In [7]:
```
import pandas as pd
subs=pd.read_csv("bollywood.csv",index_col="movie")
subs.value_counts()
```

Out[7]:
```
lead
Akshay Kumar        48
Amitabh Bachchan    45
Ajay Devgn          38
Salman Khan         31
Sanjay Dutt         26
                    ..
Raghuvir Yadav       1
Rahul Dev            1
Rahul Jaiswal        1
Rahul Khanna         1
Aadar Jain           1
Length: 566, dtype: int64
```

In [8]:
```
movies=subs.value_counts()
movies[movies>20]
```

Out[8]:
```
lead
Akshay Kumar        48
Amitabh Bachchan    45
Ajay Devgn          38
Salman Khan         31
Sanjay Dutt         26
Shah Rukh Khan      22
Emraan Hashmi       21
dtype: int64
```

In [12]:
```
marks={"ram":55,"shyam":60,"radha":70}
mark_series=pd.Series(marks,name="Result")
print(mark_series)
```

```
ram      55
shyam    60
radha    70
Name: Result, dtype: int64
```

Out[12]:
```
[55, 60, 70]
```

In [13]:
```
list(mark_series)
```

Out[13]:
```
[55, 60, 70]
```

In [14]:
```
dict(mark_series)
```

Out[14]:
```
{'ram': 55, 'shyam': 60, 'radha': 70}
```

In [16]:
```
mark_series[1]=100
mark_series
```

Out[16]:
```
ram       55
shyam    100
radha     70
Name: Result, dtype: int64
```

In [17]:
```
100+mark_series
```

Out[17]:
```
ram      155
shyam    200
radha    170
Name: Result, dtype: int64
```

In [18]:
```
mark_series
```

Out[18]:
```
ram       55
shyam    100
radha     70
Name: Result, dtype: int64
```

In [19]:
```
mark_series>=100
```

Out[19]:
```
ram      False
shyam     True
radha    False
Name: Result, dtype: bool
```

In [24]:
```
a=pd.Series([2,4,6,8,10])
b=pd.Series([1,3,5,7,10])
print(a+b)
print(a-b)
print(a==b)
print(a>b)
```

```
0     3
1     7
2    11
3    15
4    20
dtype: int64
0    1
1    1
2    1
3    1
4    0
dtype: int64
0    False
1    False
2    False
3    False
4     True
dtype: bool
0     True
1     True
2     True
3     True
4    False
dtype: bool
```

1.4 DataFrame

1. 2 Dinmensional data like 2D array or a table with roes and columns.

In [26]:
```
data={'calories':[400,434,656],'duration':[50,46,38]}
df=pd.DataFrame(data)
print(df)
```

```
   calories  duration
0       400        50
1       434        46
2       656        38
```

In [28]:
```
print(df.loc[1])
```

```
calories    434
duration     46
Name: 1, dtype: int64
```

In [29]:
```
print(df.iloc[1])
```

```
calories    434
duration     46
Name: 1, dtype: int64
```

In [30]:
```
df.calories
#df['calories']
```

Out[30]:
```
0    400
1    434
2    656
Name: calories, dtype: int64
```

In [31]:
```
df.loc[[0,1]]
```

Out[31]:

	calories	duration
0	400	50
1	434	46

In [33]:
```
df.loc[[1,2,3,4]]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-33-be503a41816a> in <module>
----> 1 df.loc[[1,2,3,4]]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in __getitem__(s
elf, key)
    877
    878             maybe_callable = com.apply_if_callable(key, self.obj)
--> 879             return self._getitem_axis(maybe_callable, axis=axis)
    880
    881     def _is_scalar_access(self, key: Tuple):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_axis
(self, key, axis)
   1097                     raise ValueError("Cannot index with multidimensional ke
y")
   1098
-> 1099                 return self._getitem_iterable(key, axis=axis)
   1100
   1101             # nested tuple slicing

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_iter
able(self, key, axis)
   1035
   1036         # A collection of keys
-> 1037         keyarr, indexer = self._get_listlike_indexer(key, axis, raise_missi
ng=False)
   1038         return self.obj._reindex_with_indexers(
   1039             {axis: [keyarr, indexer]}, copy=True, allow_dups=True

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _get_listlike
_indexer(self, key, axis, raise_missing)
   1252             keyarr, indexer, new_indexer = ax._reindex_non_unique(keyarr)
   1253
-> 1254         self._validate_read_indexer(keyarr, indexer, axis, raise_missing=ra
ise_missing)
   1255         return keyarr, indexer
   1256

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _validate_rea
d_indexer(self, key, indexer, axis, raise_missing)
   1313
   1314                 with option_context("display.max_seq_items", 10, "display.w
idth", 80):
-> 1315                     raise KeyError(
   1316                         "Passing list-likes to .loc or [] with any missing
labels "
   1317                         "is no longer supported. "

KeyError: "Passing list-likes to .loc or [] with any missing labels is no longer su
pported. The following labels were missing: Int64Index([3, 4], dtype='int64'). See
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#deprecate-loc
-reindex-listlike" (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#deprecate-loc-reindex-listlike")
```

In [37]:
```
df=pd.DataFrame(data,index=['day1','day2','day3'])
df.loc['day1']
```

Out[37]:
```
calories    400
duration     50
Name: day1, dtype: int64
```

In [36]:
```
df.loc[1]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(s
elf, key, method, tolerance)
   2894             try:
-> 2895                 return self._engine.get_loc(casted_key)
   2896             except KeyError as err:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTabl
e.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTabl
e.get_item()

KeyError: 1

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

KeyError                                  Traceback (most recent call last)
<ipython-input-36-1f09d9ff6611> in <module>
----> 1 df.loc[1]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in __getitem__(s
elf, key)
    877
    878             maybe_callable = com.apply_if_callable(key, self.obj)
--> 879             return self._getitem_axis(maybe_callable, axis=axis)
    880
    881     def _is_scalar_access(self, key: Tuple):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_axis
(self, key, axis)
   1108         # fall thru to straight lookup
   1109         self._validate_key(key, axis)
-> 1110         return self._get_label(key, axis=axis)
   1111
   1112     def _get_slice_axis(self, slice_obj: slice, axis: int):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _get_label(se
lf, label, axis)
   1057     def _get_label(self, label, axis: int):
   1058         # GH#5667 this will fail if the label is not present in the axis.
-> 1059         return self.obj.xs(label, axis=axis)
   1060
   1061     def _handle_lowerdim_multi_index_axis0(self, tup: Tuple):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in xs(self, key,
axis, level, drop_level)
   3489             loc, new_index = self.index.get_loc_level(key, drop_level=drop_
level)
   3490         else:
-> 3491             loc = self.index.get_loc(key)
   3492
   3493             if isinstance(loc, np.ndarray):
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   2895                 return self._engine.get_loc(casted_key)
   2896             except KeyError as err:
-> 2897                 raise KeyError(key) from err
   2898
   2899         if tolerance is not None:

KeyError: 1
```

In [41]:
```
df.iloc[0]
```

Out[41]:
```
calories    400
duration     50
Name: day1, dtype: int64
```

In [42]:
```
dataset=pd.read_csv("auto-mpg.csv")
dataset
```

Out[42]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

In [43]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [44]:
```
dataset.describe()
```

Out[44]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

In [49]:
```
import pandas as np
dataset.describe(inculde=[np.number])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-49-f0c43c5eb296> in <module>
      1 import pandas as np
----> 2 dataset.describe(inculde=[np.number])

C:\ProgramData\Anaconda3\lib\site-packages\pandas\__init__.py in __getattr__(name)
    256             return _SparseArray
    257
--> 258         raise AttributeError(f"module 'pandas' has no attribute '{name}'")
    259
    260

AttributeError: module 'pandas' has no attribute 'number'
```

In [52]:
```
dataset[['mpg','cylinders']].describe()
```

Out[52]:

	mpg	cylinders
count	398.000000	398.000000
mean	23.514573	5.454774
std	7.815984	1.701004
min	9.000000	3.000000
25%	17.500000	4.000000
50%	23.000000	4.000000
75%	29.000000	8.000000
max	46.600000	8.000000

In [53]:
```
dataset.describe(percentiles=[.30,.45,.60])
```

Out[53]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
30%	18.000000	4.000000	112.000000	2301.000000	14.200000	73.000000	1.000000
45%	21.065000	4.000000	140.000000	2670.650000	15.000000	75.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
60%	25.000000	6.000000	200.000000	3085.200000	16.000000	77.000000	1.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

In [55]:
```
dataset[0:2].describe()
```

Out[55]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	2.00000	2.0	2.000000	2.000000	2.000000	2.0	2.0
mean	16.50000	8.0	328.500000	3598.500000	11.750000	70.0	1.0
std	2.12132	0.0	30.405592	133.643182	0.353553	0.0	0.0
min	15.00000	8.0	307.000000	3504.000000	11.500000	70.0	1.0
25%	15.75000	8.0	317.750000	3551.250000	11.625000	70.0	1.0
50%	16.50000	8.0	328.500000	3598.500000	11.750000	70.0	1.0
75%	17.25000	8.0	339.250000	3645.750000	11.875000	70.0	1.0
max	18.00000	8.0	350.000000	3693.000000	12.000000	70.0	1.0

In [56]:
```
dataset.loc[15]
```

Out[56]:
```
mpg                          22
cylinders                     6
displacement                198
horsepower                   95
weight                     2833
acceleration               15.5
model year                   70
origin                        1
car name        plymouth duster
Name: 15, dtype: object
```

In [57]:
```
dataset.loc[5:15]
```

Out[57]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
5	15.0	8	429.0	198	4341	10.0	70	1	ford galaxie 500
6	14.0	8	454.0	220	4354	9.0	70	1	chevrolet impala
7	14.0	8	440.0	215	4312	8.5	70	1	plymouth fury iii
8	14.0	8	455.0	225	4425	10.0	70	1	pontiac catalina
9	15.0	8	390.0	190	3850	8.5	70	1	amc ambassador dpl
10	15.0	8	383.0	170	3563	10.0	70	1	dodge challenger se
11	14.0	8	340.0	160	3609	8.0	70	1	plymouth 'cuda 340
12	15.0	8	400.0	150	3761	9.5	70	1	chevrolet monte carlo
13	14.0	8	455.0	225	3086	10.0	70	1	buick estate wagon (sw)
14	24.0	4	113.0	95	2372	15.0	70	3	toyota corona mark ii
15	22.0	6	198.0	95	2833	15.5	70	1	plymouth duster

In [58]:
```
dataset.iloc[5:15]
```

Out[58]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
5	15.0	8	429.0	198	4341	10.0	70	1	ford galaxie 500
6	14.0	8	454.0	220	4354	9.0	70	1	chevrolet impala
7	14.0	8	440.0	215	4312	8.5	70	1	plymouth fury iii
8	14.0	8	455.0	225	4425	10.0	70	1	pontiac catalina
9	15.0	8	390.0	190	3850	8.5	70	1	amc ambassador dpl
10	15.0	8	383.0	170	3563	10.0	70	1	dodge challenger se
11	14.0	8	340.0	160	3609	8.0	70	1	plymouth 'cuda 340
12	15.0	8	400.0	150	3761	9.5	70	1	chevrolet monte carlo
13	14.0	8	455.0	225	3086	10.0	70	1	buick estate wagon (sw)
14	24.0	4	113.0	95	2372	15.0	70	3	toyota corona mark ii

In [59]:
```
dataset['mpg'].loc[4]
```

Out[59]:
```
17.0
```

In [61]:
```
dataset.shape
```

Out[61]:
```
(398, 9)
```

In [62]:
```
dataset.shape[0]
```

Out[62]:
```
398
```

In [63]:
```
dataset.corr()
```

Out[63]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.420289	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.505419	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.543684	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.417457	-0.306564	-0.581024
acceleration	0.420289	-0.505419	-0.543684	-0.417457	1.000000	0.288137	0.205873
model year	0.579267	-0.348746	-0.370164	-0.306564	0.288137	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.205873	0.180662	1.000000

1.5 Scatter Matrix/ Pair Plots

In [68]:
```
import matplotlib.pyplot as plt
pd.plotting.scatter_matrix(dataset,figsize=[20,20],marker='v',alpha=0.5,diagonal='kd
plt.show()
```