Data Cleaning Practical Examples

Working With Columns Names

- Get Random Dataset here https://www.generatedata.com/ (https://www.generatedata.com/)

Outline

```
+ How to check columns
+ How to rename columns
+ How to put underscore in all columns
+ How to replace a character or empty space in column names
+ How to uppercase/lowercase columns
+ How to select all columns except one
+ How to select columns of a particular order or phrase(df.filter)
+ How to select a group of column name
```

In [1]:
```
# Load Dataset
import pandas as pd
```

In [2]:
```
# Load Dataset
df = pd.read_csv("raw_dataset.csv")
```

In [4]:
```
# Firt Rows
df.head(5)
```

Out[4]:

	First Name	Last name	Age	SALARY	STREET Address1	STREET Address2	STREET Address3	
0	Joel	Padilla	10/28/2019	$92.32	431-6530 Eu, Rd.	364-2264 Augue Rd.	P.O. Box 864, 3882 Orci Street	eu@
1	Fritz	Tyler	09/27/2019	$83.91	Ap #377-2267 Ac Av.	979-2228 Vel Ave	9865 Eu Av.	est.ac.mattis@malesuadafri
2	Wing	Phelps	02/18/2019	$17.15	Ap #545-5786 Pulvinar Ave	Ap #973-5781 Sagittis Avenue	9959 Ut St.	dolor@c
3	Ryan	Ross	05/21/2019	$45.97	634-7858 Id Road	907-8824 Fringilla Ave	318-5271 In Ave	interdum.libero.dui@vitae
4	Drake	Day	01/09/2020	$84.38	999-8221 Tempor, St.	297-6939 Turpis. Ave	P.O. Box 638, 6932 Laoreet Rd.	nulla.Integer.vulputate@li

In [5]:
```
# Columns
df.columns
```

Out[5]:
```
Index(['First Name', 'Last name', 'Age', 'SALARY', 'STREET Address1',
       'STREET Address2', 'STREET Address3', 'email'],
      dtype='object')
```

In [6]:
```
## Features of Columns
dir(df.columns)
```

```
 '__delattr__',
 '__dict__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__init__',
 '__init_subclass__',
 '__invert__',
 '__iter__',
 '__le__',
 '__len__',
```

In [8]:
```
### Get The Columns As an Array
df.columns.values
```

Out[8]:
```
array(['First Name', 'Last name', 'Age', 'SALARY', 'STREET Address1',
       'STREET Address2', 'STREET Address3', 'email'], dtype=object)
```

In [9]:
```
### Get The Columns As  List
df.columns.tolist()
```

Out[9]:
```
['First Name',
 'Last name',
 'Age',
 'SALARY',
 'STREET Address1',
 'STREET Address2',
 'STREET Address3',
 'email']
```

In [10]:
```
### To View Columns Names
df.columns.view()
```

Out[10]:
```
Index(['First Name', 'Last name', 'Age', 'SALARY', 'STREET Address1',
       'STREET Address2', 'STREET Address3', 'email'],
      dtype='object')
```

In [11]:
```
### To View a Summary of the Column Names
df.columns.summary()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [11], in <cell line: 2>()
      1 ### To View a Summary of the Column Names
----> 2 df.columns.summary()

AttributeError: 'Index' object has no attribute 'summary'
```

In [12]:
```
# Convert the Column Names To Series/ DataFrame
df.columns.to_series()
```

Out[12]:
```
First Name              First Name
Last name                Last name
Age                            Age
SALARY                      SALARY
STREET Address1    STREET Address1
STREET Address2    STREET Address2
STREET Address3    STREET Address3
email                        email
dtype: object
```

In [13]:
```
# Convert the Column Names To DataFrame
df.columns.to_frame()
```

Out[13]:

	0
First Name	First Name
Last name	Last name
Age	Age
SALARY	SALARY
STREET Address1	STREET Address1
STREET Address2	STREET Address2
STREET Address3	STREET Address3
email	email

In [14]:
```
# Check to see if column names contains a phrase
df.columns.contains('First Name')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [14], in <cell line: 2>()
      1 # Check to see if column names contains a phrase
----> 2 df.columns.contains('First Name')

AttributeError: 'Index' object has no attribute 'contains'
```

In [15]:
```
# Check to see if column names are duplicated
df.columns.duplicated()
```

Out[15]:
```
array([False, False, False, False, False, False, False, False])
```

In [16]:
```
### Attributes and Methods of Str
dir(df.columns.str)
```

Out[16]:
```
['__annotations__',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__frozen',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_data',
 '_doc_args',
 '_freeze',
 '_get_series_list',
 '_index',
 '_inferred_dtype',
 '_is_categorical',
 '_is_string',
 '_name',
 '_orig',
 '_parent',
 '_validate',
 '_wrap_result',
 'capitalize',
 'casefold',
 'cat',
 'center',
 'contains',
 'count',
 'decode',
 'encode',
 'endswith',
 'extract',
 'extractall',
 'find',
 'findall',
 'fullmatch',
```

```
 'get',
 'get_dummies',
 'index',
 'isalnum',
 'isalpha',
 'isdecimal',
 'isdigit',
 'islower',
 'isnumeric',
 'isspace',
 'istitle',
 'isupper',
 'join',
 'len',
 'ljust',
 'lower',
 'lstrip',
 'match',
 'normalize',
 'pad',
 'partition',
 'removeprefix',
 'removesuffix',
 'repeat',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'slice',
 'slice_replace',
 'split',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'wrap',
 'zfill']
```

In [17]:
```
### Making Column Name Lower Case
df.columns.str.lower()
```

Out[17]:
```
Index(['first name', 'last name', 'age', 'salary', 'street address1',
       'street address2', 'street address3', 'email'],
      dtype='object')
```

In [18]:
```
### Making Column Name Upper Case
df.columns.str.upper()
```

Out[18]:
```
Index(['FIRST NAME', 'LAST NAME', 'AGE', 'SALARY', 'STREET ADDRESS1',
       'STREET ADDRESS2', 'STREET ADDRESS3', 'EMAIL'],
      dtype='object')
```

In [19]:
```
### Making Column Name Title Case
df.columns.str.title()
```

Out[19]:
```
Index(['First Name', 'Last Name', 'Age', 'Salary', 'Street Address1',
       'Street Address2', 'Street Address3', 'Email'],
      dtype='object')
```

In [20]:
```
### Replacing Empty spaces with underscore
df.columns.str.replace(' ','_')
```

Out[20]:
```
Index(['First_Name', 'Last_name', 'Age', 'SALARY', 'STREET_Address1',
       'STREET_Address2', 'STREET_Address3', 'email'],
      dtype='object')
```

In [21]:
```
### Renaming Column Name
df.rename(columns={'Age':'Date of Birth'})
```

Out[21]:

	First Name	Last name	Date of Birth	SALARY	STREET Address1	STREET Address2	STREET Address3	
0	Joel	Padilla	10/28/2019	$92.32	431-6530 Eu, Rd.	364-2264 Augue Rd.	P.O. Box 864, 3882 Orci Street	
1	Fritz	Tyler	09/27/2019	$83.91	Ap #377-2267 Ac Av.	979-2228 Vel Ave	9865 Eu Av.	est.ac.mattis@malesu
2	Wing	Phelps	02/18/2019	$17.15	Ap #545-5786 Pulvinar Ave	Ap #973-5781 Sagittis Avenue	9959 Ut St.	dol
3	Ryan	Ross	05/21/2019	$45.97	634-7858 Id Road	907-8824 Fringilla Ave	318-5271 In Ave	interdum.libero.dui@
4	Drake	Day	01/09/2020	$84.38	999-8221 Tempor, St.	297-6939 Turpis. Ave	P.O. Box 638, 6932 Laoreet Rd.	nulla.Integer.vulputa
...	...	...	...	...	...	...	...	
95	Victor	Hobbs	05/24/2019	$54.56	4034 Vitae St.	P.O. Box 930, 1683 Eu Rd.	P.O. Box 181, 3360 Mus. Rd.	ipsum@dict
96	Neil	Bradford	02/07/2020	$74.52	1434 Aliquet, Street	956-6627 Nunc Av.	Ap #727-6109 Sapien. Av.	sapien.Nunc@euodi
97	Noble	Conrad	10/29/2019	$43.99	Ap #173-7049 Eget, St.	Ap #620-2512 Ut Street	8768 Aenean St.	tellus.Nunc.lectu
98	Brody	Whitaker	08/09/2018	$96.24	Ap #371-9803 Aliquam Rd.	8892 Euismod Street	Ap #201-659 Libero. Street	non.dapibus.rutrum@
99	Alden	Mccormick	07/27/2019	$2.66	Ap #375-1139 Risus. Road	7259 Duis Avenue	955-4058 Maecenas St.	ut.erat@a

100 rows × 8 columns

In [22]:
```
### Renaming Column Name /Inplace
df.rename(columns={'Age':'Date of Birth'},inplace=True)
```

In [23]:
```
df.columns
```

Out[23]:
```
Index(['First Name', 'Last name', 'Date of Birth', 'SALARY', 'STREET Address
1',
       'STREET Address2', 'STREET Address3', 'email'],
      dtype='object')
```

In [24]:
```
len(df.columns.values)
```

Out[24]:
```
8
```

In [25]:
```
# Renaming Column Names using select values
df.columns.values[7] = 'Email Address'
```

In [26]:
```
df.columns
```

Out[26]:
```
Index(['First Name', 'Last name', 'Date of Birth', 'SALARY', 'STREET Address
1',
       'STREET Address2', 'STREET Address3', 'Email Address'],
      dtype='object')
```

In [27]:
```
### Selecting All Columns Except One
df.columns[df.columns != 'SALARY']
```

Out[27]:
```
Index(['First Name', 'Last name', 'Date of Birth', 'STREET Address1',
       'STREET Address2', 'STREET Address3', 'Email Address'],
      dtype='object')
```

In [28]:
```
### Selecting All Columns Except One
df.loc[:, df.columns != 'SALARY'].columns
```

Out[28]:
```
Index(['First Name', 'Last name', 'Date of Birth', 'STREET Address1',
       'STREET Address2', 'STREET Address3', 'Email Address'],
      dtype='object')
```

In [29]:
```
# Select Column Names Except One Using Difference
df.columns.difference(['SALARY'])
```

Out[29]:
```
Index(['Date of Birth', 'Email Address', 'First Name', 'Last name',
       'STREET Address1', 'STREET Address2', 'STREET Address3'],
      dtype='object')
```

In [30]:
```
# Select Column Names Except One Using Negation of isin
df.loc[:,~df.columns.isin(['SALARY'])].columns
```

Out[30]:
```
Index(['First Name', 'Last name', 'Date of Birth', 'STREET Address1',
       'STREET Address2', 'STREET Address3', 'Email Address'],
      dtype='object')
```

In [31]:
```
### Select Column Names that Begins with a Word or Character
df.filter(like='STREET').columns
```

Out[31]:
```
Index(['STREET Address1', 'STREET Address2', 'STREET Address3'], dtype='objec
t')
```

In [32]:
```
### Select Column Names that Begins with a Word or Character
df.loc[:,df.columns.str.startswith('STREET')].columns
```

Out[32]:
```
Index(['STREET Address1', 'STREET Address2', 'STREET Address3'], dtype='objec
t')
```

In [33]:
```
### Select Column Names that ENDS with a Word or Character
df.loc[:,df.columns.str.endswith('ame')].columns
```

Out[33]:
```
Index(['First Name', 'Last name'], dtype='object')
```

In [34]:
```
### Select Column Names that ENDS with a Word or Character Using Filter and Re
df.filter(regex='ame$',axis=1).columns
```

Out[34]:
```
Index(['First Name', 'Last name'], dtype='object')
```

In [35]:
```
### Select A Group of Column Names
df.columns.values[0:4]
```

Out[35]:
```
array(['First Name', 'Last name', 'Date of Birth', 'SALARY'], dtype=object)
```

In [36]:
```
### Select A Group of Column Names
df.columns[0:4]
```

Out[36]:
```
Index(['First Name', 'Last name', 'Date of Birth', 'SALARY'], dtype='object')
```

In [ ]:
```

```

In [ ]:
```

```