In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_excel(r"C:\Users\kallzz\Desktop\Data Analytics Stuff\Data Analyst - Boot Camp\
df
```

Out[2]:

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact	Not_Useful_(
0	1001	Frodo	Baggins	123-545-5421	123 Shire Lane, Shire	Yes	No	
1	1002	Abed	Nadir	123/643/9775	93 West Main Street	No	Yes	
2	1003	Walter	/White	7066950392	298 Drugs Driveway	N	NaN	
3	1004	Dwight	Schrute	123-543-2345	980 Paper Avenue, Pennsylvania, 18503	Yes	Y	
4	1005	Jon	Snow	876|678|3469	123 Dragons Road	Y	No	
5	1006	Ron	Swanson	304-762-2467	768 City Parkway	Yes	Yes	
6	1007	Jeff	Winger	NaN	1209 South Street	No	No	
7	1008	Sherlock	Holmes	876|678|3469	98 Clue Drive	N	No	
8	1009	Gandalf	NaN	N/a	123 Middle Earth	Yes	NaN	
9	1010	Peter	Parker	123-545-5421	25th Main Street, New York	Yes	No	
10	1011	Samwise	Gamgee	NaN	612 Shire Lane, Shire	Yes	No	
11	1012	Harry	...Potter	7066950392	2394 Hogwarts Avenue	Y	NaN	
12	1013	Don	Draper	123-543-2345	2039 Main Street	Yes	N	
13	1014	Leslie	Knope	876|678|3469	343 City Parkway	Yes	No	
14	1015	Toby	Flenderson_	304-762-2467	214 HR Avenue	N	No	
15	1016	Ron	Weasley	123-545-5421	2395 Hogwarts Avenue	No	N	
16	1017	Michael	Scott	123/643/9775	121 Paper Avenue, Pennsylvania	Yes	No	
17	1018	Clark	Kent	7066950392	3498 Super Lane	Y	NaN	
18	1019	Creed	Braton	N/a	N/a	N/a	Yes	
19	1020	Anakin	Skywalker	876|678|3469	910 Tatooine Road, Tatooine	Yes	N	
20	1020	Anakin	Skywalker	876|678|3469	910 Tatooine Road, Tatooine	Yes	N	

In [3]:
```
df = df.drop_duplicates()
df.tail()
```

Out[3]:

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact	Not_Useful_C
15	1016	Ron	Weasley	123-545-5421	2395 Hogwarts Avenue	No	N	
16	1017	Michael	Scott	123/643/9775	121 Paper Avenue, Pennsylvania	Yes	No	
17	1018	Clark	Kent	7066950392	3498 Super Lane	Y	NaN	
18	1019	Creed	Braton	N/a	N/a	N/a	Yes	
19	1020	Anakin	Skywalker	876|678|3469	910 Tatooine Road, Tatooine	Yes	N	

In [4]:
```
df = df.drop(columns = 'Not_Useful_Column')
```

In [5]:
```
df.head()
```

Out[5]:

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact
0	1001	Frodo	Baggins	123-545-5421	123 Shire Lane, Shire	Yes	No
1	1002	Abed	Nadir	123/643/9775	93 West Main Street	No	Yes
2	1003	Walter	/White	7066950392	298 Drugs Driveway	N	NaN
3	1004	Dwight	Schrute	123-543-2345	980 Paper Avenue, Pennsylvania, 18503	Yes	Y
4	1005	Jon	Snow	876|678|3469	123 Dragons Road	Y	No

Strip

In [6]:
```
# need to perform the strip iteratively to remove unwanted characters at head and tail plac
# df['Last_Name'].str.strip('/')
# df['Last_Name'].str.strip('...')
df['Last_Name'].str.strip('_')
```

Out[6]:
```
0         Baggins
1           Nadir
2          /White
3         Schrute
4            Snow
5         Swanson
6          Winger
7          Holmes
8             NaN
9          Parker
10         Gamgee
11      ...Potter
12         Draper
13          Knope
14     Flenderson
15        Weasley
16          Scott
17           Kent
18         Braton
19      Skywalker
Name: Last_Name, dtype: object
```

In [7]:
```
df['Last_Name'].str.strip(['_','...']) # passing paramaters as a list will not work
```

Out[7]:
```
0     NaN
1     NaN
2     NaN
3     NaN
4     NaN
5     NaN
6     NaN
7     NaN
8     NaN
9     NaN
10    NaN
11    NaN
12    NaN
13    NaN
14    NaN
15    NaN
16    NaN
17    NaN
18    NaN
19    NaN
Name: Last_Name, dtype: float64
```

In [8]:
```
# regular exp simplifies the process
df['Last_Name'].str.strip('123._/')
```

Out[8]:
```
0        Baggins
1          Nadir
2          White
3        Schrute
4           Snow
5        Swanson
6         Winger
7         Holmes
8            NaN
9         Parker
10        Gamgee
11        Potter
12        Draper
13         Knope
14    Flenderson
15       Weasley
16         Scott
17          Kent
18        Braton
19     Skywalker
Name: Last_Name, dtype: object
```

In [9]:
```
# assign the transformed data to that specific column
df['Last_Name'] = df['Last_Name'].str.strip('123._/')
df.head()
```

Out[9]:

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact
0	1001	Frodo	Baggins	123-545-5421	123 Shire Lane, Shire	Yes	No
1	1002	Abed	Nadir	123/643/9775	93 West Main Street	No	Yes
2	1003	Walter	White	7066950392	298 Drugs Driveway	N	NaN
3	1004	Dwight	Schrute	123-543-2345	980 Paper Avenue, Pennsylvania, 18503	Yes	Y
4	1005	Jon	Snow	876|678|3469	123 Dragons Road	Y	No

Replace

In [10]:
```
# convert phonenumber into xxx-xxx-xxxx format.
# df['Phone_Number'].str.replace('-','')
df['Phone_Number'].str.replace('/','')
```

Out[10]:
```
0     123-545-5421
1       1236439775
2              NaN
3     123-543-2345
4     876|678|3469
5     304-762-2467
6              NaN
7     876|678|3469
8               Na
9     123-545-5421
10             NaN
11             NaN
12    123-543-2345
13    876|678|3469
14    304-762-2467
15    123-545-5421
16      1236439775
17             NaN
18              Na
19    876|678|3469
Name: Phone_Number, dtype: object
```

In [11]:
```
# replace all the characters with '' and then split the string and add '-'
# reg expression

df['Phone_Number'].str.replace('[^a-zA-Z0-9]','')
```

```
C:\Users\kallzz\AppData\Local\Temp\ipykernel_29784\220277308.py:4: FutureWarning: The defa
ult value of regex will change from True to False in a future version.
  df['Phone_Number'].str.replace('[^a-zA-Z0-9]','')
```

Out[11]:
```
0     1235455421
1     1236439775
2            NaN
3     1235432345
4     8766783469
5     3047622467
6            NaN
7     8766783469
8             Na
9     1235455421
10           NaN
11           NaN
12    1235432345
13    8766783469
14    3047622467
15    1235455421
16    1236439775
17           NaN
18            Na
19    8766783469
Name: Phone_Number, dtype: object
```

In [12]:
```
df['Phone_Number'] = df['Phone_Number'].str.replace('[^a-zA-Z0-9]','')
```

```
C:\Users\kallzz\AppData\Local\Temp\ipykernel_29784\1099693601.py:1: FutureWarning: The def
ault value of regex will change from True to False in a future version.
  df['Phone_Number'] = df['Phone_Number'].str.replace('[^a-zA-Z0-9]','')
```

In [13]:
```
# lambda to change phone number in a specified format
df['Phone_Number'].apply(lambda x: x[0:3] + '-' + x[3:6] + '-' + x[6:10])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[13], line 2
      1 # lambda to change phone number in a specified format
----> 2 df['Phone_Number'].apply(lambda x: x[0:3] + '-' + x[3:6] + '-' + x[6:10])

File ~\anaconda3\lib\site-packages\pandas\core\series.py:4771, in Series.apply(self, func,
convert_dtype, args, **kwargs)
   4661 def apply(
   4662     self,
   4663     func: AggFuncType,
   (...)
   4666     **kwargs,
   4667 ) -> DataFrame | Series:
   4668     """
   4669     Invoke function on values of Series.
   4670 
   (...)
   4769     dtype: float64
   4770     """
-> 4771     return SeriesApply(self, func, convert_dtype, args, kwargs).apply()

File ~\anaconda3\lib\site-packages\pandas\core\apply.py:1123, in SeriesApply.apply(self)
   1120     return self.apply_str()
   1122 # self.f is Callable
-> 1123 return self.apply_standard()

File ~\anaconda3\lib\site-packages\pandas\core\apply.py:1174, in SeriesApply.apply_standar
d(self)
   1172     else:
   1173         values = obj.astype(object)._values
-> 1174         mapped = lib.map_infer(
   1175             values,
   1176             f,
   1177             convert=self.convert_dtype,
   1178         )
   1180 if len(mapped) and isinstance(mapped[0], ABCSeries):
   1181     # GH#43986 Need to do list(mapped) in order to get treated as nested
   1182     #  See also GH#25959 regarding EA support
   1183     return obj._constructor_expanddim(list(mapped), index=obj.index)

File ~\anaconda3\lib\site-packages\pandas\_libs\lib.pyx:2924, in pandas._libs.lib.map_infe
r()

Cell In[13], line 2, in <lambda>(x)
      1 # lambda to change phone number in a specified format
----> 2 df['Phone_Number'].apply(lambda x: x[0:3] + '-' + x[3:6] + '-' + x[6:10])

TypeError: 'float' object is not subscriptable
```

In [14]:
```
df['Phone_Number'] = df['Phone_Number'].apply(lambda x: str(x))
```

In [15]:
```
df['Phone_Number'].apply(lambda x: x[0:3] + '-' + x[3:6] + '-' + x[6:10])
```

Out[15]:
```
0     123-545-5421
1     123-643-9775
2            nan--
3     123-543-2345
4     876-678-3469
5     304-762-2467
6            nan--
7     876-678-3469
8             Na--
9     123-545-5421
10           nan--
11           nan--
12    123-543-2345
13    876-678-3469
14    304-762-2467
15    123-545-5421
16    123-643-9775
17           nan--
18            Na--
19    876-678-3469
Name: Phone_Number, dtype: object
```

In [16]:
```
df['Phone_Number'] = df['Phone_Number'].apply(lambda x: x[0:3] + '-' + x[3:6] + '-' + x[6:1
```

In [17]:
```
df.head()
```

Out[17]:

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact
0	1001	Frodo	Baggins	123-545-5421	123 Shire Lane, Shire	Yes	No
1	1002	Abed	Nadir	123-643-9775	93 West Main Street	No	Yes
2	1003	Walter	White	nan--	298 Drugs Driveway	N	NaN
3	1004	Dwight	Schrute	123-543-2345	980 Paper Avenue, Pennsylvania, 18503	Yes	Y
4	1005	Jon	Snow	876-678-3469	123 Dragons Road	Y	No

In [18]:
```
df['Phone_Number'] = df['Phone_Number'].str.replace('nan--', '')
```

In [19]:
```
df['Phone_Number'] = df['Phone_Number'].str.replace('Na--', '')
```

In [20]:
```
df.head()
```

Out[20]:

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact
0	1001	Frodo	Baggins	123-545-5421	123 Shire Lane, Shire	Yes	No
1	1002	Abed	Nadir	123-643-9775	93 West Main Street	No	Yes
2	1003	Walter	White		298 Drugs Driveway	N	NaN
3	1004	Dwight	Schrute	123-543-2345	980 Paper Avenue, Pennsylvania, 18503	Yes	Y
4	1005	Jon	Snow	876-678-3469	123 Dragons Road	Y	No

Split

In [21]:
```
# Address column has street, state and zipcode data combination
df['Address'].str.split(',', 1)
```

```
C:\Users\kallzz\AppData\Local\Temp\ipykernel_29784\2477334801.py:2: FutureWarning: In a fu
ture version of pandas all arguments of StringMethods.split except for the argument 'pat'
will be keyword-only.
  df['Address'].str.split(',', 1)
```

Out[21]:
```
0                      [123 Shire Lane,  Shire]
1                         [93 West Main Street]
2                          [298 Drugs Driveway]
3      [980 Paper Avenue,  Pennsylvania, 18503]
4                            [123 Dragons Road]
5                            [768 City Parkway]
6                           [1209 South Street]
7                               [98 Clue Drive]
8                            [123 Middle Earth]
9                 [25th Main Street,  New York]
10                     [612 Shire Lane,  Shire]
11                       [2394 Hogwarts Avenue]
12                           [2039 Main Street]
13                           [343 City Parkway]
14                              [214 HR Avenue]
15                       [2395 Hogwarts Avenue]
16            [121 Paper Avenue,  Pennsylvania]
17                            [3498 Super Lane]
18                                        [N/a]
19             [910 Tatooine Road,  Tatooine]
Name: Address, dtype: object
```

In [22]:
```
df['Address'].str.split(',', 1, expand = True)
```

```
C:\Users\kallzz\AppData\Local\Temp\ipykernel_29784\1577666541.py:1: FutureWarning: In a fu
ture version of pandas all arguments of StringMethods.split except for the argument 'pat'
will be keyword-only.
  df['Address'].str.split(',', 1, expand = True)
```

Out[22]:

	0	1
0	123 Shire Lane	Shire
1	93 West Main Street	None
2	298 Drugs Driveway	None
3	980 Paper Avenue	Pennsylvania, 18503
4	123 Dragons Road	None
5	768 City Parkway	None
6	1209 South Street	None
7	98 Clue Drive	None
8	123 Middle Earth	None
9	25th Main Street	New York
10	612 Shire Lane	Shire
11	2394 Hogwarts Avenue	None
12	2039 Main Street	None
13	343 City Parkway	None
14	214 HR Avenue	None
15	2395 Hogwarts Avenue	None
16	121 Paper Avenue	Pennsylvania
17	3498 Super Lane	None
18	N/a	None
19	910 Tatooine Road	Tatooine

In [23]:
```
df['Address'].str.split(',', 2, expand = True)
```

```
C:\Users\kallzz\AppData\Local\Temp\ipykernel_29784\2782482359.py:1: FutureWarning: In a fu
ture version of pandas all arguments of StringMethods.split except for the argument 'pat'
will be keyword-only.
  df['Address'].str.split(',', 2, expand = True)
```

Out[23]:

	0	1	2
0	123 Shire Lane	Shire	None
1	93 West Main Street	None	None
2	298 Drugs Driveway	None	None
3	980 Paper Avenue	Pennsylvania	18503
4	123 Dragons Road	None	None
5	768 City Parkway	None	None
6	1209 South Street	None	None
7	98 Clue Drive	None	None
8	123 Middle Earth	None	None
9	25th Main Street	New York	None
10	612 Shire Lane	Shire	None
11	2394 Hogwarts Avenue	None	None
12	2039 Main Street	None	None
13	343 City Parkway	None	None
14	214 HR Avenue	None	None
15	2395 Hogwarts Avenue	None	None
16	121 Paper Avenue	Pennsylvania	None
17	3498 Super Lane	None	None
18	N/a	None	None
19	910 Tatooine Road	Tatooine	None

In [24]:
```
df[['Street_Address', 'State', 'Zip_Code']] = df['Address'].str.split(',', 2, expand = True
```

```
C:\Users\kallzz\AppData\Local\Temp\ipykernel_29784\3034702943.py:1: FutureWarning: In a fu
ture version of pandas all arguments of StringMethods.split except for the argument 'pat'
will be keyword-only.
  df[['Street_Address', 'State', 'Zip_Code']] = df['Address'].str.split(',', 2, expand = T
rue)
```

In [25]:
```
df.head()
```

Out[25]:

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact	Street_Addres
0	1001	Frodo	Baggins	123-545-5421	123 Shire Lane, Shire	Yes	No	123 Shire Lan
1	1002	Abed	Nadir	123-643-9775	93 West Main Street	No	Yes	93 West Mai Stree
2	1003	Walter	White		298 Drugs Driveway	N	NaN	298 Drug Drivewa
3	1004	Dwight	Schrute	123-543-2345	980 Paper Avenue, Pennsylvania, 18503	Yes	Y	980 Pape Avenu
4	1005	Jon	Snow	876-678-3469	123 Dragons Road	Y	No	123 Dragon Roa

In [26]:
```
df = df.drop(columns = 'Address')
```

In [27]:
```
df.head()
```

Out[27]:

	CustomerID	First_Name	Last_Name	Phone_Number	Paying Customer	Do_Not_Contact	Street_Address	State
0	1001	Frodo	Baggins	123-545-5421	Yes	No	123 Shire Lane	Shire
1	1002	Abed	Nadir	123-643-9775	No	Yes	93 West Main Street	None
2	1003	Walter	White		N	NaN	298 Drugs Driveway	None
3	1004	Dwight	Schrute	123-543-2345	Yes	Y	980 Paper Avenue	Pennsylvania
4	1005	Jon	Snow	876-678-3469	Y	No	123 Dragons Road	None

fill NaN and None values

In [28]:
```
df = df.fillna('')
```

In [29]:
```
df['Paying Customer'] = df['Paying Customer'].str.replace('Yes', 'Y')
```

In [30]:
```
df['Paying Customer'] = df['Paying Customer'].str.replace('No', 'N')
```

In [31]:
```
df.head()
```

Out[31]:

	CustomerID	First_Name	Last_Name	Phone_Number	Paying Customer	Do_Not_Contact	Street_Address	State
0	1001	Frodo	Baggins	123-545-5421	Y	No	123 Shire Lane	Shire
1	1002	Abed	Nadir	123-643-9775	N	Yes	93 West Main Street	
2	1003	Walter	White		N		298 Drugs Driveway	
3	1004	Dwight	Schrute	123-543-2345	Y	Y	980 Paper Avenue	Pennsylvania
4	1005	Jon	Snow	876-678-3469	Y	No	123 Dragons Road	

In [32]:
```
df['Do_Not_Contact'] = df['Do_Not_Contact'].str.replace('Yes', 'Y')
```

In [33]:
```
df['Do_Not_Contact'] = df['Do_Not_Contact'].str.replace('No', 'N')
```

In [34]:
```
df.head()
```

Out[34]:

	CustomerID	First_Name	Last_Name	Phone_Number	Paying Customer	Do_Not_Contact	Street_Address	State
0	1001	Frodo	Baggins	123-545-5421	Y	N	123 Shire Lane	Shire
1	1002	Abed	Nadir	123-643-9775	N	Y	93 West Main Street	
2	1003	Walter	White		N		298 Drugs Driveway	
3	1004	Dwight	Schrute	123-543-2345	Y	Y	980 Paper Avenue	Pennsylvania
4	1005	Jon	Snow	876-678-3469	Y	N	123 Dragons Road	

Prepare a dataset eligible for contacting the customer

- remove Do_Not_Contact = Y rows based on index
- remove rows that have no values for Phone_Number

In [35]:
```
for x in df.index:
    if df.loc[x, "Do_Not_Contact"] == 'Y':
        df.drop(x, inplace = True)

df
```

Out[35]:

	CustomerID	First_Name	Last_Name	Phone_Number	Paying Customer	Do_Not_Contact	Street_Address	Stat
0	1001	Frodo	Baggins	123-545-5421	Y	N	123 Shire Lane	Shi
2	1003	Walter	White		N		298 Drugs Driveway	
4	1005	Jon	Snow	876-678-3469	Y	N	123 Dragons Road	
6	1007	Jeff	Winger		N	N	1209 South Street	
7	1008	Sherlock	Holmes	876-678-3469	N	N	98 Clue Drive	
8	1009	Gandalf			Y		123 Middle Earth	
9	1010	Peter	Parker	123-545-5421	Y	N	25th Main Street	New Yo
10	1011	Samwise	Gamgee		Y	N	612 Shire Lane	Shi
11	1012	Harry	Potter		Y		2394 Hogwarts Avenue	
12	1013	Don	Draper	123-543-2345	Y	N	2039 Main Street	
13	1014	Leslie	Knope	876-678-3469	Y	N	343 City Parkway	
14	1015	Toby	Flenderson	304-762-2467	N	N	214 HR Avenue	
15	1016	Ron	Weasley	123-545-5421	N	N	2395 Hogwarts Avenue	
16	1017	Michael	Scott	123-643-9775	Y	N	121 Paper Avenue	Pennsylvan
17	1018	Clark	Kent		Y		3498 Super Lane	
19	1020	Anakin	Skywalker	876-678-3469	Y	N	910 Tatooine Road	Tatooir

In [36]:
```
for x in df.index:
    if df.loc[x, "Phone_Number"] == '':
        df.drop(x, inplace = True)

df
```

Out[36]:

	CustomerID	First_Name	Last_Name	Phone_Number	Paying Customer	Do_Not_Contact	Street_Address	Stat
0	1001	Frodo	Baggins	123-545-5421	Y	N	123 Shire Lane	Shi
4	1005	Jon	Snow	876-678-3469	Y	N	123 Dragons Road	
7	1008	Sherlock	Holmes	876-678-3469	N	N	98 Clue Drive	
9	1010	Peter	Parker	123-545-5421	Y	N	25th Main Street	New Yo
12	1013	Don	Draper	123-543-2345	Y	N	2039 Main Street	
13	1014	Leslie	Knope	876-678-3469	Y	N	343 City Parkway	
14	1015	Toby	Flenderson	304-762-2467	N	N	214 HR Avenue	
15	1016	Ron	Weasley	123-545-5421	N	N	2395 Hogwarts Avenue	
16	1017	Michael	Scott	123-643-9775	Y	N	121 Paper Avenue	Pennsylvan
19	1020	Anakin	Skywalker	876-678-3469	Y	N	910 Tatooine Road	Tatooir

In [37]:
```
df.reset_index(drop = True)
```

Out[37]:

	CustomerID	First_Name	Last_Name	Phone_Number	Paying Customer	Do_Not_Contact	Street_Address	State
0	1001	Frodo	Baggins	123-545-5421	Y	N	123 Shire Lane	Shire
1	1005	Jon	Snow	876-678-3469	Y	N	123 Dragons Road	
2	1008	Sherlock	Holmes	876-678-3469	N	N	98 Clue Drive	
3	1010	Peter	Parker	123-545-5421	Y	N	25th Main Street	New York
4	1013	Don	Draper	123-543-2345	Y	N	2039 Main Street	
5	1014	Leslie	Knope	876-678-3469	Y	N	343 City Parkway	
6	1015	Toby	Flenderson	304-762-2467	N	N	214 HR Avenue	
7	1016	Ron	Weasley	123-545-5421	N	N	2395 Hogwarts Avenue	
8	1017	Michael	Scott	123-643-9775	Y	N	121 Paper Avenue	Pennsylvani
9	1020	Anakin	Skywalker	876-678-3469	Y	N	910 Tatooine Road	Tatooine