tipsandtricks

November 5, 2022

1 01- How to find the version?

```
[ ]: import pandas as pd
     pd.__version__
```

```
[ ]: '1.4.4'
```

```
[ ]: # Another way to see the version
     pd.show_versions()
```

```
INSTALLED VERSIONS
------------------
commit           : ca60aab7340d9989d9428e11a51467658190bb6b
python           : 3.10.7.final.0
python-bits      : 64
OS               : Windows
OS-release       : 10
Version          : 10.0.22623
machine          : AMD64
processor        : Intel64 Family 6 Model 142 Stepping 11, GenuineIntel
byteorder        : little
LC_ALL           : None
LANG             : None
LOCALE           : English_United States.1252

pandas           : 1.4.4
numpy            : 1.23.2
pytz             : 2022.2.1
dateutil         : 2.8.2
setuptools       : 63.2.0
pip              : 22.2.2
Cython           : None
pytest           : None
hypothesis       : None
sphinx           : None
blosc            : None
feather          : None
```

```
xlsxwriter        : None
lxml.etree        : 4.9.1
html5lib          : None
pymysql           : None
psycopg2          : None
jinja2            : 3.1.2
IPython           : 8.4.0
pandas_datareader: None
bs4               : 4.11.1
bottleneck        : None
brotli            : None
fastparquet       : None
fsspec            : None
gcsfs             : None
markupsafe        : 2.1.1
matplotlib        : 3.5.3
numba             : None
numexpr           : None
odfpy             : None
openpyxl          : 3.0.10
pandas_gbq        : None
pyarrow           : 9.0.0
pyreadstat        : None
pyxlsb            : None
s3fs              : None
scipy             : 1.9.1
snappy            : None
sqlalchemy        : None
tables            : None
tabulate          : 0.9.0
xarray            : 2022.10.0
xlrd              : None
xlwt              : None
zstandard         : None
```

2 02- Make a dataframe

[ ]:
```
pd.DataFrame({'A column': [1, 2, 3], 'B column': [4, 5, 6], 'C column': [7, 8,␣
 ↪9]})
```

[ ]:
```
   A column  B column  C column
0         1         4         7
1         2         5         8
2         3         6         9
```

[ ]:
```
# To store the dataframe in a variable, lenght must be same for all columns
```

```
df = pd.DataFrame({'A column': [1, 2, 3], 'B column': [4, 5, 6], 'C column':␣
 ↪[7, 8, 9]})
df.head()
```

```
[ ]:    A column  B column  C column
     0         1         4         7
     1         2         5         8
     2         3         6         9
```

```
[ ]: # TO save the dataframe to a csv file
     df.to_csv('df.csv')
```

```
[ ]: # Numpy array use to create a dataframe
     import numpy as np
     arr = np.array([[1, 2, 3,4,5,6,7], [4, 5, 6,7,8,9,10], [20,10,5, 4, 5, 6,7]])
     arr
```

```
[ ]: array([[ 1,  2,  3,  4,  5,  6,  7],
            [ 4,  5,  6,  7,  8,  9, 10],
            [20, 10,  5,  4,  5,  6,  7]])
```

```
[ ]: # Convert array into dataframe
     pd.DataFrame(arr)
```

```
[ ]:     0   1  2  3  4  5   6
     0   1   2  3  4  5  6   7
     1   4   5  6  7  8  9  10
     2  20  10  5  4  5  6   7
```

```
[ ]: # Numpy array use to create a dataframe
     pd.DataFrame(np.random.rand(4, 8))
```

```
[ ]:          0         1         2         3         4         5         6  \
     0  0.281124  0.305890  0.090569  0.848709  0.367534  0.875663  0.824046
     1  0.301469  0.714043  0.807162  0.000618  0.528276  0.672933  0.031890
     2  0.679655  0.386891  0.911260  0.752729  0.421829  0.282845  0.377174
     3  0.954018  0.449426  0.044306  0.467691  0.298318  0.567485  0.402173

               7
     0  0.898672
     1  0.417293
     2  0.113582
     3  0.295882
```

```
[ ]: # To give the name of columns in the above generated dataframe
     pd.DataFrame(np.random.rand(4, 8), columns=['A', 'B', 'C', 'D', 'E', 'F', 'G',␣
      ↪'H'])
```

```
[ ]:          A         B         C         D         E         F         G  \
     0  0.159447  0.753156  0.534149  0.751298  0.747182  0.632818  0.564191
     1  0.333823  0.323243  0.967866  0.761435  0.617204  0.637864  0.191056
     2  0.149273  0.555828  0.787815  0.631400  0.704353  0.284614  0.267190
     3  0.699512  0.611107  0.272135  0.361673  0.909656  0.060159  0.843427

               H
     0  0.358113
     1  0.482137
     2  0.029075
     3  0.940424
```

3 03 How to rename columns

```
[ ]: # Create dataframe from dictionary
     df = pd.DataFrame({'Col':[1,2,3,4,5,6,7,8,9,10], 'Col2':
      ↪[11,12,13,14,15,16,17,18,19,20]})
     df.head()
```

```
[ ]:    Col  Col2
     0    1    11
     1    2    12
     2    3    13
     3    4    14
     4    5    15
```

```
[ ]: # Rename columns (Method 1)
     df.rename(columns={'Col':'Column1', 'Col2':'Column2'}, inplace=True)
     df.head()
```

```
[ ]:    Column1  Column2
     0        1       11
     1        2       12
     2        3       13
     3        4       14
     4        5       15
```

```
[ ]: # Rename columns (Method 2)
     df.columns=['Column_A', 'Column_B']
     df.head()
```

```
[ ]:    Column_A  Column_B
     0         1        11
     1         2        12
     2         3        13
     3         4        14
     4         5        15
```

[ ]:
```
# To replace any character, string, _ is removed from the column name
df.columns=df.columns.str.replace('_', ' ')
df.head()
```

[ ]:
```
   Column A  Column B
0         1        11
1         2        12
2         3        13
3         4        14
4         5        15
```

[ ]:
```
# To add _ in the datafreame column name
df.columns=df.columns.str.replace(' ', '_')
df.head()
```

[ ]:
```
   Column_A  Column_B
0         1        11
1         2        12
2         3        13
3         4        14
4         5        15
```

[ ]:
```
# Adding prefix to columns
df = df.add_prefix('Aadi_')
df.head()
```

[ ]:
```
   Aadi_Column_A  Aadi_Column_B
0              1             11
1              2             12
2              3             13
3              4             14
4              5             15
```

[ ]:
```
# Adding suffix to columns
df = df.add_suffix('_Aadi')
df.head()
```

[ ]:
```
   Aadi_Column_A_Aadi  Aadi_Column_B_Aadi
0                   1                  11
1                   2                  12
2                   3                  13
3                   4                  14
4                   5                  15
```

[ ]:
```
df.columns = ['col_A', 'col_B']
df
```

[ ]:
```
   col_A  col_B
0      1     11
1      2     12
2      3     13
3      4     14
4      5     15
5      6     16
6      7     17
7      8     18
8      9     19
9     10     20
```

4 04 Using template data

[ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns

kashti = sns.load_dataset('titanic')
kashti.head()
```

[ ]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

[ ]:
```
# Renmae the columns
kashti.rename(columns={'age': 'Age', 'sex': 'Sex'}, inplace=True)
kashti.head()
```

[ ]:
```
   survived  pclass     Sex   Age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
```

```
0    man        True  NaN  Southampton   no  False
1  woman       False    C    Cherbourg  yes  False
2  woman       False  NaN  Southampton  yes   True
3  woman       False    C  Southampton  yes  False
4    man        True  NaN  Southampton   no   True
```

[ ]:
```
# Data info
kashti.describe()
```

[ ]:
```
         survived      pclass         Age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

[ ]:
```
kashti.columns
```

[ ]:
```
Index(['survived', 'pclass', 'Sex', 'Age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

[ ]:
```
# Saving a dataset
kashti.to_csv('kashti.csv')
#pip install openpyxl
kashti.to_excel('kashti.xlsx')
```

```
---------------------------------------------------------------------------
PermissionError                           Traceback (most recent call last)
c:\Users\MUHAMMAD ADNAN\Desktop\Python_02\Day-3\tipsandtricks.ipynb Cell 26 in␣
↪<cell line: 4>()
      <a href='vscode-notebook-cell:/c%3A/Users/MUHAMMAD%20ADNAN/Desktop/
↪Python_02/Day-3/tipsandtricks.ipynb#X34sZmlsZQ%3D%3D?line=1'>2</a> kashti.
↪to_csv('kashti.csv')
      <a href='vscode-notebook-cell:/c%3A/Users/MUHAMMAD%20ADNAN/Desktop/
↪Python_02/Day-3/tipsandtricks.ipynb#X34sZmlsZQ%3D%3D?line=2'>3</a> #pip␣
↪install openpyxl
----> <a href='vscode-notebook-cell:/c%3A/Users/MUHAMMAD%20ADNAN/Desktop/
↪Python_02/Day-3/tipsandtricks.ipynb#X34sZmlsZQ%3D%3D?line=3'>4</a> kashti.
↪to_excel('kashti.xlsx')
```

```
File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\g neric.
 ↪py:2345, in NDFrame.to_excel(self, excel_writer, sheet_name, na_rep,␣
 ↪float_format, columns, header, index, index_label, startrow, startcol, engine ␣
 ↪merge_cells, encoding, inf_rep, verbose, freeze_panes, storage_options)
   2332 from pandas.io.formats.excel import ExcelFormatter
   2334 formatter = ExcelFormatter(
   2335     df,
   2336     na_rep=na_rep,
   (…)
   2343     inf_rep=inf_rep,
   2344 )
-> 2345 formatter.write(
   2346     excel_writer,
   2347     sheet_name=sheet_name,
   2348     startrow=startrow,
   2349     startcol=startcol,
   2350     freeze_panes=freeze_panes,
   2351     engine=engine,
   2352     storage_options=storage_options,
   2353 )

File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\for ats\excel.
 ↪py:888, in ExcelFormatter.write(self, writer, sheet_name, startrow, startcol, 
 ↪freeze_panes, engine, storage_options)
    884     need_save = False
    885 else:
    886     # error: Cannot instantiate abstract class 'ExcelWriter' with␣
 ↪abstract
    887     # attributes 'engine', 'save', 'supported_extensions' and␣
 ↪'write_cells'
--> 888     writer = ExcelWriter(  # type: ignore[abstract]
    889         writer, engine=engine, storage_options=storage_options
    890     )
    891     need_save = True
    893 try:

File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\exc l\_openpyxl.
 ↪py:53, in OpenpyxlWriter.__init__(self, path, engine, date_format,␣
 ↪datetime_format, mode, storage_options, if_sheet_exists, engine_kwargs,␣
 ↪**kwargs)
     49 from openpyxl.workbook import Workbook
     51 engine_kwargs = combine_kwargs(engine_kwargs, kwargs)
---> 53 super().__init__(
     54     path,
     55     mode=mode,
     56     storage_options=storage_options,
     57     if_sheet_exists=if_sheet_exists,
     58     engine_kwargs=engine_kwargs,
     59 )
     61 # ExcelWriter replaced "a" by "r+" to allow us to first read the excel␣
  ↪file from
     62 # the file and later write to it
     63 if "r+" in self.mode:  # Load from existing workbook

File c:\Users\MUHAMMAD␣
  ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\exc l\_base.
  ↪py:1106, in ExcelWriter.__init__(self, path, engine, date_format,␣
  ↪datetime_format, mode, storage_options, if_sheet_exists, engine_kwargs,␣
  ↪**kwargs)
   1102 self.handles = IOHandles(
   1103     cast(IO[bytes], path), compression={"compression": None}
   1104 )
   1105 if not isinstance(path, ExcelWriter):
-> 1106     self.handles = get_handle(
   1107         path, mode, storage_options=storage_options, is_text=False
   1108     )
   1109 self.sheets: dict[str, Any] = {}
   1110 self.cur_sheet = None

File c:\Users\MUHAMMAD␣
  ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\com ion.
  ↪py:795, in get_handle(path_or_buf, mode, encoding, compression, memory_map,␣
  ↪is_text, errors, storage_options)
    786         handle = open(
    787             handle,
    788             ioargs.mode,
   (…)
    791             newline="",
    792         )
    793     else:
    794         # Binary mode
--> 795         handle = open(handle, ioargs.mode)
    796     handles.append(handle)
    798 # Convert BytesIO or file objects passed with an encoding

PermissionError: [Errno 13] Permission denied: 'kashti.xlsx'
```

5 05 - Using your own data

[ ]:
```python
import pandas as pd
df = pd.read_csv('kashti.csv')
df.head()
```

```
[ ]:    Unnamed: 0  survived  pclass     Sex   Age  sibsp  parch     fare embarked  \
     0           0         0       3    male  22.0      1      0   7.2500        S
     1           1         1       1  female  38.0      1      0  71.2833        C
     2           2         1       3  female  26.0      0      0   7.9250        S
     3           3         1       1  female  35.0      1      0  53.1000        S
     4           4         0       3    male  35.0      0      0   8.0500        S

        class    who  adult_male deck  embark_town alive  alone
     0  Third    man        True  NaN  Southampton    no  False
     1  First  woman       False    C    Cherbourg   yes  False
     2  Third  woman       False  NaN  Southampton   yes   True
     3  First  woman       False    C  Southampton   yes  False
     4  Third    man        True  NaN  Southampton    no   True
```

```
[ ]: # Read Excell File
     df1 = pd.read_excel('kashti.xlsx')
     df1.head()
```

```
[ ]:    Unnamed: 0  survived  pclass     sex   age  sibsp  parch     fare embarked  \
     0           0         0       3    male  22.0      1      0   7.2500        S
     1           1         1       1  female  38.0      1      0  71.2833        C
     2           2         1       3  female  26.0      0      0   7.9250        S
     3           3         1       1  female  35.0      1      0  53.1000        S
     4           4         0       3    male  35.0      0      0   8.0500        S

        class    who  adult_male deck  embark_town alive  alone
     0  Third    man        True  NaN  Southampton    no  False
     1  First  woman       False    C    Cherbourg   yes  False
     2  Third  woman       False  NaN  Southampton   yes   True
     3  First  woman       False    C  Southampton   yes  False
     4  Third    man        True  NaN  Southampton    no   True
```

6 06 - Reverse Row Order

```
[ ]: import seaborn as sns
     import pandas as pd

     df = sns.load_dataset('titanic')
     df.head()
```

```
[ ]:    survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
     0         0       3    male  22.0      1      0   7.2500        S  Third
     1         1       1  female  38.0      1      0  71.2833        C  First
     2         1       3  female  26.0      0      0   7.9250        S  Third
     3         1       1  female  35.0      1      0  53.1000        S  First
     4         0       3    male  35.0      0      0   8.0500        S  Third
     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

```
[ ]: # reverse the order of the rows
     df.loc[::-1].head()
```

```
[ ]:      survived  pclass     sex   age  sibsp  parch   fare embarked   class  \
     890         0       3    male  32.0      0      0   7.75        Q   Third
     889         1       1    male  26.0      0      0  30.00        C   First
     888         0       3  female   NaN      1      2  23.45        S   Third
     887         1       1  female  19.0      0      0  30.00        S   First
     886         0       2    male  27.0      0      0  13.00        S  Second

            who  adult_male deck  embark_town alive  alone
     890    man        True  NaN   Queenstown    no   True
     889    man        True    C    Cherbourg   yes   True
     888  woman       False  NaN  Southampton    no  False
     887  woman       False    B  Southampton   yes   True
     886    man        True  NaN  Southampton    no   True
```

```
[ ]: # Reset the index
     df.loc[::-1].reset_index(drop=True).head()
```

```
[ ]:    survived  pclass     sex   age  sibsp  parch   fare embarked   class  \
     0         0       3    male  32.0      0      0   7.75        Q   Third
     1         1       1    male  26.0      0      0  30.00        C   First
     2         0       3  female   NaN      1      2  23.45        S   Third
     3         1       1  female  19.0      0      0  30.00        S   First
     4         0       2    male  27.0      0      0  13.00        S  Second

          who  adult_male deck  embark_town alive  alone
     0    man        True  NaN   Queenstown    no   True
     1    man        True    C    Cherbourg   yes   True
     2  woman       False  NaN  Southampton    no  False
     3  woman       False    B  Southampton   yes   True
     4    man        True  NaN  Southampton    no   True
```

7 07 - Reverse Column order

```
[ ]: df.loc[: , ::-1].head()
```

```
[ ]:    alone alive  embark_town deck  adult_male    who  class embarked     fare  \
     0  False    no  Southampton  NaN        True    man  Third        S   7.2500
```

```
1  False   yes    Cherbourg   C       False  woman  First         C  71.2833
2   True   yes  Southampton  NaN      False  woman  Third         S   7.9250
3  False   yes  Southampton    C      False  woman  First         S  53.1000
4   True    no  Southampton  NaN       True    man  Third         S   8.0500

   parch  sibsp   age     sex  pclass  survived
0      0      1  22.0    male       3         0
1      0      1  38.0  female       1         1
2      0      0  26.0  female       3         1
3      0      1  35.0  female       1         1
4      0      0  35.0    male       3         0
```

8 08 - Select a column by dtype

[ ]:
```
df.dtypes
```

[ ]:
```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

[ ]:
```
# Only select those have numeric type
df.select_dtypes(include=['number']).head()
```

[ ]:
```
   survived  pclass   age  sibsp  parch     fare
0         0       3  22.0      1      0   7.2500
1         1       1  38.0      1      0  71.2833
2         1       3  26.0      0      0   7.9250
3         1       1  35.0      1      0  53.1000
4         0       3  35.0      0      0   8.0500
```

[ ]:
```
# only select those have object type
df.select_dtypes(include=['object']).head()
```

```
[ ]:       sex embarked    who  embark_town alive
     0    male        S    man  Southampton    no
     1  female        C  woman    Cherbourg   yes
     2  female        S  woman  Southampton   yes
     3  female        S  woman  Southampton   yes
     4    male        S    man  Southampton    no
```

```
[ ]: # Only select those have multiple types
     df.select_dtypes(include=['number', 'object']).head()
```

```
[ ]:    survived  pclass     sex   age  sibsp  parch     fare embarked    who  \
     0         0       3    male  22.0      1      0   7.2500        S    man
     1         1       1  female  38.0      1      0  71.2833        C  woman
     2         1       3  female  26.0      0      0   7.9250        S  woman
     3         1       1  female  35.0      1      0  53.1000        S  woman
     4         0       3    male  35.0      0      0   8.0500        S    man

        embark_town alive
     0  Southampton    no
     1    Cherbourg   yes
     2  Southampton   yes
     3  Southampton   yes
     4  Southampton    no
```

```
[ ]: # Use exclude to remove the unwanted columns
     df.select_dtypes(exclude=['number']).head()
```

```
[ ]:       sex embarked  class    who  adult_male deck  embark_town alive  alone
     0    male        S  Third    man        True  NaN  Southampton    no  False
     1  female        C  First  woman       False    C    Cherbourg   yes  False
     2  female        S  Third  woman       False  NaN  Southampton   yes   True
     3  female        S  First  woman       False    C  Southampton   yes  False
     4    male        S  Third    man        True  NaN  Southampton    no   True
```

```
[ ]: # Use exclude to remove the unwanted columns
     df.select_dtypes(exclude=['object']).head()
```

```
[ ]:    survived  pclass   age  sibsp  parch     fare  class  adult_male deck  \
     0         0       3  22.0      1      0   7.2500  Third        True  NaN
     1         1       1  38.0      1      0  71.2833  First       False    C
     2         1       3  26.0      0      0   7.9250  Third       False  NaN
     3         1       1  35.0      1      0  53.1000  First       False    C
     4         0       3  35.0      0      0   8.0500  Third        True  NaN

        alone
     0  False
     1  False
     2   True
```

```
3   False
4    True
```

9 09 - Convert strings to number

```
[ ]: df = pd.DataFrame({'A': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'B': [11, 12, 13, 14,␣
      ↪15, 16, 17, 18, 19, 20]})
     df.head()
```

```
[ ]:    A   B
     0  1  11
     1  2  12
     2  3  13
     3  4  14
     4  5  15
```

```
[ ]: df.dtypes
```

```
[ ]: A    int64
     B    int64
     dtype: object
```

```
[ ]: df = pd.DataFrame({'A': ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10'],␣
      ↪'B': ['11', '12', '13', '14', '15', '16', '17', '18', '19', '20']})
     df.dtypes
```

```
[ ]: A    object
     B    object
     dtype: object
```

```
[ ]: # Convert string object into integer
     df.astype({'A': 'float64', 'B': 'int64'}).dtypes
```

```
[ ]: A    float64
     B      int64
     dtype: object
```

10 10 - Reduce dataframe size

```
[ ]: df = sns.load_dataset('titanic')
     df.shape
```

```
[ ]: (891, 15)
```

```
[ ]: # take some fraction of dataset like sample
     df.sample(frac=0.1).shape  # 10% of the dataset
```

```
[ ]: (89, 15)
```

11 11 - Copy data from clip board

```
[ ]: # Dataset dowlnoad from the internet
     import seaborn as sns
     import pandas as pd

     df = sns.load_dataset('titanic')
     df.to_excel('kashti.xlsx')
```

```
---------------------------------------------------------------------------
PermissionError                           Traceback (most recent call last)
c:\Users\MUHAMMAD ADNAN\Desktop\Python_02\Day-3\tipsandtricks.ipynb Cell 52 in␣
 ↪<cell line: 6>()
      <a href='vscode-notebook-cell:/c%3A/Users/MUHAMMAD%20ADNAN/Desktop/
 ↪Python_02/Day-3/tipsandtricks.ipynb#Y104sZmlsZQ%3D%3D?line=2'>3</a> import␣
 ↪pandas as pd
      <a href='vscode-notebook-cell:/c%3A/Users/MUHAMMAD%20ADNAN/Desktop/
 ↪Python_02/Day-3/tipsandtricks.ipynb#Y104sZmlsZQ%3D%3D?line=4'>5</a> df = sns.
 ↪load_dataset('titanic')
----> <a href='vscode-notebook-cell:/c%3A/Users/MUHAMMAD%20ADNAN/Desktop/
 ↪Python_02/Day-3/tipsandtricks.ipynb#Y104sZmlsZQ%3D%3D?line=5'>6</a> df.
 ↪to_excel('kashti.xlsx')


File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\generic.
 ↪py:2345, in NDFrame.to_excel(self, excel_writer, sheet_name, na_rep,␣
 ↪float_format, columns, header, index, index_label, startrow, startcol, engine␣
 ↪merge_cells, encoding, inf_rep, verbose, freeze_panes, storage_options)
   2332 from pandas.io.formats.excel import ExcelFormatter
   2334 formatter = ExcelFormatter(
   2335     df,
   2336     na_rep=na_rep,
   (…)
   2343     inf_rep=inf_rep,
   2344 )
-> 2345 formatter.write(
   2346     excel_writer,
   2347     sheet_name=sheet_name,
   2348     startrow=startrow,
   2349     startcol=startcol,
   2350     freeze_panes=freeze_panes,
   2351     engine=engine,
   2352     storage_options=storage_options,
   2353 )
```

```
File c:\Users\MUHAMMAD␣
↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\formats\excel.
↪py:888, in ExcelFormatter.write(self, writer, sheet_name, startrow, startcol,␣
↪freeze_panes, engine, storage_options)
    884     need_save = False
    885 else:
    886     # error: Cannot instantiate abstract class 'ExcelWriter' with␣
↪abstract
    887     # attributes 'engine', 'save', 'supported_extensions' and␣
↪'write_cells'
--> 888     writer = ExcelWriter(  # type: ignore[abstract]
    889         writer, engine=engine, storage_options=storage_options
    890     )
    891     need_save = True
    893 try:

File c:\Users\MUHAMMAD␣
↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\excel\_openpyxl.
↪py:53, in OpenpyxlWriter.__init__(self, path, engine, date_format,␣
↪datetime_format, mode, storage_options, if_sheet_exists, engine_kwargs,␣
↪**kwargs)
     49 from openpyxl.workbook import Workbook
     51 engine_kwargs = combine_kwargs(engine_kwargs, kwargs)
---> 53 super().__init__(
     54     path,
     55     mode=mode,
     56     storage_options=storage_options,
     57     if_sheet_exists=if_sheet_exists,
     58     engine_kwargs=engine_kwargs,
     59 )
     61 # ExcelWriter replaced "a" by "r+" to allow us to first read the excel␣
↪file from
     62 # the file and later write to it
     63 if "r+" in self.mode:  # Load from existing workbook

File c:\Users\MUHAMMAD␣
↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\excel\_base.
↪py:1106, in ExcelWriter.__init__(self, path, engine, date_format,␣
↪datetime_format, mode, storage_options, if_sheet_exists, engine_kwargs,␣
↪**kwargs)
   1102 self.handles = IOHandles(
   1103     cast(IO[bytes], path), compression={"compression": None}
   1104 )
   1105 if not isinstance(path, ExcelWriter):
-> 1106     self.handles = get_handle(
   1107         path, mode, storage_options=storage_options, is_text=False
   1108     )
   1109 self.sheets: dict[str, Any] = {}
   1110 self.cur_sheet = None
```

```
File c:\Users\MUHAMMAD␣
↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\common.
↪py:795, in get_handle(path_or_buf, mode, encoding, compression, memory_map,␣
↪is_text, errors, storage_options)
    786         handle = open(
    787             handle,
    788             ioargs.mode,
   (…)
    791             newline="",
    792         )
    793     else:
    794         # Binary mode
--> 795         handle = open(handle, ioargs.mode)
    796     handles.append(handle)
    798 # Convert BytesIO or file objects passed with an encoding

PermissionError: [Errno 13] Permission denied: 'kashti.xlsx'
```

```
[ ]: # Read clipboard in python
     df = pd.read_clipboard()
     df
     # To save this clipboard data into a csv file
     df.to_csv('clipboard.csv')
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
File c:\Users\MUHAMMAD␣
↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\parsers\python_pa
↪py:364, in PythonParser._infer_columns(self)
    363 try:
--> 364     line = self._buffered_line()
    366     while self.line_pos <= hr:

File c:\Users\MUHAMMAD␣
↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\parsers\python_pa
↪py:596, in PythonParser._buffered_line(self)
    595 else:
--> 596     return self._next_line()

File c:\Users\MUHAMMAD␣
↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\parsers\python_pa
↪py:696, in PythonParser._next_line(self)
    695 while True:
--> 696     orig_line = self._next_iter_line(row_num=self.pos + 1)
    697     self.pos += 1
```

```
File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\par ers\python_pa
 ↪py:760, in PythonParser._next_iter_line(self, row_num)
    759 assert self.data is not None
--> 760 line = next(self.data)
    761 # for mypy

StopIteration:

The above exception was the direct cause of the following exception:

EmptyDataError                            Traceback (most recent call last)
c:\Users\MUHAMMAD ADNAN\Desktop\Python_02\Day-3\tipsandtricks.ipynb Cell 53 in␣
 ↪<cell line: 2>()
      <a href='vscode-notebook-cell:/c%3A/Users/MUHAMMAD%20ADNAN/Desktop/
 ↪Python_02/Day-3/tipsandtricks.ipynb#Y106sZmlsZQ%3D%3D?line=0'>1</a> # Read␣
 ↪clipboard in python
----> <a href='vscode-notebook-cell:/c%3A/Users/MUHAMMAD%20ADNAN/Desktop/
 ↪Python_02/Day-3/tipsandtricks.ipynb#Y106sZmlsZQ%3D%3D?line=1'>2</a> df = pd.
 ↪read_clipboard()
      <a href='vscode-notebook-cell:/c%3A/Users/MUHAMMAD%20ADNAN/Desktop/
 ↪Python_02/Day-3/tipsandtricks.ipynb#Y106sZmlsZQ%3D%3D?line=2'>3</a> df
      <a href='vscode-notebook-cell:/c%3A/Users/MUHAMMAD%20ADNAN/Desktop/
 ↪Python_02/Day-3/tipsandtricks.ipynb#Y106sZmlsZQ%3D%3D?line=3'>4</a> # To save␣
 ↪this clipboard data into a csv file

File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\cli boards.
 ↪py:85, in read_clipboard(sep, **kwargs)
     80 elif len(sep) > 1 and kwargs.get("engine") == "c":
     81     warnings.warn(
     82         "read_clipboard with regex separator does not work properly wit ␣
 ↪c engine."
     83     )
---> 85 return read_csv(StringIO(text), sep=sep, **kwargs)

File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\util\_ ecorators.
 ↪py:311, in deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.
 ↪wrapper(*args, **kwargs)
    305 if len(args) > num_allow_args:
    306     warnings.warn(
    307         msg.format(arguments=arguments),
    308         FutureWarning,
    309         stacklevel=stacklevel,
    310     )
--> 311 return func(*args, **kwargs)
```

```
File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\parsers\readers.
 ↪py:678, in read_csv(filepath_or_buffer, sep, delimiter, header, names,␣
 ↪index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine,␣
 ↪converters, true_values, false_values, skipinitialspace, skiprows, skipfooter ␣
 ↪nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines,␣
 ↪parse_dates, infer_datetime_format, keep_date_col, date_parser, dayfirst,␣
 ↪cache_dates, iterator, chunksize, compression, thousands, decimal,␣
 ↪lineterminator, quotechar, quoting, doublequote, escapechar, comment,␣
 ↪encoding, encoding_errors, dialect, error_bad_lines, warn_bad_lines,␣
 ↪on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision,␣
 ↪storage_options)
    663 kwds_defaults = _refine_defaults_read(
    664     dialect,
    665     delimiter,
   (…)
    674     defaults={"delimiter": ","},
    675 )
    676 kwds.update(kwds_defaults)
--> 678 return _read(filepath_or_buffer, kwds)

File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\parsers\readers.
 ↪py:575, in _read(filepath_or_buffer, kwds)
    572 _validate_names(kwds.get("names", None))
    574 # Create the parser.
--> 575 parser = TextFileReader(filepath_or_buffer, **kwds)
    577 if chunksize or iterator:
    578     return parser

File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\parsers\readers.
 ↪py:932, in TextFileReader.__init__(self, f, engine, **kwds)
    929     self.options["has_index_names"] = kwds["has_index_names"]
    931 self.handles: IOHandles | None = None
--> 932 self._engine = self._make_engine(f, self.engine)

File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\parsers\readers.
 ↪py:1234, in TextFileReader._make_engine(self, f, engine)
   1231     raise ValueError(msg)
   1233 try:
-> 1234     return mapping[engine](f, **self.options)
   1235 except Exception:
   1236     if self.handles is not None:

File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\parsers\python_pa
 ↪py:115, in PythonParser.__init__(self, f, **kwds)
    109 self._col_indices: list[int] | None = None
    110 columns: list[list[Scalar | None]]
    111 (
    112     columns,
    113     self.num_original_columns,
    114     self.unnamed_cols,
--> 115 ) = self._infer_columns()
    117 # Now self.columns has the set of columns that we will process.
    118 # The original set is stored in self.original_columns.
    119 # error: Cannot determine type of 'index_names'
    120 self.columns: list[Hashable]

File c:\Users\MUHAMMAD␣
 ↪ADNAN\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\io\parsers\python_pa
 ↪py:386, in PythonParser._infer_columns(self)
    383         return columns, num_original_columns, unnamed_cols
    385     if not self.names:
--> 386         raise EmptyDataError("No columns to parse from file") from err
    388     line = self.names[:]
    390 this_columns: list[Scalar | None] = []

EmptyDataError: No columns to parse from file
```

12 12 - Split datasets into two subsets

[ ]:
```python
# Dataframe download
import seaborn as sns
import pandas as pd

df = sns.load_dataset('titanic')
df.head()
```

[ ]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

[ ]:
```python
len(df)
```

[ ]:
```
891
```

```
[ ]: df.shape
```

```
[ ]: (891, 15)
```

```
[ ]: from random import random
     kashti_1 = df.sample(frac=0.50, random_state=1)
     kashti_1.shape
```

```
[ ]: (446, 15)
```

```
[ ]: kashti_2 = df.drop(kashti_1.index)
     kashti_2.shape
```

```
[ ]: (445, 15)
```

```
[ ]: kashti_1.head()
```

```
[ ]:      survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
     862         1       1  female  48.0      0      0  25.9292        S   First
     223         0       3    male   NaN      0      0   7.8958        S   Third
     84          1       2  female  17.0      0      0  10.5000        S  Second
     680         0       3  female   NaN      0      0   8.1375        Q   Third
     535         1       2  female   7.0      0      2  26.2500        S  Second

            who  adult_male deck  embark_town alive  alone
     862  woman       False    D  Southampton   yes   True
     223    man        True  NaN  Southampton    no   True
     84   woman       False  NaN  Southampton   yes   True
     680  woman       False  NaN   Queenstown    no   True
     535  child       False  NaN  Southampton   yes  False
```

```
[ ]: kashti_2.head()
```

```
[ ]:     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
     1          1       1  female  38.0      1      0  71.2833        C   First
     7          0       3    male   2.0      3      1  21.0750        S   Third
     10         1       3  female   4.0      1      1  16.7000        S   Third
     15         1       2  female  55.0      0      0  16.0000        S  Second
     18         0       3  female  31.0      1      0  18.0000        S   Third

           who  adult_male deck  embark_town alive  alone
     1   woman       False    C    Cherbourg   yes  False
     7   child       False  NaN  Southampton    no  False
     10  child       False    G  Southampton   yes  False
     15  woman       False  NaN  Southampton   yes   True
     18  woman       False  NaN  Southampton    no  False
```

```
[ ]: len(kashti_1) + len(kashti_2)
[ ]: 891
```

13 13 - Join two split datasets together

```
[ ]: # Appened again to make the original dataset
     df1 = kashti_1.append(kashti_2)
     df1.shape
```

```
C:\Users\MUHAMMAD ADNAN\AppData\Local\Temp\ipykernel_17616\1670357066.py:2:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  df1 = kashti_1.append(kashti_2)
```

```
[ ]: (891, 15)
```

14 14 - Filtering a dataset

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

```
[ ]:    survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
     0         0       3    male  22.0      1      0   7.2500        S  Third
     1         1       1  female  38.0      1      0  71.2833        C  First
     2         1       3  female  26.0      0      0   7.9250        S  Third
     3         1       1  female  35.0      1      0  53.1000        S  First
     4         0       3    male  35.0      0      0   8.0500        S  Third

          who  adult_male deck  embark_town alive  alone
     0    man        True  NaN  Southampton    no  False
     1  woman       False    C    Cherbourg   yes  False
     2  woman       False  NaN  Southampton   yes   True
     3  woman       False    C  Southampton   yes  False
     4    man        True  NaN  Southampton    no   True
```

```
[ ]: # Find unique values in a column
     df['sex'].unique()
```

```
[ ]: array(['male', 'female'], dtype=object)
```

```
[ ]: df['age'].unique()
```

```
[ ]: array([22.  , 38.  , 26.  , 35.  ,   nan, 54.  ,  2.  , 27.  , 14.  ,
             4.  , 58.  , 20.  , 39.  , 55.  , 31.  , 34.  , 15.  , 28.  ,
             8.  , 19.  , 40.  , 66.  , 42.  , 21.  , 18.  ,  3.  ,  7.  ,
            49.  , 29.  , 65.  , 28.5 ,  5.  , 11.  , 45.  , 17.  , 32.  ,
            16.  , 25.  ,  0.83, 30.  , 33.  , 23.  , 24.  , 46.  , 59.  ,
            71.  , 37.  , 47.  , 14.5 , 70.5 , 32.5 , 12.  ,  9.  , 36.5 ,
            51.  , 55.5 , 40.5 , 44.  ,  1.  , 61.  , 56.  , 50.  , 36.  ,
            45.5 , 20.5 , 62.  , 41.  , 52.  , 63.  , 23.5 ,  0.92, 43.  ,
            60.  , 10.  , 64.  , 13.  , 48.  ,  0.75, 53.  , 57.  , 80.  ,
            70.  , 24.5 ,  6.  ,  0.67, 30.5 ,  0.42, 34.5 , 74.  ])
```

[ ]:
```
# Take only female or male dataset
df[(df.sex=='female')].head()
```

[ ]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
1         1       1  female  38.0      1      0  71.2833        C   First
2         1       3  female  26.0      0      0   7.9250        S   Third
3         1       1  female  35.0      1      0  53.1000        S   First
8         1       3  female  27.0      0      2  11.1333        S   Third
9         1       2  female  14.0      1      0  30.0708        C  Second

     who  adult_male deck  embark_town alive  alone
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
8  woman       False  NaN  Southampton   yes  False
9  child       False  NaN    Cherbourg   yes  False
```

[ ]:
```
df[(df.embark_town=='Cherbourg')].shape
```

[ ]:
```
(168, 15)
```

[ ]:
```
# Just take data from the specific gender and embarked town
df[(df.embark_town=='Southampton') &
   (df.sex=='female')].shape
```

[ ]:
```
(203, 15)
```

[ ]:
```
df[((df.embark_town=='Southampton') |
    (df.embark_town=='Queenstown')) &
    (df.sex=='male')].head()
```

[ ]:
```
   survived  pclass   sex   age  sibsp  parch     fare embarked  class    who  \
0         0       3  male  22.0      1      0   7.2500        S  Third    man
4         0       3  male  35.0      0      0   8.0500        S  Third    man
5         0       3  male   NaN      0      0   8.4583        Q  Third    man
6         0       1  male  54.0      0      0  51.8625        S  First    man
7         0       3  male   2.0      3      1  21.0750        S  Third  child

   adult_male deck  embark_town alive  alone
0        True  NaN  Southampton    no  False
4        True  NaN  Southampton    no   True
5        True  NaN   Queenstown    no   True
6        True    E  Southampton    no   True
7       False  NaN  Southampton    no  False
```

```
[ ]: # Take the data only one town out of three.
     df[df.embark_town.isin(['Queenstown'])].head()
```

```
[ ]:     survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
     5          0       3    male   NaN      0      0   8.4583        Q  Third
     16         0       3    male   2.0      4      1  29.1250        Q  Third
     22         1       3  female  15.0      0      0   8.0292        Q  Third
     28         1       3  female   NaN      0      0   7.8792        Q  Third
     32         1       3  female   NaN      0      0   7.7500        Q  Third

           who  adult_male deck embark_town alive  alone
     5     man        True  NaN  Queenstown    no   True
     16  child       False  NaN  Queenstown    no  False
     22  child       False  NaN  Queenstown   yes   True
     28  woman       False  NaN  Queenstown   yes   True
     32  woman       False  NaN  Queenstown   yes   True
```

```
[ ]: # Take the age greater than 20 years
     df[df.age > 20].shape
```

```
[ ]: (535, 15)
```

15 15 - Filtering by large categories

```
[ ]: df.embark_town.value_counts()
```

```
[ ]: Southampton    644
     Cherbourg      168
     Queenstown      77
     Name: embark_town, dtype: int64
```

```
[ ]: df.sex.value_counts()
```

```
[ ]: male      577
     female    314
     Name: sex, dtype: int64
```

```
[ ]: # Largest value in a column (MOST FREQUENT) (Method 1)
     df.age.value_counts().nlargest(5)
```

```
[ ]: 24.0    30
     22.0    27
     18.0    26
     19.0    25
     28.0    25
     Name: age, dtype: int64
```

```
[ ]: df.age.value_counts().nlargest(5).index
```

```
[ ]: Float64Index([24.0, 22.0, 18.0, 19.0, 28.0], dtype='float64')
```

```
[ ]: # Largest value in a column (MOST FREQUENT) (Method 2)
     counts = df.age.value_counts()
     counts.nlargest(3)
```

```
[ ]: 24.0    30
     22.0    27
     18.0    26
     Name: age, dtype: int64
```

```
[ ]: # Largest value in a column (MOST FREQUENT) (Method 2)
     counts = df.embark_town.value_counts()
     counts.nlargest(3).index
```

```
[ ]: Index(['Southampton', 'Cherbourg', 'Queenstown'], dtype='object')
```

```
[ ]: counts = df.who.value_counts()
     counts.nlargest(3)
```

```
[ ]: man      537
     woman    271
     child     83
     Name: who, dtype: int64
```

```
[ ]: df[df.who.isin(counts.nlargest(1).index)].head()
```

```
[ ]:     survived  pclass   sex   age  sibsp  parch     fare embarked  class  who  \
     0          0       3  male  22.0      1      0   7.2500        S  Third  man
     4          0       3  male  35.0      0      0   8.0500        S  Third  man
     5          0       3  male   NaN      0      0   8.4583        Q  Third  man
     6          0       1  male  54.0      0      0  51.8625        S  First  man
     12         0       3  male  20.0      0      0   8.0500        S  Third  man

         adult_male deck  embark_town alive  alone
     0         True  NaN  Southampton    no  False
     4         True  NaN  Southampton    no   True
     5         True  NaN   Queenstown    no   True
     6         True    E  Southampton    no   True
     12        True  NaN  Southampton    no   True
```

16 16 - Spliting a string into multiple columns

[ ]:
```
# Import libraries
import pandas as pd

df = pd.DataFrame({'Name':['Muhammad Adnan', 'Usman ali', 'shafi Ullah',␣
 ↪'Muzammil Hussain', 'Mubashir Hussain'],
                   'Location': ['Dera, Pakistan', 'Islamabad, Pakistan', 'Dera,␣
 ↪Pakistan', 'Dera, Pakistan', 'Dera, Pakistan']})
df
```

[ ]:
```
               Name             Location
0    Muhammad Adnan       Dera, Pakistan
1         Usman ali  Islamabad, Pakistan
2       shafi Ullah       Dera, Pakistan
3  Muzammil Hussain       Dera, Pakistan
4  Mubashir Hussain       Dera, Pakistan
```

[ ]:
```
df.Name.str.split(' ').head()
```

[ ]:
```
0      [Muhammad, Adnan]
1           [Usman, ali]
2         [shafi, Ullah]
3    [Muzammil, Hussain]
4    [Mubashir, Hussain]
Name: Name, dtype: object
```

[ ]:
```
# Split the column into two columns and then add into dataset and update the␣
 ↪dataset
df[['First_name', 'Second_name']] = df.Name.str.split(' ', expand=True)
df[['City', 'Country']] = df.Location.str.split(',', expand=True)
df
```

[ ]:
```
               Name             Location First_name Second_name       City  \
0    Muhammad Adnan       Dera, Pakistan   Muhammad       Adnan       Dera
1         Usman ali  Islamabad, Pakistan      Usman         ali  Islamabad
2       shafi Ullah       Dera, Pakistan      shafi       Ullah       Dera
3  Muzammil Hussain       Dera, Pakistan   Muzammil     Hussain       Dera
4  Mubashir Hussain       Dera, Pakistan   Mubashir     Hussain       Dera

     Country
0   Pakistan
1   Pakistan
2   Pakistan
3   Pakistan
4   Pakistan
```

```
[ ]: # Refine data manipulation
     df = df[['First_name', 'Second_name', 'City', 'Country']]
     df
```

```
[ ]:   First_name Second_name       City    Country
     0   Muhammad       Adnan       Dera   Pakistan
     1      Usman         ali  Islamabad   Pakistan
     2      shafi       Ullah       Dera   Pakistan
     3   Muzammil     Hussain       Dera   Pakistan
     4   Mubashir     Hussain       Dera   Pakistan
```

17 17 - Aggregate by multiple groups/functions

```
[ ]: # Libraries
     import pandas as pd
     import seaborn as sns

     # Import dataset
     kashti = sns.load_dataset('titanic')
     kashti.head()
```

```
[ ]:    survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
     0         0       3    male  22.0      1      0   7.2500        S  Third
     1         1       1  female  38.0      1      0  71.2833        C  First
     2         1       3  female  26.0      0      0   7.9250        S  Third
     3         1       1  female  35.0      1      0  53.1000        S  First
     4         0       3    male  35.0      0      0   8.0500        S  Third

          who  adult_male deck  embark_town alive  alone
     0    man        True  NaN  Southampton    no  False
     1  woman       False    C    Cherbourg   yes  False
     2  woman       False  NaN  Southampton   yes   True
     3  woman       False    C  Southampton   yes  False
     4    man        True  NaN  Southampton    no   True
```

```
[ ]: # Group by the column
     kashti.groupby('who').count()
```

```
[ ]:        survived  pclass  sex  age  sibsp  parch  fare  embarked  class  \
     who
     child        83      83   83   83     83     83    83        83     83
     man         537     537  537  413    537    537   537       537    537
     woman       271     271  271  218    271    271   271       269    271

            adult_male  deck  embark_town  alive  alone
     who
     child          83    13           83     83     83
```

```
man           537    99           537    537    537
woman         271    91           269    271    271
```

[ ]:
```
# sum of the column
kashti.groupby('who').sum()
```

[ ]:
```
       survived  pclass       age  sibsp  parch        fare  adult_male  alone
who
child        49     218    528.67    144    105   2721.2210           0      6
man          88    1274  13700.50    159     82  13352.0656         537    410
woman       205     565   6976.00    163    153  12620.6627           0    121
```

[ ]:
```
# To check the value counts in each column
len(kashti.groupby('class'))
```

[ ]:
```
3
```

[ ]:
```
# Counts of many columns at once
kashti.groupby(['sex', 'pclass', 'embarked']).count()
```

[ ]:
```
                        survived  age  sibsp  parch  fare  class  who  \
sex    pclass embarked
female 1      C               43   38     43     43    43     43   43
              Q                1    1      1      1     1      1    1
              S               48   44     48     48    48     48   48
       2      C                7    7      7      7     7      7    7
              Q                2    1      2      2     2      2    2
              S               67   66     67     67    67     67   67
       3      C               23   16     23     23    23     23   23
              Q               33   10     33     33    33     33   33
              S               88   76     88     88    88     88   88
male   1      C               42   36     42     42    42     42   42
              Q                1    1      1      1     1      1    1
              S               79   64     79     79    79     79   79
       2      C               10    8     10     10    10     10   10
              Q                1    1      1      1     1      1    1
              S               97   90     97     97    97     97   97
       3      C               43   25     43     43    43     43   43
              Q               39   14     39     39    39     39   39
              S              265  214    265    265   265    265  265

                        adult_male  deck  embark_town  alive  alone
sex    pclass embarked
female 1      C                 43    35           43     43     43
              Q                  1     1            1      1      1
              S                 48    43           48     48     48
       2      C                  7     1            7      7      7
```

```
            Q                  2      1          2      2      2
            S                 67      8         67     67     67
       3    C                 23      1         23     23     23
            Q                 33      0         33     33     33
            S                 88      5         88     88     88
male   1    C                 42     31         42     42     42
            Q                  1      1          1      1      1
            S                 79     62         79     79     79
       2    C                 10      1         10     10     10
            Q                  1      0          1      1      1
            S                 97      5         97     97     97
       3    C                 43      0         43     43     43
            Q                 39      1         39     39     39
            S                265      5        265    265    265
```

18 18 - Select specific rows and columns

[ ]:
```
# Select the column
kashti[['sex', 'class']]
```

[ ]:
```
        sex   class
0      male   Third
1    female   First
2    female   Third
3    female   First
4      male   Third
..      …       …
886    male  Second
887  female   First
888  female   Third
889    male   First
890    male   Third

[891 rows x 2 columns]
```

[ ]:
```
kashti.describe()
```

[ ]:
```
         survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

```
[ ]: # kashti.describe().loc[['min', '25%', '50%', '75%', 'max']] # (Method 1)
     kashti.describe().loc['min':'max'] # (Method 2)
```

```
[ ]:      survived  pclass     age  sibsp  parch      fare
     min       0.0     1.0   0.420    0.0    0.0    0.0000
     25%       0.0     2.0  20.125    0.0    0.0    7.9104
     50%       0.0     3.0  28.000    0.0    0.0   14.4542
     75%       1.0     3.0  38.000    1.0    0.0   31.0000
     max       1.0     3.0  80.000    8.0    6.0  512.3292
```

```
[ ]: kashti.describe().loc['min':'max', 'age': 'fare']
```

```
[ ]:         age  sibsp  parch      fare
     min   0.420    0.0    0.0    0.0000
     25%  20.125    0.0    0.0    7.9104
     50%  28.000    0.0    0.0   14.4542
     75%  38.000    1.0    0.0   31.0000
     max  80.000    8.0    6.0  512.3292
```

19 19 - Reshape multiindex series

```
[ ]: # Calculate the mean of the column
     kashti.age.mean()
```

```
[ ]: 29.69911764705882
```

```
[ ]: # Calculate the mean of both male and female based on survived variable
     kashti.groupby('sex').survived.mean()
```

```
[ ]: sex
     female    0.742038
     male      0.188908
     Name: survived, dtype: float64
```

```
[ ]: # Calculate the mean of (sex, class & embark_town) based on survived variable
     kashti.groupby(['sex', 'class', 'embark_town']).survived.mean()
```

```
[ ]: sex     class   embark_town
     female  First   Cherbourg      0.976744
                     Queenstown     1.000000
                     Southampton    0.958333
             Second  Cherbourg      1.000000
                     Queenstown     1.000000
                     Southampton    0.910448
             Third   Cherbourg      0.652174
                     Queenstown     0.727273
                     Southampton    0.375000
```

```
male     First    Cherbourg      0.404762
                  Queenstown     0.000000
                  Southampton    0.354430
         Second   Cherbourg      0.200000
                  Queenstown     0.000000
                  Southampton    0.154639
         Third    Cherbourg      0.232558
                  Queenstown     0.076923
                  Southampton    0.128302
Name: survived, dtype: float64
```

20 20 - Continous to categorical data conversion

[ ]:
```
kashti.head()
```

[ ]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

[ ]:
```
kashti.age.head()
```

[ ]:
```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
Name: age, dtype: float64
```

[ ]:
```
# Creating bins
pd.cut(kashti.age, bins = [0, 18, 25, 99], labels=['Child', 'Young', 'Adult']).
 ↪head()
kashti['new_age'] = pd.cut(kashti.age, bins = [0, 18, 25, 99], labels=['Child',␣
 ↪'Young', 'Adult'])
kashti.head()
```

```
[ ]:    survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
     0         0       3    male  22.0      1      0   7.2500        S  Third
     1         1       1  female  38.0      1      0  71.2833        C  First
     2         1       3  female  26.0      0      0   7.9250        S  Third
     3         1       1  female  35.0      1      0  53.1000        S  First
     4         0       3    male  35.0      0      0   8.0500        S  Third

          who  adult_male deck  embark_town alive  alone new_age
     0    man        True  NaN  Southampton    no  False   Young
     1  woman       False    C    Cherbourg   yes  False   Adult
     2  woman       False  NaN  Southampton   yes   True   Adult
     3  woman       False    C  Southampton   yes  False   Adult
     4    man        True  NaN  Southampton    no   True   Adult
```

```
[ ]: kashti['new_age'].value_counts()
```

```
[ ]: Adult    413
     Young    162
     Child    139
     Name: new_age, dtype: int64
```

21   21 - Convert one set of values into an other one

```
[ ]: kashti.sex.head()
```

```
[ ]: 0      male
     1    female
     2    female
     3    female
     4      male
     Name: sex, dtype: object
```

```
[ ]: # Convert the categorical variable into numerical variable
     kashti['Sex_num'] = kashti.sex.map({'male':0, 'female':1})
     kashti.head()
```

```
[ ]:    survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
     0         0       3    male  22.0      1      0   7.2500        S  Third
     1         1       1  female  38.0      1      0  71.2833        C  First
     2         1       3  female  26.0      0      0   7.9250        S  Third
     3         1       1  female  35.0      1      0  53.1000        S  First
     4         0       3    male  35.0      0      0   8.0500        S  Third

          who  adult_male deck  embark_town alive  alone new_age Sex_num
     0    man        True  NaN  Southampton    no  False   Young       0
     1  woman       False    C    Cherbourg   yes  False   Adult       1
     2  woman       False  NaN  Southampton   yes   True   Adult       1
```

```
3   woman        False    C  Southampton   yes  False   Adult        1
4     man         True  NaN  Southampton    no   True   Adult        0
```

[ ]:
```
# convert the categorical variable into numerical variable more than three␣
 ↪subcategories
kashti['embarked_num'] = kashti.embarked.factorize()[0]
kashti.head(15)
```

[ ]:
```
    survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0          0       3    male  22.0      1      0   7.2500        S   Third
1          1       1  female  38.0      1      0  71.2833        C   First
2          1       3  female  26.0      0      0   7.9250        S   Third
3          1       1  female  35.0      1      0  53.1000        S   First
4          0       3    male  35.0      0      0   8.0500        S   Third
5          0       3    male   NaN      0      0   8.4583        Q   Third
6          0       1    male  54.0      0      0  51.8625        S   First
7          0       3    male   2.0      3      1  21.0750        S   Third
8          1       3  female  27.0      0      2  11.1333        S   Third
9          1       2  female  14.0      1      0  30.0708        C  Second
10         1       3  female   4.0      1      1  16.7000        S   Third
11         1       1  female  58.0      0      0  26.5500        S   First
12         0       3    male  20.0      0      0   8.0500        S   Third
13         0       3    male  39.0      1      5  31.2750        S   Third
14         0       3  female  14.0      0      0   7.8542        S   Third

      who  adult_male deck  embark_town alive  alone new_age Sex_num  \
0     man        True  NaN  Southampton    no  False   Young       0
1   woman       False    C    Cherbourg   yes  False   Adult       1
2   woman       False  NaN  Southampton   yes   True   Adult       1
3   woman       False    C  Southampton   yes  False   Adult       1
4     man        True  NaN  Southampton    no   True   Adult       0
5     man        True  NaN   Queenstown    no   True     NaN       0
6     man        True    E  Southampton    no   True   Adult       0
7   child       False  NaN  Southampton    no  False   Child       0
8   woman       False  NaN  Southampton   yes  False   Adult       1
9   child       False  NaN    Cherbourg   yes  False   Child       1
10  child       False    G  Southampton   yes  False   Child       1
11  woman       False    C  Southampton   yes   True   Adult       1
12    man        True  NaN  Southampton    no   True   Young       0
13    man        True  NaN  Southampton    no  False   Adult       0
14  child       False  NaN  Southampton    no   True   Child       1

    embarked_num
0              0
1              1
2              0
3              0
```

```
4             0
5             2
6             0
7             0
8             0
9             1
10            0
11            0
12            0
13            0
14            0
```

22 22 - Transpose a wide dataframe

[ ]:
```python
import numpy as np
import pandas as pd

# Creating a new dataset
dd = pd.DataFrame(np.random.rand(200,25),␣
 ↪columns=list('ABCDEFGHIJKLMNOPQRSTUVWXY'))
dd.head()
```

[ ]:
```
          A         B         C         D         E         F         G  \
0  0.143670  0.426703  0.252598  0.844743  0.910284  0.713044  0.359033
1  0.281825  0.053178  0.000188  0.925794  0.967202  0.204736  0.946789
2  0.046354  0.861823  0.876958  0.189724  0.282258  0.541298  0.529835
3  0.156297  0.629973  0.695055  0.622788  0.801721  0.881718  0.053921
4  0.478483  0.386189  0.260960  0.665449  0.932963  0.339064  0.110927

          H         I         J  …         P         Q         R         S  \
0  0.698907  0.627262  0.470933  …  0.876990  0.088320  0.076386  0.022555
1  0.480048  0.928505  0.672022  …  0.775715  0.943715  0.519437  0.189680
2  0.943904  0.660787  0.914902  …  0.260070  0.247387  0.592540  0.884364
3  0.051032  0.864200  0.368168  …  0.500977  0.355818  0.245379  0.926770
4  0.020734  0.747317  0.895862  …  0.222065  0.076016  0.075220  0.250870

          T         U         V         W         X         Y
0  0.392101  0.809865  0.708977  0.686948  0.649985  0.389469
1  0.772814  0.593720  0.194726  0.350917  0.526939  0.286099
2  0.304713  0.135501  0.469152  0.906661  0.702320  0.401458
3  0.623016  0.142638  0.832409  0.865212  0.793916  0.872834
4  0.698427  0.381307  0.482080  0.961771  0.359486  0.088955

[5 rows x 25 columns]
```

```
[ ]: # Transpose the dataset (Columns to rows & rows to columns)
     dd.head(10).T
```

```
[ ]:          0         1         2         3         4         5         6  \
     A  0.143670  0.281825  0.046354  0.156297  0.478483  0.176885  0.377511
     B  0.426703  0.053178  0.861823  0.629973  0.386189  0.498281  0.515508
     C  0.252598  0.000188  0.876958  0.695055  0.260960  0.049361  0.172716
     D  0.844743  0.925794  0.189724  0.622788  0.665449  0.999688  0.097357
     E  0.910284  0.967202  0.282258  0.801721  0.932963  0.731052  0.434867
     F  0.713044  0.204736  0.541298  0.881718  0.339064  0.001201  0.661017
     G  0.359033  0.946789  0.529835  0.053921  0.110927  0.418822  0.789808
     H  0.698907  0.480048  0.943904  0.051032  0.020734  0.876065  0.161053
     I  0.627262  0.928505  0.660787  0.864200  0.747317  0.998535  0.585099
     J  0.470933  0.672022  0.914902  0.368168  0.895862  0.272380  0.786924
     K  0.352799  0.383705  0.244226  0.094901  0.626042  0.158970  0.731366
     L  0.434371  0.852754  0.423320  0.945835  0.582836  0.075567  0.411987
     M  0.373314  0.586754  0.572455  0.161335  0.395064  0.282962  0.357300
     N  0.272833  0.496665  0.278694  0.242589  0.083799  0.753122  0.254408
     O  0.113905  0.370749  0.092933  0.976538  0.013531  0.286264  0.537829
     P  0.876990  0.775715  0.260070  0.500977  0.222065  0.300653  0.805833
     Q  0.088320  0.943715  0.247387  0.355818  0.076016  0.736384  0.321514
     R  0.076386  0.519437  0.592540  0.245379  0.075220  0.587289  0.748326
     S  0.022555  0.189680  0.884364  0.926770  0.250870  0.561506  0.074712
     T  0.392101  0.772814  0.304713  0.623016  0.698427  0.342702  0.945281
     U  0.809865  0.593720  0.135501  0.142638  0.381307  0.323299  0.033142
     V  0.708977  0.194726  0.469152  0.832409  0.482080  0.877824  0.592849
     W  0.686948  0.350917  0.906661  0.865212  0.961771  0.635219  0.033575
     X  0.649985  0.526939  0.702320  0.793916  0.359486  0.070561  0.392812
     Y  0.389469  0.286099  0.401458  0.872834  0.088955  0.371785  0.115149

                7         8         9
     A  0.480737  0.506740  0.982098
     B  0.873085  0.913914  0.317381
     C  0.667192  0.597266  0.561242
     D  0.427538  0.227840  0.482498
     E  0.842123  0.054008  0.256057
     F  0.093569  0.162614  0.067567
     G  0.914693  0.967376  0.393270
     H  0.741533  0.421940  0.525966
     I  0.318017  0.870788  0.173555
     J  0.088439  0.605775  0.071832
     K  0.434930  0.881009  0.315967
     L  0.515695  0.316154  0.914994
     M  0.502835  0.673946  0.370740
     N  0.884435  0.218056  0.843675
     O  0.220787  0.748308  0.522375
     P  0.887006  0.708295  0.989552
```

```
Q  0.188827  0.199910  0.434978
R  0.695607  0.880591  0.745124
S  0.928009  0.506011  0.530864
T  0.308779  0.806448  0.454734
U  0.726959  0.079253  0.419944
V  0.142106  0.726824  0.985653
W  0.178857  0.323820  0.730973
X  0.485476  0.139078  0.576835
Y  0.780702  0.945455  0.177119
```

[ ]:
```
dd.describe().T
```

[ ]:
```
   count      mean       std       min       25%       50%       75%       max
A  200.0  0.504029  0.295181  0.000798  0.231674  0.481889  0.786726  0.999998
B  200.0  0.484367  0.277784  0.007332  0.289783  0.460883  0.695330  0.984384
C  200.0  0.493837  0.285195  0.000188  0.266152  0.505142  0.702597  0.997154
D  200.0  0.506595  0.284447  0.010404  0.254646  0.484582  0.764958  0.999688
E  200.0  0.520093  0.295809  0.002137  0.258813  0.532283  0.793696  0.999050
F  200.0  0.523081  0.292085  0.001201  0.276216  0.540681  0.775124  0.999528
G  200.0  0.519297  0.290883  0.011019  0.276145  0.496633  0.805369  0.997842
H  200.0  0.504005  0.277771  0.000810  0.270427  0.500873  0.738310  0.986890
I  200.0  0.496821  0.294264  0.001843  0.218705  0.496676  0.744006  0.998535
J  200.0  0.501684  0.278720  0.001030  0.274798  0.485647  0.730206  0.997851
K  200.0  0.502273  0.288305  0.000365  0.253777  0.490507  0.739226  0.993128
L  200.0  0.528660  0.287530  0.011934  0.269838  0.543491  0.777358  0.997377
M  200.0  0.462612  0.291823  0.004356  0.216150  0.410520  0.711579  0.998392
N  200.0  0.477975  0.278687  0.000521  0.242245  0.459424  0.718697  0.992644
O  200.0  0.508073  0.299476  0.003884  0.260091  0.523659  0.750709  0.997998
P  200.0  0.516407  0.300680  0.004157  0.252597  0.531388  0.777569  0.996777
Q  200.0  0.504151  0.288905  0.003003  0.264820  0.530792  0.757080  0.999323
R  200.0  0.513217  0.297398  0.006809  0.258230  0.506103  0.775424  0.999918
S  200.0  0.513290  0.291363  0.000879  0.274048  0.505563  0.787715  0.993440
T  200.0  0.538002  0.290591  0.001009  0.306387  0.537605  0.807399  0.993590
U  200.0  0.457520  0.261401  0.001891  0.238063  0.464105  0.654725  0.998675
V  200.0  0.513380  0.279596  0.003662  0.274882  0.523026  0.760989  0.998565
W  200.0  0.471797  0.300541  0.000624  0.199012  0.432584  0.770100  0.995471
X  200.0  0.511791  0.291545  0.009855  0.269058  0.488275  0.788821  0.995954
Y  200.0  0.447684  0.293832  0.001193  0.191799  0.422730  0.668345  0.999575
```

23 23 - Reshaping a dataframe

[ ]:
```
fasla = pd.DataFrame([['12345', 100, 200, 300], ['34567', 400, 500, 600],␣
 ↪['56789', 700, 800, 900]],
                     columns=['zip', 'factory', 'warehouse', 'retail'])
fasla.head()
```

```
[ ]:      zip  factory  warehouse  retail
     0  12345      100        200     300
     1  34567      400        500     600
     2  56789      700        800     900
```

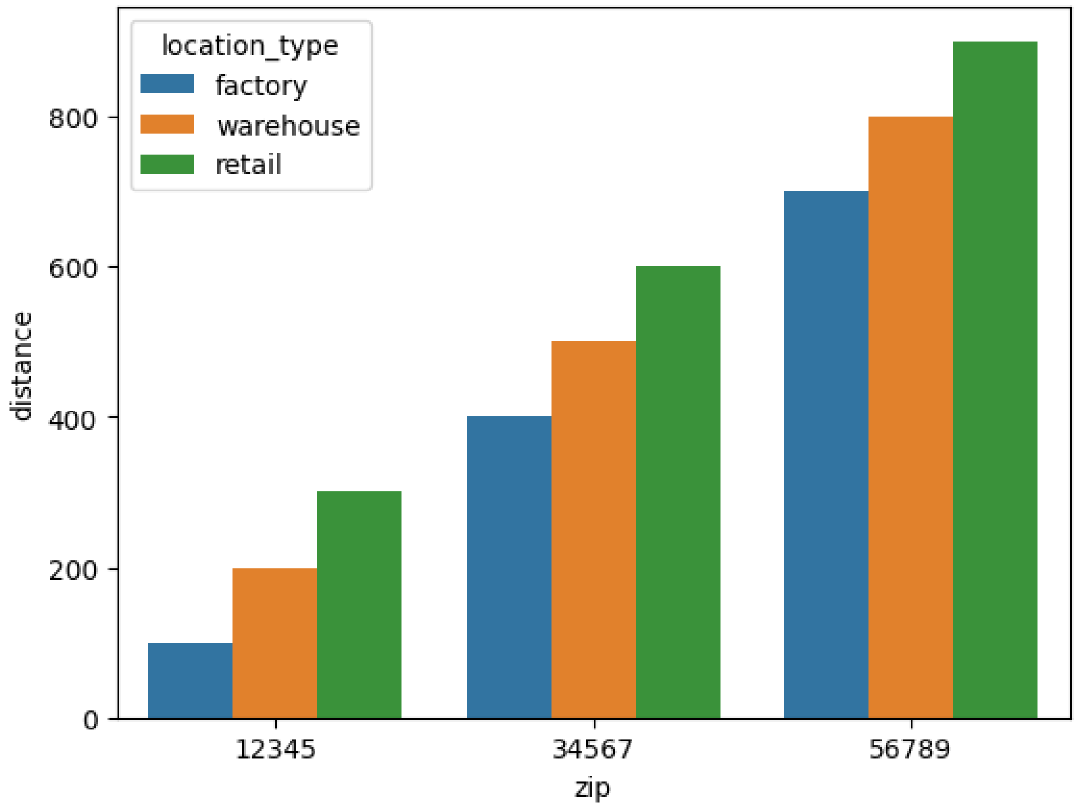
```
[ ]: fasla2 = pd.DataFrame([[1, '12345', 'factory'], [2, '34567', 'warehouse']],
                           columns=['user_id', 'zip', 'location_type'])
     fasla2.head()
```

```
[ ]:    user_id    zip location_type
     0        1  12345       factory
     1        2  34567     warehouse
```

```
[ ]: fasla2 = fasla.melt(id_vars='zip', var_name='location_type',␣
      ↪value_name='distance')
```

```
[ ]: import seaborn as sns
     sns.barplot(x='zip', y='distance', hue='location_type', data=fasla.
      ↪melt(id_vars='zip', var_name='location_type', value_name='distance'))
```

```
[ ]: <AxesSubplot:xlabel='zip', ylabel='distance'>
```



[ ]:
```
import plotly.express as px
fig = px.bar(fasla2, x='zip', y='distance', color='location_type')
fig.update_traces(width=0.5)
fig.update_layout(width=600, height=400)
fig.show()
```

[ ]:
```

```