```
import pandas as pd
```

Problem 1: Take the data sets Append_1.csv, and Append_2.csv and append the two sets together. Name the new data set Append.

```
append_1 = pd.read_csv("Append_1.csv", index_col=0)
append_2 = pd.read_csv("Append_2.csv", index_col=0)
```

```
append_1
```

	Id	Score
1	78917851	13
2	34554367	77
3	22173883	10

```
append_2
```

	Id	Score
1	56993289	72
2	26856261	51
3	33921834	99
4	97613637	63
5	78816868	28
6	67731229	17

```
Append = pd.concat([append_1, append_2])
Append.to_csv("Append.csv")
Append
```

	Id	Score
1	78917851	13
2	34554367	77
3	22173883	10
1	56993289	72
2	26856261	51
3	33921834	99
4	97613637	63
5	78816868	28
6	67731229	17

Problem 2: Take the data sets Merge_1.csv and Merge_2.csv and perform an

inner join, left join, right join, full join. Name the resulting data sets, Inner, Left, Right, and Full.

```
merge_1 = pd.read_csv("Merge_1.csv", index_col=0)
merge_2 = pd.read_csv("Merge_2.csv", index_col=0)
```

```
merge_1
```

	Id	Score
4	68134933	71
7	22113381	69
9	31937926	98
2	17245265	41
3	42428425	9
10	92922546	67
1	31674694	96

```
merge_2
```

	Id	Score
8	23525437	54
7	22113381	69
9	31937926	98
2	17245265	41
10	92922546	67
6	38672872	76
1	31674694	96

Inner

```
Inner = pd.merge(merge_1, merge_2, on='Id', how='inner')
Inner
```

	Id	Score_x	Score_y
0	22113381	69	69
1	31937926	98	98
2	17245265	41	41
3	92922546	67	67
4	31674694	96	96

Left

```
Left = pd.merge(merge_1, merge_2, on='Id', how='left')
Left
```

	Id	Score_x	Score_y
0	68134933	71	NaN
1	22113381	69	69.0
2	31937926	98	98.0
3	17245265	41	41.0
4	42428425	9	NaN
5	92922546	67	67.0
6	31674694	96	96.0

```
<ipython-input-116-eab92df07757>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  index = ["Id"] + list(Vowels.mean()[Vowels.mean() > 0].index)
```

	Id	V6	V7	V8	V9	V10
1	a	-0.030081	0.228564	0.509134	0.816408	1.209791
2	a	-0.029370	0.228574	0.510424	0.817482	1.215493
3	a	-0.027879	0.230532	0.511642	0.818404	1.224739
4	a	-0.027428	0.235311	0.515277	0.819011	1.225300
21	e	0.033444	0.284396	0.559895	0.871592	1.354335
22	e	0.035520	0.289390	0.560585	0.873613	1.359896
35	i	0.072672	0.326803	0.603491	0.936952	1.461178
36	i	0.073014	0.330990	0.613375	0.937777	1.469812
55	o	0.140601	0.395279	0.662332	1.025431	1.634369

Problem 4. Take the Filter.csv dataset, and take a simple random sample of the data with ten rows. Keep all columns. Name the new dataset SRS_Filter.

```
SRS_Filter = Filter.sample(n=10)
SRS_Filter
```

	Id	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
70	r	-1.512410	-0.962193	-0.594802	-0.339151	-0.107537	0.169531	0.435238	0.730242	1.094578	1.814198
49	l	-1.656177	-1.046861	-0.674250	-0.380643	-0.163883	0.121138	0.371061	0.648888	0.994604	1.592101
14	d	-2.194228	-1.220047	-0.786685	-0.484320	-0.244800	0.012640	0.268868	0.535639	0.850074	1.302671
19	d	-2.093890	-1.176263	-0.763212	-0.469929	-0.231471	0.027452	0.273076	0.546232	0.862737	1.342850
29	g	-1.907103	-1.127302	-0.746737	-0.433587	-0.212005	0.051428	0.307884	0.585474	0.910425	1.421806
99	z	-1.300555	-0.844371	-0.517238	-0.280365	-0.033863	0.225745	0.498943	0.809836	1.204194	2.583077
11	c	-2.248656	-1.228823	-0.796115	-0.487631	-0.252037	0.004767	0.255484	0.529901	0.840513	1.278978
72	s	-1.507544	-0.958284	-0.586552	-0.338557	-0.106609	0.171311	0.439231	0.734304	1.101666	1.849731
47	l	-1.674005	-1.057892	-0.680548	-0.386315	-0.167965	0.113935	0.356816	0.645855	0.987294	1.571017
83	u	-1.402569	-0.908152	-0.561200	-0.313108	-0.068641	0.193865	0.465258	0.769225	1.135336	2.136592

Problem 5. Randomly partition the Filter.csv data set into three subsets:

Train (70% of your data), Validation (15% of your data), Test (15% of your data). For each of the three subsets, print the first three rows and the dimensions of the data set.

This can be easily done using scikit learn's train_test split.

```
from sklearn.model_selection import train_test_split

train, rest = train_test_split(Filter, train_size=0.7)
val, test = train_test_split(rest, test_size=0.5)
```

Train

```
train.head(3)
```

	Id	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
34	h	-1.848870	-1.108606	-0.728289	-0.411311	-0.204454	0.072663	0.326538	0.597154	0.935854	1.449525
2	a	-2.753743	-1.293420	-0.830057	-0.510220	-0.271765	-0.029370	0.228574	0.510424	0.817482	1.215493
52	m	-1.641384	-1.032022	-0.670981	-0.375518	-0.162222	0.126508	0.384024	0.652923	1.021864	1.625707

```
train.shape
```

```
(70, 11)
```

Validation

```
val.head(3)
```

	Id	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
62	p	-1.583904	-0.989101	-0.629909	-0.365573	-0.123204	0.156192	0.414330	0.692836	1.051478	1.694691
49	l	-1.656177	-1.046861	-0.674250	-0.380643	-0.163883	0.121138	0.371061	0.648888	0.994604	1.592101
90	w	-1.362826	-0.887245	-0.542298	-0.293752	-0.056517	0.212445	0.482841	0.793072	1.170550	2.298134

```
val.shape
```

```
(15, 11)
```

Test

```
test.head(3)
```

	Id	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
24	f	-1.977538	-1.145779	-0.750031	-0.453491	-0.222002	0.036691	0.295670	0.568302	0.890023	1.375874
76	s	-1.460825	-0.941320	-0.577046	-0.324442	-0.101695	0.177586	0.456526	0.750048	1.123588	1.888790
20	d	-2.076374	-1.175226	-0.762930	-0.465575	-0.231032	0.027937	0.284120	0.556447	0.869824	1.348264

```
test.shape
```

```
(15, 11)
```