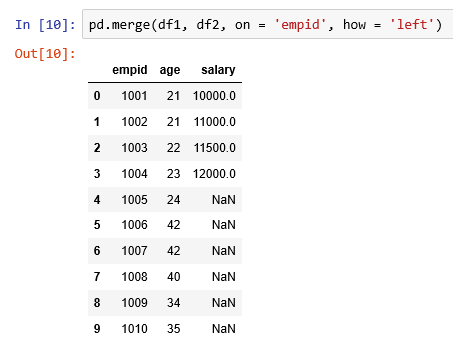
*We can use the suffixes option for specifying strings to append to overlapping names in the left and right Data Frame objects.*



*Outer join takes the union of the keys combining the effect of applying both left and right joins.*



*Many to many merges have well-defined, though not necessarily intuitive, behavior. Many to many joins form the Cartesian product of the rows.*

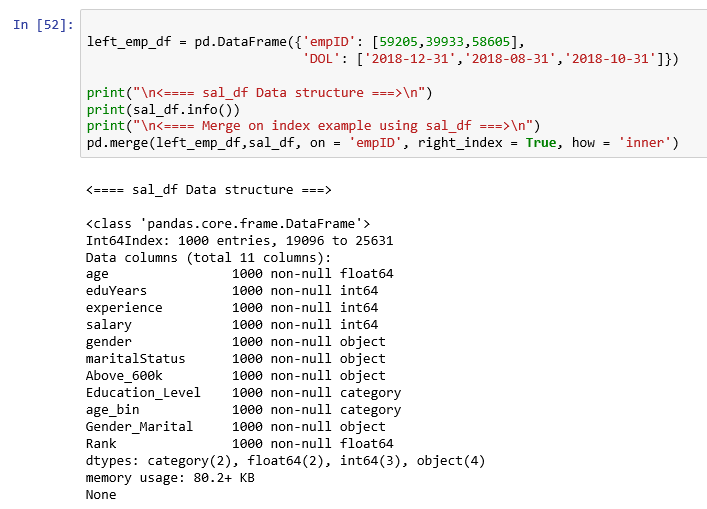


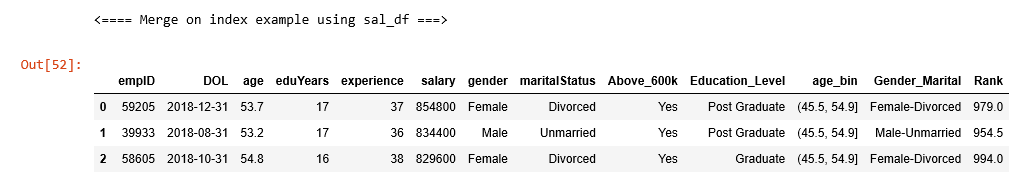
### Merge function arguments

|  |  |
| --- | --- |
| **Argument** | **Description** |
|  |  |
| left | DataFrame to be merges on LHS |
| right | DataFrame to be merges on RHS |
| how | One of inner, outer, left or right defaults to inner |
| on | Column names to join on |
| left\_on | Columns in the left DataFrame to use as join key |
| right\_on | Columns in the right DataFrame to use as join key |
| left\_index | Use row index in left as its join key for keys |
| right\_index | Use row index in right as its join key for keys |
| sort | Sort merged data lexicographically by join keys; default true |
| suffixes | Tuple of string values to append column names in case of overlap; defaults to (’\_x’,’\_y’) |
| copy | Copies resulting data structure; default True |
| indicator | Adds a special column\_merge that indicates the source of each row such as 'left\_only', 'right\_only', 'both' based on the origin of the joined data in each row |

### Merging on index

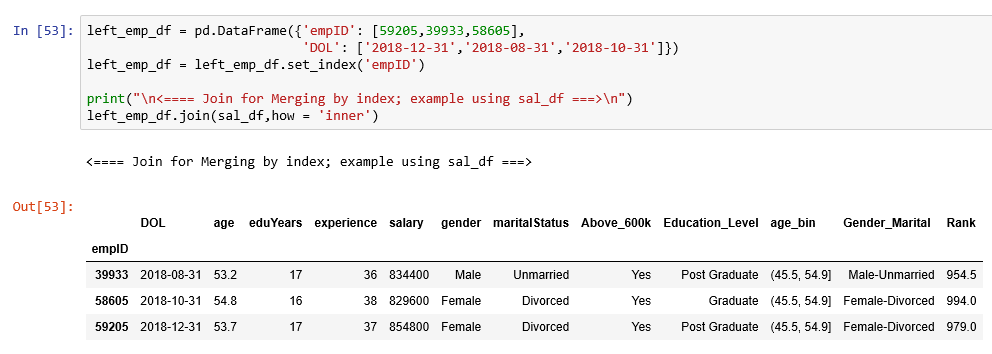
*There may be cases, index of a data frame may contain the merge keys and you need to pass left\_index = True or right\_index = True.*



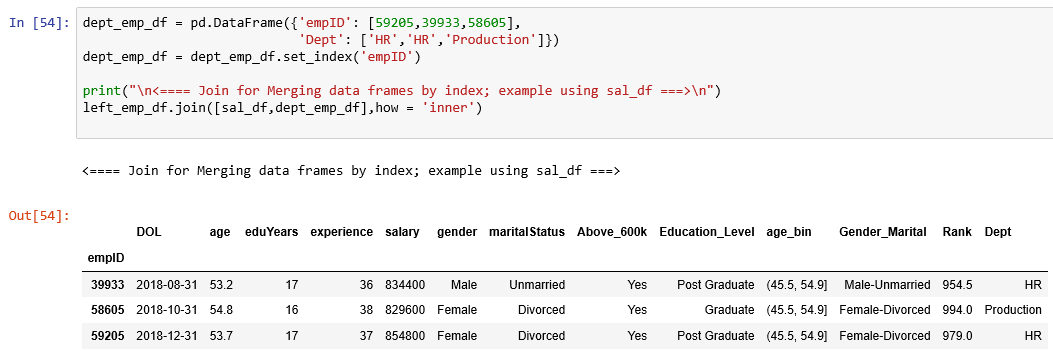


### Join

*Data frame has a convenient join instance for merging by index. We can combine many data frames together having the same or similar indexes but non-overlapping columns.*



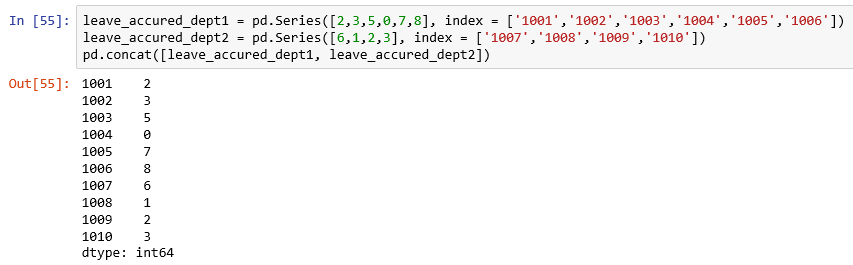
*You can pass a list of data frames to join as an alternative way to concat() function.*

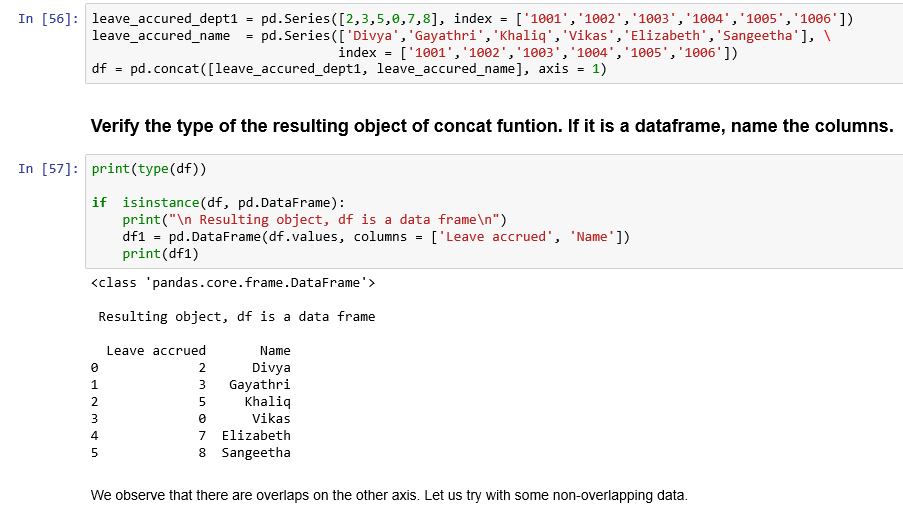


## **Append**

### a. concat function

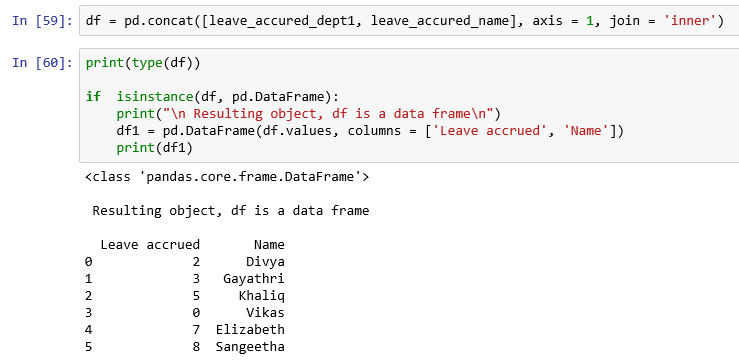
* *The concat function in pandas provides a uniform way to combine the data as shown below:*
* *By default, concat works along axis = 0 yielding another series. If you specify axis = 1, you get a data frame.*



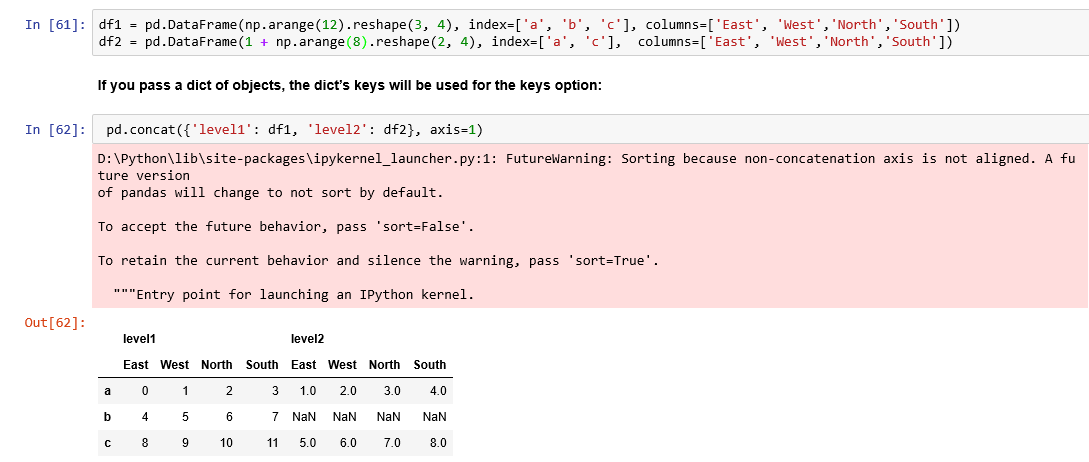


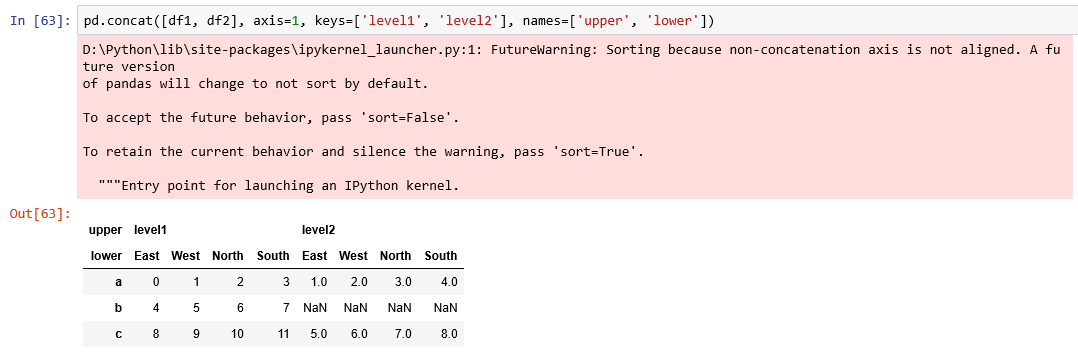


* *By using inner join, the non-overlapping values indicated by NaN having labels ’Ra- jesh’,’Raju’,’Susan’,’Samuel’ disappear as shown below:*



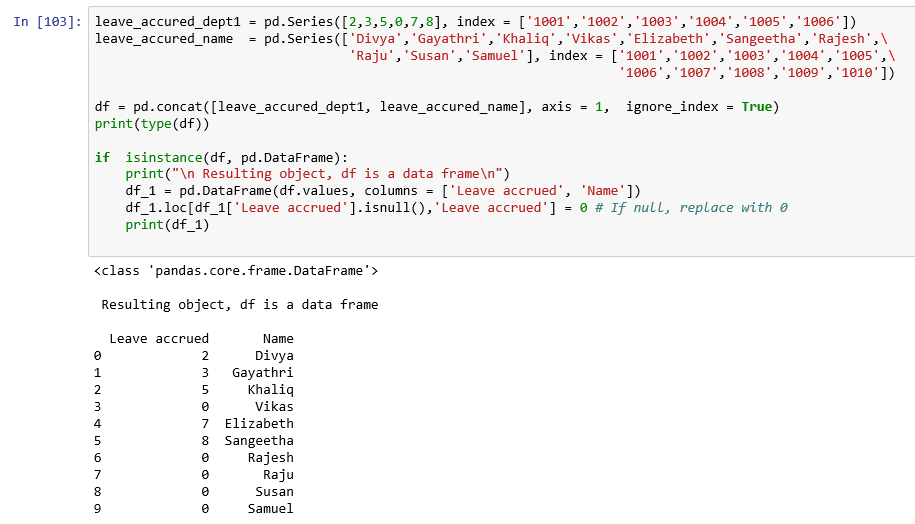
* *You can create horizontal axis and name the created axis levels.*





*Refer:* [*http://www.datasciencemadesimple.com/hierarchical-indexing-multiple-indexing-python-pandas/*](http://www.datasciencemadesimple.com/hierarchical-indexing-multiple-indexing-python-pandas/)

* *If you want data to be created for all employees, whether they have accrued leave or not, you pass the argument, ignore\_index = True.*

**

***Concat function arguments***

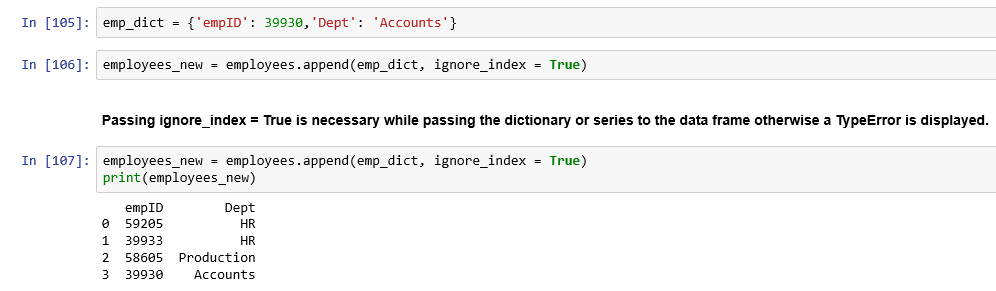
| **Argument** | **Description** |
| --- | --- |
| **objs** | **List or dict of pandas objects to be concatenated** |
| **axis** | **axis to concatenate along 0 means rows 1 means columns** |
| **join** | **'inner' or 'outer' Intersection or union; def: outer** |
| **join\_axes** | **Specific indexes to unse for other n-1 axes instead of performing union or intersection logic** |
| **keys** | **Values to associate with objects being concatenated, forming a hierarchical index along the concatenation axis; can either be a list or array or arbitrary values, an array of tuples, or a list of arrays** |
| **levels** | **Specific indexes to use as hierachical index level or levels of keys passed** |
| **names** | **Names for created hierarchical levels if keys and / or levels passed** |
| **verify\_integrity** | **Check new axis in concatenated object for duplicates; default: False** |
| **ignore\_index** | **Do not preserve indexes along concatenation axis, instead producing a new range (total\_length) index** |

### 1.5.2 append functionfunction

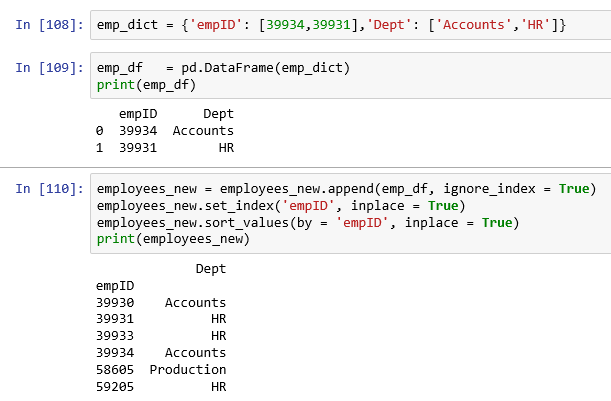
1. ***Add rows in a DataFrame using append()***

******

1. ***Add a row to the data frame using dataframe.append() and dictionary***

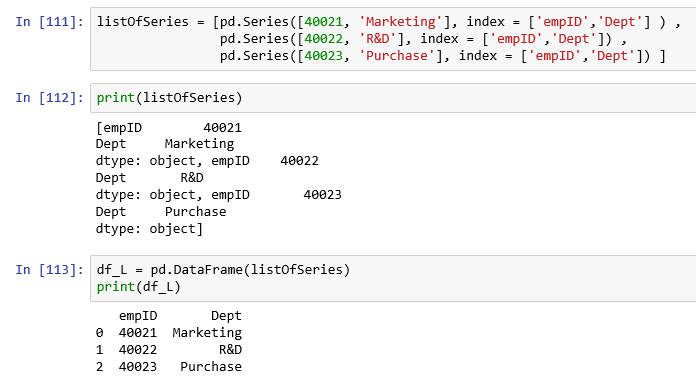


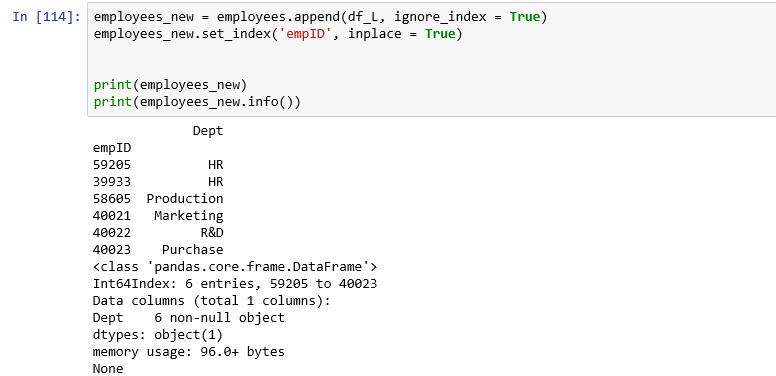
1. ***Add rows in the data frame using dataframe.append()***



[*https://thispointer.com/python-pandas-how-to-add-rows-in-a-dataframe-using-dataframe- append-loc-iloc/*](https://thispointer.com/python-pandas-how-to-add-rows-in-a-dataframe-using-dataframe-%20append-loc-iloc/)

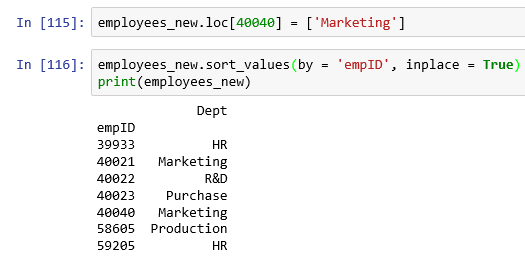
1. ***Add multiple rows in the data frame using dataframe.append() and Series***





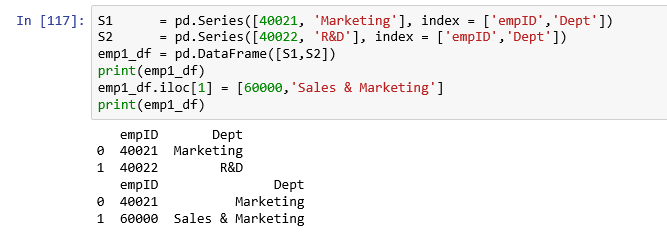
* *We observe that employees\_new, data frame has one index column and one other column, Dept.*
* *We shall add a new row at index 40040 with values provided in list. It will replace any existing value in that row or add to the new row.*

1. ***Add a row in the data frame using loc[] & list***

****

1. ***Add a row in the data frame at the index position using iloc[]***

* *It will replace the row at index position 1 in dataframe emp1\_df with new row*



1. ***The function, combine\_first does patching missing data in the calling object with data from the object you pass:***



# **Case study**



## Titanic Data set

This popular data set provides information on the fate of passengers on the fatal, maiden voyage of the ocean liner, RMS Titanic on April 15, 1912 after colliding with an iceberg.

The data set has the following attributes :

1. PassengerId - An idenfier for each passenger

2. Survived - Survived (1) or died (0)

3. Pclass - Passenger's Class

4. Name - Name of the passenger

5. Sex - Gender of the passenger

6. Age - Age of the passenger

7. SibSp - Number of siblings / spouses aboard

8. Parch - Number of parents / children aboard

9. Ticket - Ticket number

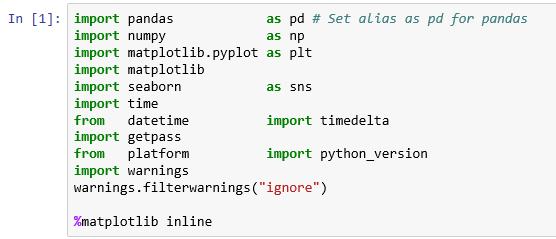
10. Fare - Fare paid by the passenger

11. Cabin - Cabin

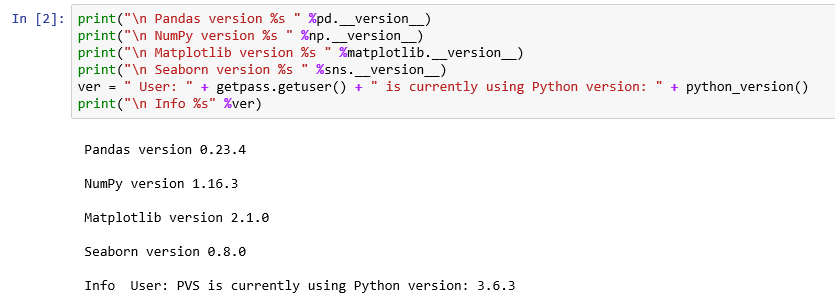
12. Embarked - Port of embarkation

Explore the data set to get a good understanding using data manipulation techniques. Here we attempt to explore only a few independent variables to illustrate the data manipulation techniques.

### Load required libraries



### Get the version number of each of the main libraries



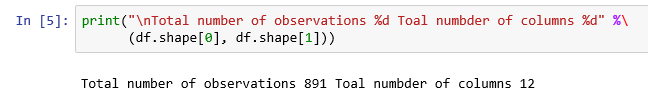
### note the start time for this EDA exercise



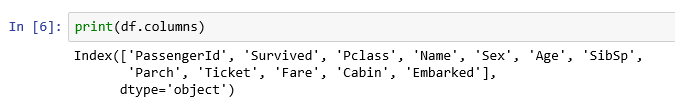
### load titanic dataset



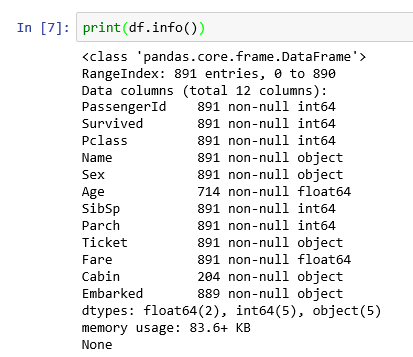
**Get the number of observations and number of columns**



**Get column names**

****

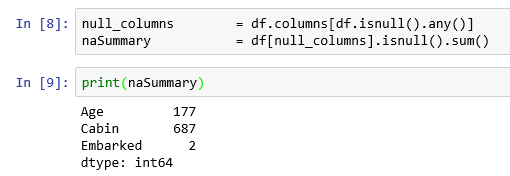
**Get data structure**

****

**We observe that there are missing values for the variables:**

1. Age: 177 values missing (= 891 – 714)
2. Cabin: 687 values missing (= 891 – 204)
3. Embarked: 2 values missing (= 891 – 889)

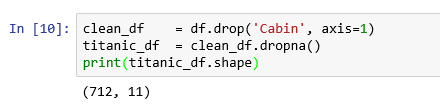
We can get the same information by using the following Python code:



### Missing values treatment

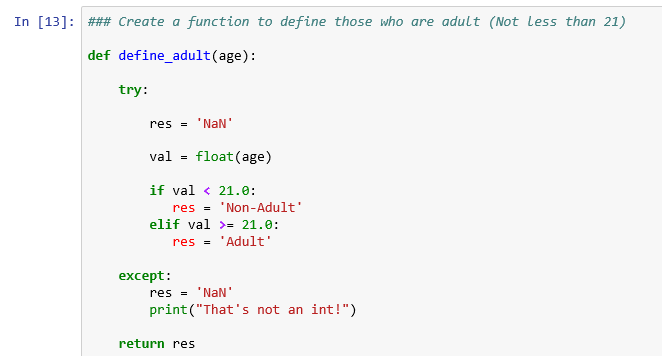
**Remove missing values**

* *Drop column Cabin since the % of missing values is 77% (= 687 / 891 )*
* *Delete the missing values row-wise*

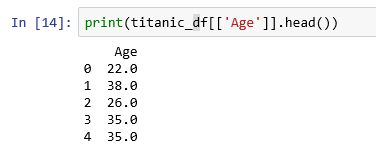
**

### Create New column - adult

* *A passenger is classified as an Adult if the age is 21 and above. Create a function to define those who are adult (Not less.*



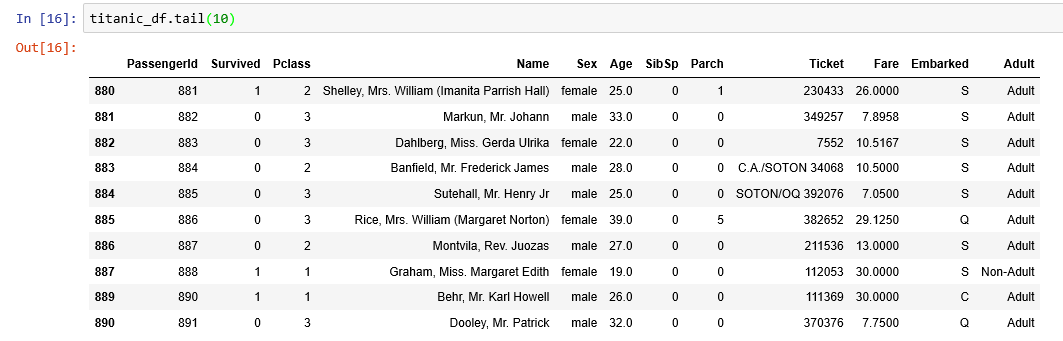
* *Let us see the first five observations for the field, Age.*



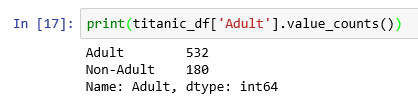
* *Create a new column, Adult by applying the function, define\_adult() we have defined earlier.*



* *Let us see the lat five observations for the data frame, titanic\_df.*



* *We have created a new column, called Adult. Let us check the number of Adults and non-adults*



**We have 534 adult passengers and 180 non-adult passengers.**

### Factor plots

* *A factor plot is simply the same plot generated for different response and factor variables and arranged on a single page.*
* *Using factor plot we can plot any graph by specifying 'kind' argument. We can pass one of the values mentioned below to the kind, argument:*

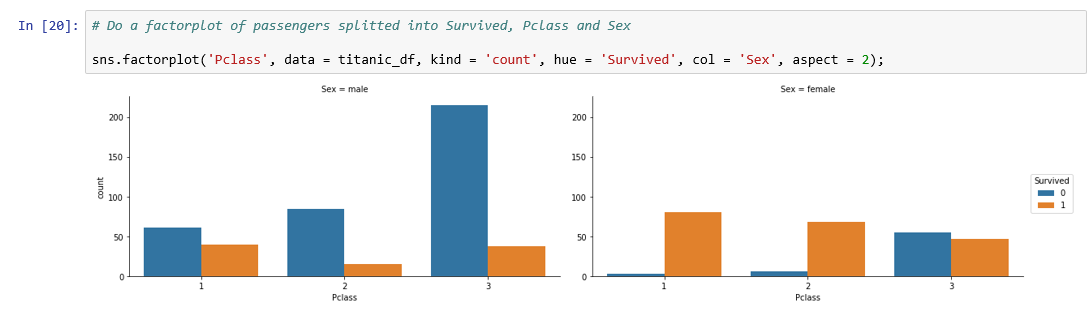
|  |  |
| --- | --- |
| **Value for kind argument** | **Description** |
| point | A point plot represents an estimate of central tendency for a numeric variable by the position of scatter plot points and provides some indication of the uncertainty around that estimate using error bars.  <http://seaborn.pydata.org/generated/seaborn.pointplot.html> |
| bar | A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.  <https://en.wikipedia.org/wiki/Bar_chart> |
| count | This plot shows the counts of observations in each categorical bin using bars.  <https://seaborn.pydata.org/generated/seaborn.countplot.html#seaborn.countplot>. |
| box | A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range.  <https://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot> |
| violin | A violin plot plays a similar role as a box and whisker plot. It shows the distribution of quantitative data across several levels of one (or more) categorical variables such that those distributions can be compared. Unlike a box plot, in which all of the plot components correspond to actual datapoints, the violin plot features a kernel density estimation of the underlying distribution.  <https://seaborn.pydata.org/generated/seaborn.violinplot.html#seaborn.violinplot> |
| strip | A strip plot can be drawn on its own, but it is also a good complement to a box or violin plot in cases where you want to show all observations along with some representation of the underlying distribution.  <https://seaborn.pydata.org/generated/seaborn.stripplot.html#seaborn.stripplot>  <http://seaborn.pydata.org/tutorial/categorical.html> |

#### factor plot for Survived Vs Adult and Sex



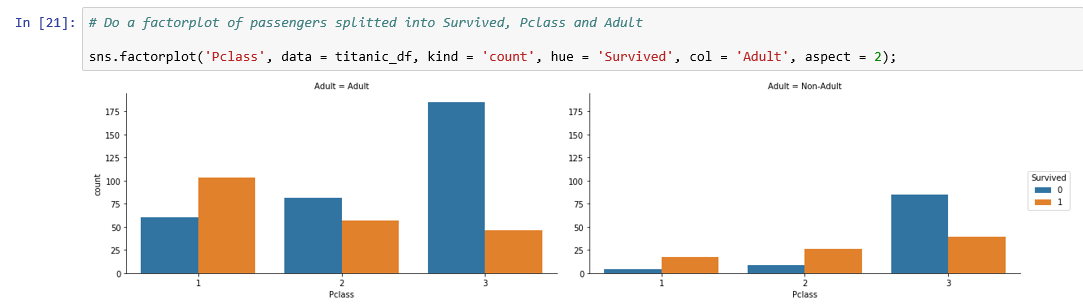
* *When compared with Male passengers, many female passengers were lucky to survive.*
* *There are a greater number of Female young persons (Non Adults) among survivors than the rest.*

#### factor plot for Survived Vs PClass and Sex



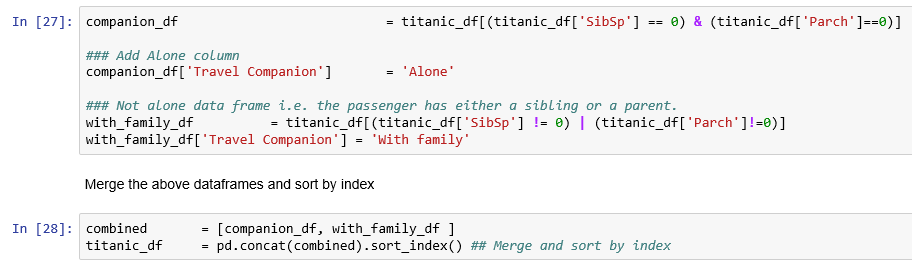
* *When compared with Third and Second Class passengers, First Class passengers were lucky to survive irrespective of their gender.*
* *There are a greater number of Female first class passengers among survivors than the rest.*

#### factor plot for Survived Vs PClass and adult

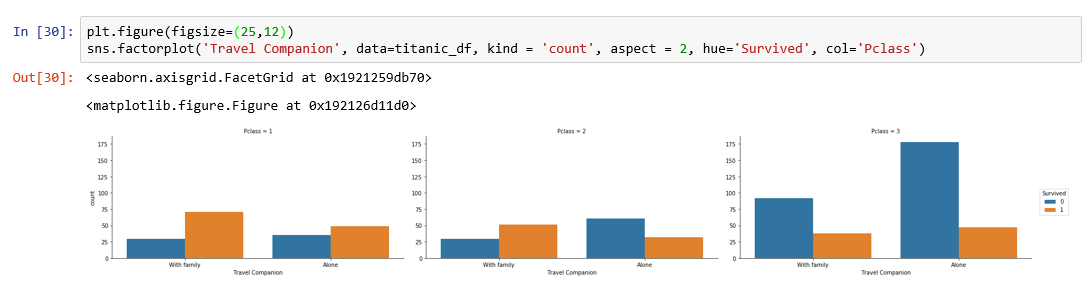


* *When compared with Adult passengers, young passengers were lucky to survive irrespective of their Passenger class except for Passenger class 3.*
* *There are a greater number of survivors among adults in passenger class 1 than the rest and a greater number of adult passengers died in passenger class 3.*

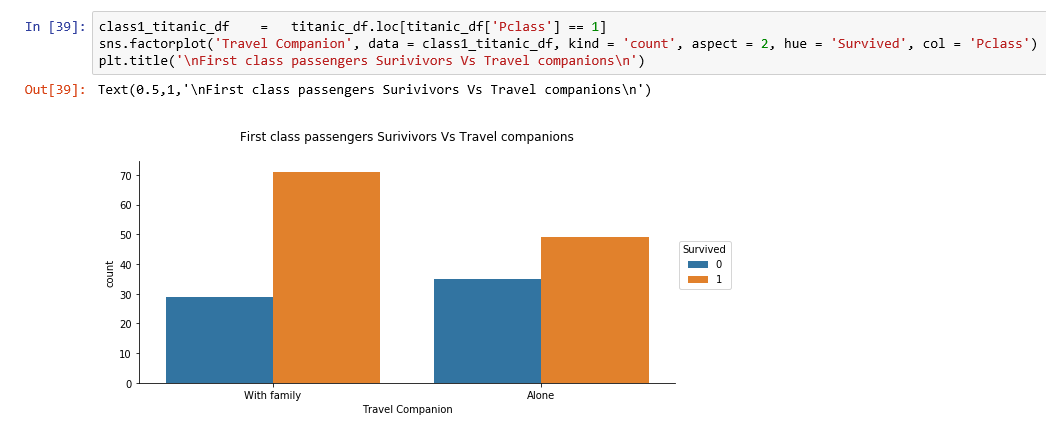
### Create New column – travel companion



**Let us do a factor plot for Survived vs. Travel companion Grouped by Class**



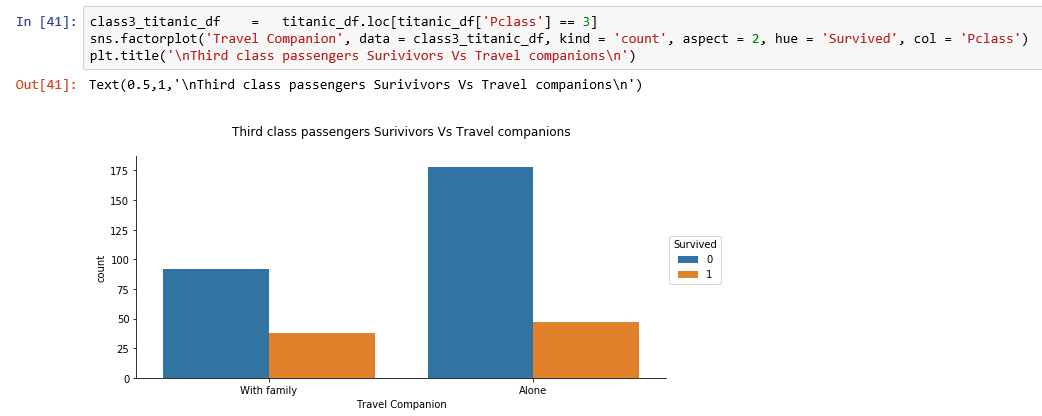
**Since the above graphs are not clear, let us do it for each passenger class.**



* *When compared with passengers travelling alone, passengers travelling with family were lucky to survive in passenger class 1.*



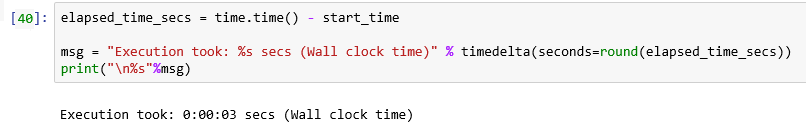
* *When compared with passengers travelling alone, passengers travelling with family were lucky to survive in passenger class 2 like passengers in class 1.*



* *When compared with passengers travelling alone, proportion of passengers travelling with family were lucky to survive in passenger class 3 like passengers in class 1 or class 2.*

### Compute the elapsed time for this exercise

Remember you have stored the start time of running this exercise in the variable, start\_time in page 45.



# Excercises

Explore the following data sets and give your inferences:

1. Boston Housing Prices data set - boston\_house\_prices.csv
2. Pima Indians Diabetes data set - pima-indians-diabetes.data.csv
3. The mtcars data set – mtcars.csv

# appendix

**Employee data set – a few records only**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| empID | age | eduYears | experience | salary | gender | maritalStatus |
| 19096 | 49.5 | 13 | 32 | 599200 | Female | Unmarried |
| 59783 | 37.6 | 13 | 21 | 427600 | Male | Widower |
| 58741 | 24.8 | 15 | 8 | 244000 | Male | Divorced |
| 59870 | 32.9 | 15 | 16 | 388000 | Male | Widower |
| 78873 | 46.9 | 12 | 30 | 532000 | Male | Divorced |
| 61224 | 34.2 | 15 | 17 | 406000 | Male | Separated |
| 10759 | 36.5 | 16 | 20 | 484000 | Male | Separated |
| 28604 | 24.6 | 12 | 8 | 215200 | Female | Widow |
| 63286 | 53.7 | 16 | 37 | 810400 | Male | Unmarried |
| 51140 | 54.6 | 14 | 38 | 738400 | Female | Widow |
| 65487 | 34.7 | 12 | 18 | 359200 | Male | Unmarried |
| 53597 | 50.5 | 17 | 34 | 793600 | Female | Divorced |
| 32618 | 23.2 | 16 | 6 | 215200 | Female | Unmarried |
| 83874 | 44.3 | 13 | 27 | 521200 | Male | Married |
| 33385 | 44.4 | 14 | 27 | 553600 | Male | Widower |
| 76983 | 31.7 | 14 | 15 | 352000 | Male | Widower |
| 32897 | 42.7 | 13 | 26 | 505600 | Male | Married |
| 31345 | 42.8 | 16 | 26 | 599200 | Female | Married |
| 24937 | 37.8 | 15 | 21 | 478000 | Male | Unmarried |
| 28578 | 50.1 | 13 | 33 | 614800 | Male | Separated |
| 35328 | 25.9 | 16 | 9 | 272800 | Male | Married |
| 34215 | 35.3 | 16 | 18 | 445600 | Female | Separated |
| 22723 | 27.8 | 17 | 11 | 324400 | Male | Unmarried |
| 13199 | 54.3 | 12 | 37 | 632800 | Female | Widow |
| 27503 | 37.8 | 13 | 21 | 427600 | Female | Widow |
| 74847 | 33.7 | 17 | 17 | 446800 | Female | Separated |
| 52055 | 49.9 | 14 | 33 | 654400 | Female | Widow |
| 83172 | 44.6 | 13 | 28 | 536800 | Male | Separated |
| 76507 | 33.2 | 15 | 16 | 388000 | Female | Unmarried |
| 13661 | 33.7 | 13 | 17 | 365200 | Male | Divorced |
| 69852 | 43.6 | 15 | 27 | 586000 | Male | Divorced |