In [1]:
```
!pip install pandas
```

```
Defaulting to user installation because normal site-packages is not writea
ble
Collecting pandas
  Downloading pandas-2.2.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014
_x86_64.whl.metadata (89 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 89.9/89.9 kB 1.3 MB/s eta
0:00:00a 0:00:01
Requirement already satisfied: numpy>=1.22.4 in ./.local/lib/python3.10/si
te-packages (from pandas) (2.1.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/tljh/user/li
b/python3.10/site-packages (from pandas) (2.9.0.post0)
Collecting pytz>=2020.1 (from pandas)
  Downloading pytz-2024.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2024.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in /opt/tljh/user/lib/python3.10/s
ite-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Downloading pandas-2.2.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x
86_64.whl (13.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 5.9 MB/s eta 0:0
0:0000:0100:01
Downloading pytz-2024.2-py2.py3-none-any.whl (508 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 508.0/508.0 kB 4.3 MB/s eta
0:00:0000:0100:01
Downloading tzdata-2024.2-py2.py3-none-any.whl (346 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 346.6/346.6 kB 4.1 MB/s eta
0:00:00a 0:00:01
Installing collected packages: pytz, tzdata, pandas
Successfully installed pandas-2.2.3 pytz-2024.2 tzdata-2024.2

[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: pip install --upgrade pip
```

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

In [4]:
```
ss=pd.Series([1,2,3,4,5])
print(type(ss))
print(ss)
```

```
<class 'pandas.core.series.Series'>
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

Series with customized index

In [6]:
```
ss=pd.Series([1,2,3,4,5],index=[11,12,13,14,15])
print(ss)
```

```
11    1
12    2
13    3
14    4
15    5
dtype: int64
```

Basic Mathematical opertions on series

In [11]:
```
ss[1:3]
ss+2
ss*3
```

Out[11]:
```
11     3
12     6
13     9
14    12
15    15
dtype: int64
```

In [20]:
```
ss.loc[11:14]
```

Out[20]:
```
11    1
12    2
13    3
14    4
dtype: int64
```

Creating Data frame with multiple series

In [23]:
```
data={
    'apples':[3,2,4,2],
    'oranges':[0,3,7,2]
}
purchase=pd.DataFrame(data)
print(purchase)
```

```
   apples  oranges
0       3        0
1       2        3
2       4        7
3       2        2
```

customize index and locate customer order

In [34]:
```
purchase=pd.DataFrame(data,index=['mick','robert','amit','allu arjun'])
print(purchase)
print(20*'-','customer order',20*'-')

print(purchase.loc['robert'])
```

```
            apples  oranges
mick             3        0
robert           2        3
amit             4        7
allu arjun       2        2
-------------------- customer order --------------------
apples     2
oranges    3
Name: robert, dtype: int64
```

In [38]:
```
df=pd.read_csv('temdata.csv')
print(df)
```

```
  city  temp
0  DWD    49
1  LXR    50
```

In [39]:
```
df=pd.read_csv('temdata.csv',index_col=0)
print(df)
```

```
      temp
city
DWD     49
LXR     50
```

In [40]:
```
df=pd.read_csv('temdata.csv',index_col=1)
print(df)
```

```
     city
temp
49    DWD
50    LXR
```

In [6]:
```
f=pd.read_csv('datasets/petrol_consumption.csv')
print(f)
f.shape
```

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence (%)
0	9.00	3571	1976	0.525
1	9.00	4092	1250	0.572
2	9.00	3865	1586	0.580
3	7.50	4870	2351	0.529
4	8.00	4399	431	0.544
5	10.00	5342	1333	0.571
6	8.00	5319	11868	0.451
7	8.00	5126	2138	0.553
8	8.00	4447	8577	0.529
9	7.00	4512	8507	0.552
10	8.00	4391	5939	0.530
11	7.50	5126	14186	0.525
12	7.00	4817	6930	0.574
13	7.00	4207	6580	0.545
14	7.00	4332	8159	0.608
15	7.00	4318	10340	0.586
16	7.00	4206	8508	0.572
17	7.00	3718	4725	0.540
18	7.00	4716	5915	0.724
19	8.50	4341	6010	0.677
20	7.00	4593	7834	0.663
21	8.00	4983	602	0.602
22	9.00	4897	2449	0.511
23	9.00	4258	4686	0.517
24	8.50	4574	2619	0.551
25	9.00	3721	4746	0.544
26	8.00	3448	5399	0.548
27	7.50	3846	9061	0.579
28	8.00	4188	5975	0.563

```
29        9.00            3601            4650                    0.4
93
30        7.00            3640            6905                    0.5
18
31        7.00            3333            6594                    0.5
13
32        8.00            3063            6524                    0.5
78
33        7.50            3357            4121                    0.5
47
34        8.00            3528            3495                    0.4
87
35        6.58            3802            7834                    0.6
29
36        5.00            4045           17782                    0.5
66
37        7.00            3897            6385                    0.5
86
38        8.50            3635            3274                    0.6
63
39        7.00            4345            3905                    0.6
72
40        7.00            4449            4639                    0.6
26
41        7.00            3656            3985                    0.5
63
42        7.00            4300            3635                    0.6
03
43        7.00            3745            2611                    0.5
08
44        6.00            5215            2302                    0.6
72
45        9.00            4476            3942                    0.5
71
46        7.00            4296            4083                    0.6
23
47        7.00            5002            9794                    0.5
93

    Petrol_Consumption
0                  541
1                  524
2                  561
3                  414
4                  410
5                  457
6                  344
7                  467
8                  464
9                  498
10                 580
11                 471
12                 525
13                 508
14                 566
15                 635
16                 603
17                 714
18                 865
19                 640
```

```
20                649
21                540
22                464
23                547
24                460
25                566
26                577
27                631
28                574
29                534
30                571
31                554
32                577
33                628
34                487
35                644
36                640
37                704
38                648
39                968
40                587
41                699
42                632
43                591
44                782
45                510
46                610
47                524
```

Out[6]: (48, 5)

In [48]:
```
f.tail()
```

Out[48]:

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol
43	7.0	3745	2611	0.508	
44	6.0	5215	2302	0.672	
45	9.0	4476	3942	0.571	
46	7.0	4296	4083	0.623	
47	7.0	5002	9794	0.593	

In [50]:
```
f.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 47
Data columns (total 5 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Petrol_tax                   48 non-null     float64
 1   Average_income               48 non-null     int64  
 2   Paved_Highways               48 non-null     int64  
 3   Population_Driver_licence(%)  48 non-null     float64
 4   Petrol_Consumption           48 non-null     int64  
dtypes: float64(2), int64(3)
memory usage: 2.0 KB
```

Handling duplicate return boolean series denoting rows

In [67]:
```
import pandas as pd

# Original DataFrame
emp = pd.DataFrame({
    'name': ['vinay', 'vijay', 'Dhoni', 'D BOSS', 'vinay', 'Dhoni'],
    'age': [25, 28, 25, 20, 25, 23],
    'salary': [547, 745, 78, 7545, 547, 7875]
})

# # Concatenating the DataFrame with itself to create duplicates
# emp = pd.concat([emp, emp])

# # Identifying duplicates but showing only the first occurrence
# duplicates = emp[emp.duplicated(keep='first')]  # Shows only the second
# print("Duplicates in the DataFrame (excluding first occurrence):")
# print(duplicates)

# Dropping duplicates and printing the result
emp.drop_duplicates(inplace=True)
print("\nDataFrame after dropping duplicates:")
print(emp)
```

```
DataFrame after dropping duplicates:
     name  age  salary
0   vinay   25     547
1   vijay   28     745
2   Dhoni   25      78
3  D BOSS   20    7545
5   Dhoni   23    7875
```

In [65]:
```
emp.duplicated(subset=['salary'])
emp.duplicated(subset=['age'])
emp.duplicated(subset=['name'])
```

Out[65]:
```
0    False
1    False
2    False
3    False
5     True
dtype: bool
```

cuatomize to required duplicate row with argument keep,first,last,false,default

In [68]:
```
print(emp)
print('keeping required duplicates element \n')
firstrow=emp.duplicated(keep='first')
lastrow=emp.duplicated(keep='last')
print('after manipulation')
print(firstrow)
print(lastrow)
```

```
     name  age  salary
0   vinay   25     547
1   vijay   28     745
2   Dhoni   25      78
3  D BOSS   20    7545
5   Dhoni   23    7875
keeping required duplicates element

after manipulation
0    False
1    False
2    False
3    False
5    False
dtype: bool
0    False
1    False
2    False
3    False
5    False
dtype: bool
```

In [69]:
```
from pathlib import Path
ffp=Path('TopRatedMovies.json')
ffp.touch()
ffp.exists()
```

Out[69]: True

In [72]:
```
mdf=pd.read_json('TopRatedMovies.json')
md.info
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
Cell In[72], line 1
----> 1 mdf=pd.read_json('TopRatedMovies.json')
      2 md.info()

File ~/.local/lib/python3.10/site-packages/pandas/io/json/_json.py:815, in
read_json(path_or_buf, orient, typ, dtype, convert_axes, convert_dates, ke
ep_default_dates, precise_float, date_unit, encoding, encoding_errors, lin
es, chunksize, compression, nrows, storage_options, dtype_backend, engine)
    813     return json_reader
    814 else:
--> 815     return json_reader.read()

File ~/.local/lib/python3.10/site-packages/pandas/io/json/_json.py:1025, i
n JsonReader.read(self)
   1023         obj = self._get_object_parser(self._combine_lines(data_lin
es))
   1024 else:
-> 1025     obj = self._get_object_parser(self.data)
   1026 if self.dtype_backend is not lib.no_default:
   1027     return obj.convert_dtypes(
   1028         infer_objects=False, dtype_backend=self.dtype_backend
   1029     )

File ~/.local/lib/python3.10/site-packages/pandas/io/json/_json.py:1051, i
n JsonReader._get_object_parser(self, json)
   1049 obj = None
   1050 if typ == "frame":
-> 1051     obj = FrameParser(json, **kwargs).parse()
   1053 if typ == "series" or obj is None:
   1054     if not isinstance(dtype, bool):

File ~/.local/lib/python3.10/site-packages/pandas/io/json/_json.py:1187, i
n Parser.parse(self)
   1185 @final
   1186 def parse(self):
-> 1187     self._parse()
   1189     if self.obj is None:
   1190         return None

File ~/.local/lib/python3.10/site-packages/pandas/io/json/_json.py:1402, i
n FrameParser._parse(self)
   1399 orient = self.orient
   1401 if orient == "columns":
-> 1402     self.obj = DataFrame(
   1403         ujson_loads(json, precise_float=self.precise_float), dtype
=None
   1404     )
   1405 elif orient == "split":
   1406     decoded = {
   1407         str(k): v
   1408         for k, v in ujson_loads(json, precise_float=self.precise_f
loat).items()
   1409     }

File ~/.local/lib/python3.10/site-packages/pandas/core/frame.py:778, in Da
taFrame.__init__(self, data, index, columns, dtype, copy)
```

```
    772     mgr = self._init_mgr(
    773         data, axes={"index": index, "columns": columns}, dtype=dty
pe, copy=copy
    774     )
    776 elif isinstance(data, dict):
    777     # GH#38939 de facto copy defaults to False only in non-dict ca
ses
--> 778     mgr = dict_to_mgr(data, index, columns, dtype=dtype, copy=cop
y, typ=manager)
    779 elif isinstance(data, ma.MaskedArray):
    780     from numpy.ma import mrecords

File ~/.local/lib/python3.10/site-packages/pandas/core/internals/construct
ion.py:503, in dict_to_mgr(data, index, columns, dtype, typ, copy)
    499     else:
    500         # dtype check to exclude e.g. range objects, scalars
    501         arrays = [x.copy() if hasattr(x, "dtype") else x for x in
arrays]
--> 503 return arrays_to_mgr(arrays, columns, index, dtype=dtype, typ=typ,
consolidate=copy)

File ~/.local/lib/python3.10/site-packages/pandas/core/internals/construct
ion.py:114, in arrays_to_mgr(arrays, columns, index, dtype, verify_integri
ty, typ, consolidate)
    111 if verify_integrity:
    112     # figure out the index, if necessary
    113     if index is None:
--> 114         index = _extract_index(arrays)
    115     else:
    116         index = ensure_index(index)

File ~/.local/lib/python3.10/site-packages/pandas/core/internals/construct
ion.py:667, in _extract_index(data)
    664         raise ValueError("Per-column arrays must each be 1-dimensi
onal")
    666 if not indexes and not raw_lengths:
--> 667     raise ValueError("If using all scalar values, you must pass an
index")
    669 if have_series:
    670     index = union_indexes(indexes)

ValueError: If using all scalar values, you must pass an index
```

viewing specified number of rows using head() function

In [13]:
```
import pandas as pd
df=pd.read_csv('datasets/petrol_consumption.csv')
print(df.head())
print(45*'-')
print(df.head(3))
```

```
   Petrol_tax  Average_income  Paved_Highways  Population_Driver_licence
(%)  \
0         9.0            3571            1976                       0.52
5
1         9.0            4092            1250                       0.57
2
2         9.0            3865            1586                       0.58
0
3         7.5            4870            2351                       0.52
9
4         8.0            4399             431                       0.54
4

   Petrol_Consumption
0                 541
1                 524
2                 561
3                 414
4                 410
-----------------------------------------
   Petrol_tax  Average_income  Paved_Highways  Population_Driver_licence
(%)  \
0         9.0            3571            1976                       0.52
5
1         9.0            4092            1250                       0.57
2
2         9.0            3865            1586                       0.58
0

   Petrol_Consumption
0                 541
1                 524
2                 561
```

In [14]:
```
df.tail()
```

Out[14]:

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol
43	7.0	3745	2611	0.508	
44	6.0	5215	2302	0.672	
45	9.0	4476	3942	0.571	
46	7.0	4296	4083	0.623	
47	7.0	5002	9794	0.593	

In [19]:
```
Tempdata=pd.read_csv("temdata.csv")
print(Tempdata)
print(60*'-')
Tempdata.info()
```

```
  city  temp
0  DWD    49
1  LXR    50
------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   city    2 non-null      object
 1   temp    2 non-null      int64
dtypes: int64(1), object(1)
memory usage: 160.0+ bytes
```

In [20]:
```
Tempdata.shape
```

Out[20]: (2, 2)

duplicates finding

In [29]:
```
Empd=pd.DataFrame({
    'name':['vinay','vivek','vinay','vinod','vinay'],
    'age':[25,28,25,30,25],
    'salary':[500,600,500,700,500]
})
Empd=pd.concat([Empd,Empd])
print(Empd,"\n")
print("Duplicate Elements")

print(Empd.drop_duplicates(inplace=False))
Empd.duplicated()
```

```
    name  age  salary
0  vinay   25     500
1  vivek   28     600
2  vinay   25     500
3  vinod   30     700
4  vinay   25     500
0  vinay   25     500
1  vivek   28     600
2  vinay   25     500
3  vinod   30     700
4  vinay   25     500

Duplicate Elements
    name  age  salary
0  vinay   25     500
1  vivek   28     600
3  vinod   30     700
```

Out[29]:
```
0    False
1    False
2     True
3    False
4     True
0     True
1     True
2     True
3     True
4     True
dtype: bool
```

to find duplicates on specifed collumns ->use subset

In [33]:
```
print(Empd.duplicated(subset=['salary']))
print("**********************************************************************")
Empd.duplicated(subset=['age'])
```

```
0    False
1    False
2     True
3    False
4     True
0     True
1     True
2     True
3     True
4     True
dtype: bool
**********************************************************************
```

Out[33]:
```
0    False
1    False
2     True
3    False
4     True
0     True
1     True
2     True
3     True
4     True
dtype: bool
```

customize to required duplicate row with argument keep,first,last,false,default

In [34]:
```
print(Empd)
print('keeping required duplicates element \n')
firstrow=Empd.duplicated(keep='first')
lastrow=Empd.duplicated(keep='last')
print('after manipulation')
print(firstrow)
print(lastrow)
```

```
    name  age  salary
0  vinay   25     500
1  vivek   28     600
2  vinay   25     500
3  vinod   30     700
4  vinay   25     500
0  vinay   25     500
1  vivek   28     600
2  vinay   25     500
3  vinod   30     700
4  vinay   25     500
keeping required duplicates element

after manipulation
0    False
1    False
2     True
3    False
4     True
0     True
1     True
2     True
3     True
4     True
dtype: bool
0     True
1     True
2     True
3     True
4     True
0     True
1    False
2     True
3    False
4    False
dtype: bool
```

In [41]:
```
mdf=pd.read_json('TopRatedMovies.json')
mdf.columns
mdf.columns=[columns.upper() for columns in mdf]
print(mdf)
```

```
   TITLE  GENERE  RELEASEYEAR  RATING
0  Devil  action         2023     4.5
1    RRR  action         2023     4.5
2     UI  action         2024     5.0
```

In [ ]: