Pandas Documentation

Installation

On Ananconda Distributions, Pandas can be installed using the following command:

```
conda install pandas
```

Pandas can also be installed from the command line using pip using

```
pip install pandas
```

Series and DataFrames

Pandas Series is a labelled 1d array capable of holding data of any type (int,str,float,objects,etc.) The axis labels are collectively called as index

Pandas DataFrames is a 2d array that is able to store heterogeneous data items which can input in the form of a dictionary

```
In [2]: data = {
             'apple' : [3,1,4,5],
             'orange' : [1,5,6,8]
         print(data)
         print(type(data))
         {'apple': [3, 1, 4, 5], 'orange': [1, 5, 6, 8]}
         <class 'dict'>
        df = pd.DataFrame(data)
In [3]:
In [4]:
Out[4]:
           apple orange
         0
           3
                  1
         1
           1
                  5
                  6
           4
         3
           5
                  8
```

Accessing Elements from a Pandas Series/DataFrame

Elements can from a pandas dataframe just like a dictionary

./media/image2.png

Reading CSV Files and inserting them, constructing Pandas Data Frames and viewing the data

Path can be a chosen as a raw string and feed into pandas via the read_csv(), read_tsv() or read_table() command as illustrated below:

The path variable must a reference to a tabular datasheet in the user's local machine

Once the DataFrame has been created, the top 5 rows can be viewed using the head() command



Moreover, the bottom rows can also be viewed using the tail command (). Also, a desirable number of rows can be passed into the bracket.



Setting a Different Index

Pandas gives the user an option of setting indices to other data columns using the index_col parameter in the read_csv command

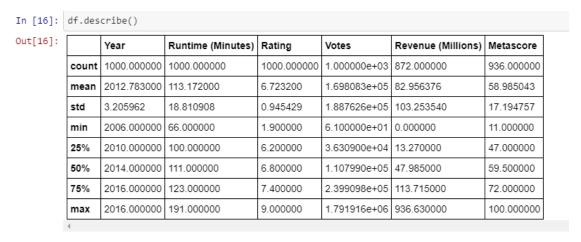
```
In [11]: df = pd.read_csv(path,index_col='Name')
          df.head()
Out[11]:
                                         Number Position Age
                          Team
                                                                Height Weight College
                                                                                                 Salary
           Name
           Avery Bradley
                          Boston Celtics 0.0
                                                 PG
                                                           25.0
                                                                6-2
                                                                        180.0
                                                                               Texas
                                                                                                7730337.0
           Jae Crowder
                          Boston Celtics
                                                 SF
                                                           25.0
                                                                6-6
                                                                        235.0
                                                                               Marquette
                                                                                                6796117.0
           John Holland
                          Boston Celtics
                                                 SG
                                                           27.0 6-5
                                                                       205.0
                                                                               Boston University
                                        30.0
                                                                                                NaN
                          Boston Celtics 28.0
           R.J. Hunter
                                                           22.0 6-5
                                                                        185.0
                                                                               Georgia State
                                                                                                1148640.0
                                                 SG
           Jonas Jerebko | Boston Celtics | 8.0
                                                           29.0 6-10
                                                                       231.0
                                                                               NaN
                                                                                                5000000.0
```

Getting Useful Insights into the data

The user can get description of Column names, nullability and data types of the columns using the info() command

```
In [15]: df.info()
         <class 'pandas.core.frame.DataFrame'>
         Int64Index: 1000 entries, 1 to 1000
         Data columns (total 11 columns):
              Column
                                 Non-Null Count Dtype
             -----
                                 -----
                                                ----
             Title
                                 1000 non-null
                                                 object
          0
          1
             Genre
                                 1000 non-null
                                                 object
          2
             Description
                                1000 non-null
                                                 object
             Director
                                1000 non-null
          3
                                                 object
          4
             Actors
                                 1000 non-null
                                                 object
          5
             Year
                                1000 non-null
                                                 int64
             Runtime (Minutes) 1000 non-null
                                                 int64
          6
          7
             Rating
                                 1000 non-null
                                                 float64
          8
             Votes
                                 1000 non-null
                                                 int64
             Revenue (Millions) 872 non-null
                                                 float64
          10 Metascore
                                 936 non-null
                                                 float64
         dtypes: float64(3), int64(3), object(5)
         memory usage: 74.2+ KB
```

Further, detailed statistics can constructed from the data using the describe() command.



The shape of the Data-Frame can be checked using shape command

```
In [17]: df.shape
Out[17]: (1000, 11)
```

Checking for Duplicates

The Data-Frames might contain duplicates and this can be checked using duplicated () cmd.

```
In [18]: df.duplicated()
Out[18]: Rank
                 False
         4
                 False
         5
                 False
                 False
         996
         997
                 False
         998
                 False
                 False
         Length: 1000, dtype: bool
In [19]: sum(df.duplicated())
Out[19]: 0
In [20]: df1 = df.append(df)
         df1.shape
Out[20]: (2000, 11)
In [21]: df1.duplicated().sum()
Out[21]: 1000
In [22]: df2 = df1.drop_duplicates()
In [23]: print(df.shape)
         print(df1.shape)
         print(df2.shape)
         (1000, 11)
         (2000, 11)
          (1000, 11)
```

The columns can be viewed using df.columns

```
In [25]: # Columns
In [26]: df.columns
'Metascore'],
dtype='object')
In [27]: len(df.columns)
Out[27]: 11
In [28]: df.describe()
Out[28]:
                             Runtime (Minutes) Rating
                                                           Votes
                                                                        Revenue (Millions) Metascore
           count 1000.000000 1000.000000
                                               1000.000000 1.000000e+03 872.000000
                                                                                          936.000000
                 2012.783000
                             113.172000
                                               6.723200
                                                           1.698083e+05
                                                                        82.956376
                                                                                          58.985043
                 3.205962
                             18.810908
                                               0.945429
                                                           1.887626e+05
                                                                        103.253540
                                                                                          17.194757
                 2006.000000 66.000000
                                               1.900000
                                                           6.100000e+01
                                                                        0.000000
                                                                                          11.000000
           25%
                 2010.000000
                             100 000000
                                               6.200000
                                                           3.630900e+04 13.270000
                                                                                          47 000000
           50%
                 2014 0000000
                             111 000000
                                               6.800000
                                                           1 107990e+05 47 985000
                                                                                          59 500000
                 2016.000000
                             123.000000
                                               7.400000
                                                           2.399098e+05
                                                                        113.715000
                                                                                          72.000000
                 2016.000000 191.000000
                                               9.000000
                                                           1.791916e+06 936.630000
                                                                                          100.000000
In [29]: col = df.columns
          type(col)
Out[29]: pandas.core.indexes.base.Index
In [30]: type(list(col))
Out[30]: list
In [31]: # Renaming the Columns
          col
Out[31]: Index(['Title', 'Genre', 'Description', 'Director', 'Actors', 'Year', 'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',
                 'Metascore'],
                dtype='object')
```

The columns can also be manipulated using df.columns

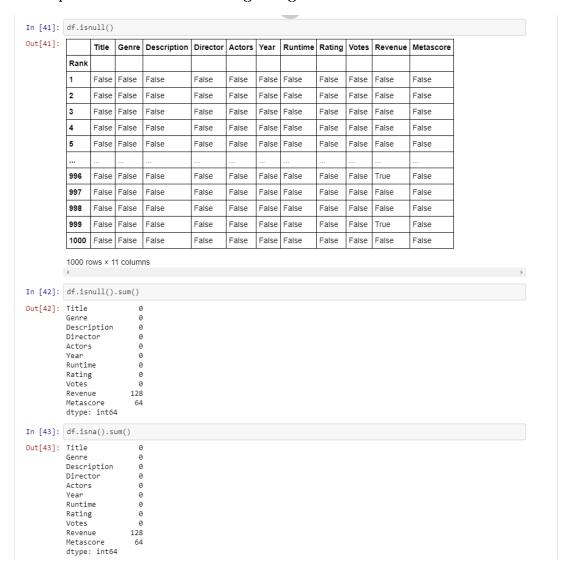
./media/image11.png

The columns can also be manipulated using df.columns

Checking for NULL values in the dataset

1. To check for NULL values in the dataset use the isnull() or isna() cmd

2. To perform sum use the isnull().sum() cmd.



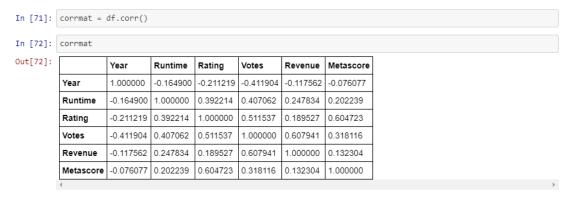
Gathering Valuable insights into the dataset

- 1. To get the total counts of each category use the value_counts () cmd
- 2. To check check for uniqueness in the data use the unique (). This would return the number of unique values in a particular Pandas Series.

```
In [65]: df['Genre'].describe()
Out[65]: count
                                      1000
                                        207
         unique
                  Action,Adventure,Sci-Fi
         top
         freq
         Name: Genre, dtype: object
In [66]: df['Genre'].value_counts().head(10)
Out[66]: Action, Adventure, Sci-Fi
         Drama
         Comedy, Drama, Romance
                                        35
         Comedy
         Drama, Romance
                                        31
         Animation, Adventure, Comedy
         Action, Adventure, Fantasy
         Comedy,Drama
         Comedy, Romance
                                        26
         Crime,Drama,Thriller
         Name: Genre, dtype: int64
In [69]: len(df['Genre'].unique())
Out[69]: 207
```

The Correlation Matrix

The correlation matrix shows the relationship between various features and can be implemented using the corr () method.



Bibliography

- 1. TakenMind Course [Udemy]
- 2. Google
- 3. Stack Overflow
- 4. Wikipedia