Assignment-1

Aim:

1. Get the nrows, ncolumns, datatype, summary stats of each column of a dataframe? Also get the array and list equivalent.

2. Extract the row and column number of a particular cell with given criterion?

3. Rename a specific columns in a dataframe?

4. Count the number of missing values in each column?

5. Replace missing values of multiple numeric columns with the mean?

6. Replace one missing value with Imputer class.

7. Apply function on existing columns with global variables as additional arguments?

8. Change the order of columns of a dataframe?

9. Set the number of rows and columns displayed in the output?

10. Create a primary key index by combining relevant columns?

11. Get the row number of the nth largest value in a column?

12. Find the position of the nth largest value greater than a given value?

13. Get the last n rows of a dataframe with row sum > 100?

14. Create a column containing the minimum by maximum of each row?

15. Convert categorical values into numerical using one hot and label encoding?

16. Normalize some columns in a dataframe using minmax normalization?

17. Normalize some columns in a dataframe using zscore normalization?

18. Import a file from a url into a dataframe link:
http://www.fdic.gov/bank/individual/failed/banklist.html

19. Print count of unique values in a column

20. Sort values by col2 in descending order

Description:
In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating nu-

merical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

Series: Series is a one-dimensional labeled array capable of holding any data type.

1.

```
In [40]:
1 df = pd.read_csv('Automobile_data.csv')
```

```
In [41]:
1 df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61 entries, 0 to 60
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   index             61 non-null     int64
 1   company           57 non-null     object
 2   body-style        61 non-null     object
 3   wheel-base        58 non-null     float64
 4   length            59 non-null     float64
 5   engine-type       61 non-null     object
 6   num-of-cylinders  60 non-null     object
 7   horsepower        61 non-null     int64
 8   average-mileage   61 non-null     int64
 9   price             58 non-null     float64
dtypes: float64(3), int64(3), object(4)
memory usage: 4.9+ KB
```

```
In [5]:
1 df.shape

(61, 10)
```

```
In [8]:
1 type(df)

pandas.core.frame.DataFrame
```

In [9]:
```
1 df.head(10)
```

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	price
0	0	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	13495.0
1	1	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	16500.0
2	2	NaN	hatchback	94.5	171.2	ohcv	six	154	19	16500.0
3	3	audi	sedan	99.8	176.6	ohc	four	102	24	13950.0
4	4	audi	sedan	99.4	176.6	ohc	five	115	18	17450.0
5	5	audi	sedan	99.8	177.3	ohc	five	110	19	15250.0
6	6	audi	wagon	105.8	192.7	ohc	five	110	19	18920.0
7	9	NaN	sedan	101.2	176.8	ohc	four	101	23	16430.0
8	10	bmw	sedan	NaN	176.8	ohc	four	101	23	16925.0
9	11	bmw	sedan	NaN	176.8	ohc	six	121	21	20970.0

In [10]:
```
1 df[df.columns[0:3]]
```

	index	company	body-style
0	0	alfa-romero	convertible
1	1	alfa-romero	convertible
2	2	NaN	hatchback
3	3	audi	sedan
4	4	audi	sedan
...	...	...	...
56	81	NaN	sedan
57	82	volkswagen	sedan
58	86	volkswagen	sedan
59	87	volvo	sedan
60	88	volvo	wagon

61 rows × 3 columns

In [11]:
```
1 df.describe()
```

	index	wheel-base	length	horsepower	average-mileage	price
count	61.000000	58.000000	59.000000	61.000000	61.000000	58.000000
mean	40.885246	98.301724	173.015254	107.852459	25.803279	15387.000000
std	25.429706	6.798981	13.845313	53.524398	8.129821	11320.259841
min	0.000000	88.400000	141.100000	48.000000	13.000000	5151.000000
25%	18.000000	94.500000	162.200000	68.000000	19.000000	6808.500000
50%	39.000000	96.000000	171.200000	100.000000	25.000000	11095.000000
75%	61.000000	101.000000	177.050000	123.000000	31.000000	18120.500000
max	88.000000	120.900000	208.100000	288.000000	47.000000	45400.000000

In [12]:

```
# Python
## Extract the third row
df.iloc[2]
### or ###

```

```
index                   2
company               NaN
body-style          hatchback
wheel-base             94.5
length                171.2
engine-type            ohcv
num-of-cylinders        six
horsepower              154
average-mileage          19
price                 16500
Name: 2, dtype: object
```

In [15]:

```
## Extract the first three rows
df.iloc[:3]

```

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	price
0	0	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	13495.0
1	1	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	16500.0
2	2	NaN	hatchback	94.5	171.2	ohcv	six	154	19	16500.0

2.

In [20]:

```
df[df['company']=='audi']['horsepower']
```

```
3    102
4    115
5    110
6    110
Name: horsepower, dtype: int64
```

3.

In [21]:
```
#Rename
df.rename(columns={"wheel-base": "wheel_base"})
```

	index	company	body-style	wheel_base	length	engine-type	num-of-cylinders	horsepower	average-mileage	price
0	0	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	13495.0
1	1	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	16500.0
2	2	NaN	hatchback	94.5	171.2	ohcv	six	154	19	16500.0
3	3	audi	sedan	99.8	176.6	ohc	four	102	24	13950.0
4	4	audi	sedan	99.4	176.6	ohc	five	115	18	17450.0
...	...	...	...	...	...	...	...	...	...	...
56	81	NaN	sedan	97.3	171.7	ohc	four	85	27	7975.0
57	82	volkswagen	sedan	97.3	171.7	ohc	four	52	37	7995.0
58	86	volkswagen	sedan	97.3	171.7	ohc	four	100	26	9995.0
59	87	volvo	sedan	104.3	188.8	ohc	four	114	23	12940.0
60	88	volvo	wagon	104.3	188.8	ohc	four	114	23	13415.0

61 rows × 10 columns

4.

In [22]:
```
#Null values in each column
df.isnull().sum()
```

```
index              0
company            4
body-style         0
wheel-base         3
length             2
engine-type        0
num-of-cylinders   1
horsepower         0
average-mileage    0
price              3
dtype: int64
```

5.

In [23]:
```
df['wheel-base'].fillna(df['wheel-base'].mean(), inplace=True)
df['length'].fillna(df['length'].mean(), inplace=True)
df.isnull().sum()
```

```
index              0
company            4
body-style         0
wheel-base         0
length             0
engine-type        0
num-of-cylinders   1
horsepower         0
average-mileage    0
price              3
dtype: int64
```

6.

```
In [24]:
1 from  sklearn.impute import SimpleImputer
2 imputer = SimpleImputer(strategy='mean')
3 imputer = imputer.fit(df[['price']])
4 df[['price']]=imputer.transform(df[['price']])
5 df.isnull().sum()
```

```
index              0
company            4
body-style         0
wheel-base         0
length             0
engine-type        0
num-of-cylinders   1
horsepower         0
average-mileage    0
price              0
dtype: int64
```

```
In [25]:
1 df[['price']]
```

	price
0	13495.0
1	16500.0
2	16500.0
3	13950.0
4	17450.0
...	...
56	7975.0
57	7995.0
58	9995.0
59	12940.0
60	13415.0

61 rows × 1 columns

7.

```
In [26]:
1 df['new_col']=df.price.apply(lambda x:x*2)
```

```
In [27]:
1 df[['new_col']]
```

	new_col
0	26980.0
1	33000.0
2	33000.0
3	27900.0
4	34900.0
...	...
56	15950.0
57	15980.0
58	19980.0
59	25880.0
60	26830.0

61 rows × 1 columns

8.

```
In [28]:
1 df.columns
```

```
Index(['index', 'company', 'body-style', 'wheel-base', 'length', 'engine-type',
       'num-of-cylinders', 'horsepower', 'average-mileage', 'price',
       'new_col'],
      dtype='object')
```

```
In [29]:
1 df=df[['index','average-mileage','body-style','company','engine-type','horsepower','num-of-cylinders','
2 df.columns
```

```
Index(['index', 'average-mileage', 'body-style', 'company', 'engine-type',
       'horsepower', 'num-of-cylinders', 'price', 'new_col'],
      dtype='object')
```

9.

```
In [30]:
pd.set_option('display.max_rows', 10)
pd.set_option('display.max_columns', 10)
pd.set_option('display.width', 6)
df
```

	index	average-mileage	body-style	company	engine-type	horsepower	num-of-cylinders	price	new_col
0	0	21	convertible	alfa-romero	dohc	111	four	13495.0	26990.0
1	1	21	convertible	alfa-romero	dohc	111	four	16500.0	33000.0
2	2	19	hatchback	NaN	ohcv	154	six	16500.0	33000.0
3	3	24	sedan	audi	ohc	102	four	13950.0	27900.0
4	4	18	sedan	audi	ohc	115	five	17450.0	34900.0
...	...	...	...	...	...	...	...	...	...
56	81	27	sedan	NaN	ohc	85	four	7975.0	15950.0
57	82	37	sedan	volkswagen	ohc	52	four	7995.0	15990.0
58	86	26	sedan	volkswagen	ohc	100	four	9995.0	19990.0
59	87	23	sedan	volvo	ohc	114	four	12940.0	25880.0
60	88	23	wagon	volvo	ohc	114	four	13415.0	26830.0

61 rows × 9 columns

10.

```
In [53]:
df2=df.set_index(["index", "company"],append = True, drop = False)
df2.head()
```

			index	company	body-style	wheel-base	length	...	horsepower	average-mileage	price	new_col	body_style_cat
	index	company											
0	0	alfa-romero	0	alfa-romero	convertible	88.6	168.8	...	111	21	13495.0	-0.273592	0
1	1	alfa-romero	1	alfa-romero	convertible	88.6	168.8	...	111	21	16500.0	-0.271915	0
2	2	NaN	2	NaN	hatchback	94.5	171.2	...	154	19	16500.0	-0.270239	2
3	3	audi	3	audi	sedan	99.8	176.6	...	102	24	13950.0	-0.267644	3
4	4	audi	4	audi	sedan	99.4	176.6	...	115	18	17450.0	-0.267252	3

5 rows × 12 columns

11.

```
In [32]:
n=int(input())#if n==1
df.nlargest(n, ['price'])
```

```
1
```

		index	company	average-mileage	body-style	company	engine-type	horsepower	num-of-cylinders	price	new_col
	index	company									
35	47	mercedes-benz	47	14	hardtop	mercedes-benz	ohcv	184	eight	45400.0	90800.0

12.

```
In [33]:
k=int(input())
t=df[df['price']>k].price
n=1
t=t.nlargest(n)
t
```

```
31000

    index  company
35  47     mercedes-benz    45400.0
Name: price, dtype: float64
```

13.

```
In [36]:
tmp=df
tmp['row_sum']=tmp.sum(axis=1)
tmp=tmp[tmp['row_sum']>=100].row_sum
n=int(input())
tmp.tail(n)
```

```
2

    index  company
59  87     volvo      39044.0
60  88     volvo      40470.0
Name: row_sum, dtype: float64
```

14.

```
In [42]:
tmp=df
tmp['new_col']=tmp.min(axis=1)/tmp.max(axis=1)
tmp
```

	index	company	body-style	wheel-base	length	...	num-of-cylinders	horsepower	average-mileage	price	new_col
0	0	alfa-romero	convertible	88.6	168.8	...	four	111	21	13495.0	0.000000
1	1	alfa-romero	convertible	88.6	168.8	...	four	111	21	16500.0	0.000061
2	2	NaN	hatchback	94.5	171.2	...	six	154	19	16500.0	0.000121
3	3	audi	sedan	99.8	176.6	...	four	102	24	13950.0	0.000215
4	4	audi	sedan	99.4	176.6	...	five	115	18	17450.0	0.000229
...	...	...	...	...	...	...	...	...	...	...	...
56	81	NaN	sedan	97.3	171.7	...	four	85	27	7975.0	0.003386
57	82	volkswagen	sedan	97.3	171.7	...	four	52	37	7995.0	0.004628
58	86	volkswagen	sedan	97.3	171.7	...	four	100	26	9995.0	0.002601
59	87	volvo	sedan	104.3	188.8	...	four	114	23	12940.0	0.001777
60	88	volvo	wagon	104.3	188.8	...	four	114	23	13415.0	0.001714

61 rows × 11 columns

15.

In [45]:

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
df['body_style_cat'] = labelencoder.fit_transform(df['body-style'])
df.head(5)
```

	index	company	body-style	wheel-base	length	...	horsepower	average-mileage	price	new_col	body_style_cat
0	0	alfa-romero	convertible	88.6	168.8	...	111	21	13495.0	0.000000	0
1	1	alfa-romero	convertible	88.6	168.8	...	111	21	16500.0	0.000061	0
2	2	NaN	hatchback	94.5	171.2	...	154	19	16500.0	0.000121	2
3	3	audi	sedan	99.8	176.6	...	102	24	13950.0	0.000215	3
4	4	audi	sedan	99.4	176.6	...	115	18	17450.0	0.000229	3

5 rows × 12 columns

In [46]:

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(handle_unknown='ignore')
enc_df = pd.DataFrame(enc.fit_transform(df[['body_style_cat']]).toarray())
enc_df
```

	0	1	2	3	4
0	1.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0
4	0.0	0.0	0.0	1.0	0.0
...	...	...	...	...	...
56	0.0	0.0	0.0	1.0	0.0
57	0.0	0.0	0.0	1.0	0.0
58	0.0	0.0	0.0	1.0	0.0
59	0.0	0.0	0.0	1.0	0.0
60	0.0	0.0	0.0	0.0	1.0

61 rows × 5 columns

16.

In [47]:

```
from sklearn import preprocessing
mnorm=preprocessing.MinMaxScaler()
scaled=mnorm.fit_transform(df['price'].values.reshape(-1,1))
pf=pd.DataFrame(scaled)
pf
```

	0
0	0.207309
1	0.281970
2	0.281970
3	0.218614
4	0.305573
...	...
56	0.070163
57	0.070660
58	0.120351
59	0.193520
60	0.205322

61 rows × 1 columns

17.

In [48]:

```
from scipy.stats import zscore
df[['new_col']]=df[['new_col']].apply(zscore)
df['new_col'].head(5)
```

```
0   -0.273592
1   -0.271915
2   -0.270239
3   -0.267644
4   -0.267252
Name: new_col, dtype: float64
```

18.

In [49]:

```
import io
import requests
url="http://www.fdic.gov/bank/individual/failed/banklist.csv"
s=requests.get(url).content
c=pd.read_csv(io.StringIO(s.decode('utf-8')))
c
```

	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date
0	The First State Bank	Barboursville	WV	14361	MVB Bank, Inc.	3-Apr-20
1	Ericson State Bank	Ericson	NE	18265	Farmers and Merchants Bank	14-Feb-20
2	City National Bank of New Jersey	Newark	NJ	21111	Industrial Bank	1-Nov-19
3	Resolute Bank	Maumee	OH	58317	Buckeye State Bank	25-Oct-19
4	Louisa Community Bank	Louisa	KY	58112	Kentucky Farmers Bank Corporation	25-Oct-19
...	...	...	...	...	...	...
556	Superior Bank, FSB	Hinsdale	IL	32646	Superior Federal, FSB	27-Jul-01
557	Malta National Bank	Malta	OH	6629	North Valley Bank	3-May-01
558	First Alliance Bank & Trust Co.	Manchester	NH	34264	Southern New Hampshire Bank & Trust	2-Feb-01
559	National State Bank of Metropolis	Metropolis	IL	3815	Banterra Bank of Marion	14-Dec-00
560	Bank of Honolulu	Honolulu	HI	21029	Bank of the Orient	13-Oct-00

561 rows × 6 columns

19.

In [51]:

```
df['body-style'].value_counts()
```

```
sedan          32
hatchback      15
wagon           9
convertible     3
hardtop         2
Name: body-style, dtype: int64
```

20.

In [52]:

```
srt_df=df.sort_values('average-mileage',ascending=False)
srt_df.head(5)
```

	index	company	body-style	wheel-base	length	...	horsepower	average-mileage	price	new_col	body_style_cat
13	16	chevrolet	hatchback	88.4	141.1	...	48	47	5151.0	-0.187685	2
40	53	nissan	sedan	94.5	165.3	...	55	45	7099.0	-0.098280	3
15	18	chevrolet	sedan	94.5	158.8	...	70	38	6575.0	-0.197878	3
23	32	isuzu	sedan	94.5	155.9	...	70	38	NaN	5.403161	3
22	31	isuzu	sedan	94.5	155.9	...	70	38	NaN	5.225762	3

5 rows × 12 columns