a) Creating and loading different datasets in Python.

In [79]:
```
import pandas as pd
import numpy as np
```

In [80]:
```
df = pd.DataFrame({'Names':['Nayan', 'aditya', 'Om', 'Shrawani', 'Ninad', 'Prajw
                   'Marks':['82', '98', '11', '52', '88', '98', '56', '92', '52'
```

In [81]:
```
df
```

Out[81]:

	Names	Marks
0	Nayan	82
1	aditya	98
2	Om	11
3	Shrawani	52
4	Ninad	88
5	Prajwal	98
6	Ayush	56
7	Sumedh	92
8	Yash	52
9	Pranav	22

In [83]:
```
df.to_csv("name_and_marks.csv",index=False)
```

In [84]:
```
a1 = pd.read_csv('name_and_marks.csv')
```

In [85]:
```
a1
```

Out[85]:

	Names	Marks
0	Nayan	82
1	aditya	98
2	Om	11
3	Shrawani	52
4	Ninad	88
5	Prajwal	98
6	Ayush	56
7	Sumedh	92
8	Yash	52
9	Pranav	22

In [ ]:
```
################################################################################
```

swiggy_file dataset operations

In [3]:
```
import numpy as np
import pandas as pd
df = pd.DataFrame()
df = pd.read_csv("swiggy_file.csv")
```

In [4]:
```
df
```

Out[4]:

	Restaurant Name	Cuisine	Rating	Number of Ratings	Average Price	Number of Offers	Offer Name	
0	La Pino'Z Pizza	Pizzas, Pastas	4.0	10+ ratings	₹250 for two	2	FLAT DEAL\r\nFLAT ₹125 OFF\r\nUSE FLAT125ABOVE...	N
1	The Second Wife	Indian, North Indian	3.6	50+ ratings	₹250 for two	2	30% OFF UPTO ₹75\r\nUSE TRYNEWABOVE ₹149, FLAT...	
2	Tasty Bites	Italian, Beverages	3.8	10+ ratings	₹200 for two	1	FLAT ₹120 OFF\r\nUSE AXIS120ABOVE ₹500	
3	Food Studio	Pizzas, Burgers	3.5	8 ratings	₹49 for two	5	50% OFF UPTO ₹100\r\nUSE TRYNEWABOVE ₹129, FLA...	
4	Roll Express	Fast Food, Snacks	4.3	100+ ratings	₹200 for two	2	DEAL OF DAY\r\n10% OFF UPTO ₹40\r\nUSE STEALDE...	
...	...	...	...	...	...	...	...	
140652	Yummy Momo'S Cafe	Chinese, Fast Food	4.6	3 ratings	₹100 for two	3	20% OFF UPTO ₹50\r\nUSE TRYNEWABOVE ₹149, FLAT...	Y
140653	CAFE FIRST FLOOR	Beverages, Snacks	3.2	3 ratings	₹200 for two	2	FLAT ₹120 OFF\r\nUSE AXIS120ABOVE ₹500, FLAT ₹...	Y
140654	Cafe Coffee Aani Barach Kahi	Snacks	3.2	50+ ratings	₹150 for two	2	FLAT ₹120 OFF\r\nUSE AXIS120ABOVE ₹500, FLAT ₹...	Y
140655	Patil Family Restaurant	North Indian, Biryani	4.3	9 ratings	₹200 for two	2	FLAT ₹120 OFF\r\nUSE AXIS120ABOVE ₹500, FLAT ₹...	Y
140656	Prabhakar Mama Cha Dhaba	North Indian	--	Too Few Ratings	₹350 for two	2	FLAT ₹120 OFF\r\nUSE AXIS120ABOVE ₹500, FLAT ₹...	Y

140657 rows × 10 columns

In [5]:
```
df.head()
```

Out[5]:

	Restaurant Name	Cuisine	Rating	Number of Ratings	Average Price	Number of Offers	Offer Name	Area
0	La Pino'Z Pizza	Pizzas, Pastas	4.0	10+ ratings	₹250 for two	2	FLAT DEAL\r\nFLAT ₹125 OFF\r\nUSE FLAT125ABOVE...	LALA LAJPAT RA MARKET
1	The Second Wife	Indian, North Indian	3.6	50+ ratings	₹250 for two	2	30% OFF UPTO ₹75\r\nUSE TRYNEWABOVE ₹149, FLAT...	Centra Abohar
2	Tasty Bites	Italian, Beverages	3.8	10+ ratings	₹200 for two	1	FLAT ₹120 OFF\r\nUSE AXIS120ABOVE ₹500	Centra Abohar
3	Food Studio	Pizzas, Burgers	3.5	8 ratings	₹49 for two	5	50% OFF UPTO ₹100\r\nUSE TRYNEWABOVE ₹129, FLA...	Centra Abohar
4	Roll Express	Fast Food, Snacks	4.3	100+ ratings	₹200 for two	2	DEAL OF DAY\r\n10% OFF UPTO ₹40\r\nUSE STEALDE...	Circular Road

In [6]:
```
df.tail()
```

Out[6]:

	Restaurant Name	Cuisine	Rating	Number of Ratings	Average Price	Number of Offers	Offer Name	
140652	Yummy Momo'S Cafe	Chinese, Fast Food	4.6	3 ratings	₹100 for two	3	20% OFF UPTO ₹50\r\nUSE TRYNEWABOVE ₹149, FLAT...	Ya L
140653	CAFE FIRST FLOOR	Beverages, Snacks	3.2	3 ratings	₹200 for two	2	FLAT ₹120 OFF\r\nUSE AXIS120ABOVE ₹500, FLAT ₹...	Ya
140654	Cafe Coffee Aani Barach Kahi	Snacks	3.2	50+ ratings	₹150 for two	2	FLAT ₹120 OFF\r\nUSE AXIS120ABOVE ₹500, FLAT ₹...	Ya L
140655	Patil Family Restaurant	North Indian, Biryani	4.3	9 ratings	₹200 for two	2	FLAT ₹120 OFF\r\nUSE AXIS120ABOVE ₹500, FLAT ₹...	Ya L
140656	Prabhakar Mama Cha Dhaba	North Indian	--	Too Few Ratings	₹350 for two	2	FLAT ₹120 OFF\r\nUSE AXIS120ABOVE ₹500, FLAT ₹...	Ya L

In [7]:
```
df.shape
```

Out[7]:
```
(140657, 10)
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 140657 entries, 0 to 140656
Data columns (total 10 columns):
 #   Column             Non-Null Count   Dtype 
---  ------             --------------   ----- 
 0   Restaurant Name    140657 non-null  object
 1   Cuisine            140630 non-null  object
 2   Rating             140657 non-null  object
 3   Number of Ratings  126115 non-null  object
 4   Average Price      140657 non-null  object
 5   Number of Offers   140657 non-null  int64 
 6   Offer Name         138849 non-null  object
 7   Area               140655 non-null  object
 8   Pure Veg           140657 non-null  object
 9   Location           140657 non-null  object
dtypes: int64(1), object(9)
memory usage: 10.7+ MB
```

In [9]:
```
df.describe()
```

Out[9]:

	Number of Offers
count	140657.000000
mean	3.187890
std	1.583943
min	0.000000
25%	1.000000
50%	4.000000
75%	5.000000
max	5.000000

In [10]:
```
df.isnull()
```

Out[10]:

	Restaurant Name	Cuisine	Rating	Number of Ratings	Average Price	Number of Offers	Offer Name	Area	Pure Veg
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
140652	False	False	False	False	False	False	False	False	False
140653	False	False	False	False	False	False	False	False	False
140654	False	False	False	False	False	False	False	False	False
140655	False	False	False	False	False	False	False	False	False
140656	False	False	False	False	False	False	False	False	False

140657 rows × 10 columns

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Restaurant Name          0
Cuisine                 27
Rating                   0
Number of Ratings    14542
Average Price            0
Number of Offers         0
Offer Name            1808
Area                     2
Pure Veg                 0
Location                 0
dtype: int64
```

In [12]:
```
##############################################################################
```

Automobile dataset operations

In [13]:
```
df1 = pd.read_csv('Automobile.csv')
```

In [14]:
```
df1.head()
```

Out[14]:

	name	mpg	cylinders	displacement	horsepower	weight	acceleration	model_y
0	chevrolet chevelle malibu	18.0	8	307.0	130.0	3504	12.0	
1	buick skylark 320	15.0	8	350.0	165.0	3693	11.5	
2	plymouth satellite	18.0	8	318.0	150.0	3436	11.0	
3	amc rebel sst	16.0	8	304.0	150.0	3433	12.0	
4	ford torino	17.0	8	302.0	140.0	3449	10.5	

In [15]:
```
df1.tail()
```

Out[15]:

	name	mpg	cylinders	displacement	horsepower	weight	acceleration	model
393	ford mustang gl	27.0	4	140.0	86.0	2790	15.6	
394	vw pickup	44.0	4	97.0	52.0	2130	24.6	
395	dodge rampage	32.0	4	135.0	84.0	2295	11.6	
396	ford ranger	28.0	4	120.0	79.0	2625	18.6	
397	chevy s-10	31.0	4	119.0	82.0	2720	19.4	

In [17]:
```
df1['origin']=='usa'
```

Out[17]:
```
0       True
1       True
2       True
3       True
4       True
       ...
393     True
394    False
395     True
396     True
397     True
Name: origin, Length: 398, dtype: bool
```

In [18]:
```
min(df1['weight'])
```

Out[18]:
```
1613
```

In [19]:
```
max(df1['weight'])
```

Out[19]:
```
5140
```

In [20]:
```
df1.isna()
```

Out[20]:

	name	mpg	cylinders	displacement	horsepower	weight	acceleration	model_ye
0	False	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	False	Fal
4	False	False	False	False	False	False	False	Fal
...	...	...	...	...	...	...	...	
393	False	False	False	False	False	False	False	Fal
394	False	False	False	False	False	False	False	Fal
395	False	False	False	False	False	False	False	Fal
396	False	False	False	False	False	False	False	Fal
397	False	False	False	False	False	False	False	Fal

398 rows × 9 columns

In [21]:
```
df1.value_counts()
```

Out[21]:
```
name                       mpg   cylinders  displacement  horsepower  weight  ac
celeration  model_year  origin
amc ambassador brougham    13.0  8          360.0         175.0       3821    1
1.0         73          usa       1
opel manta                 24.0  4          116.0         75.0        2158    1
5.5         73          europe    1
opel 1900                  25.0  4          116.0         81.0        2220    1
6.9         76          europe    1
oldsmobile vista cruiser   12.0  8          350.0         180.0       4499    1
2.5         73          usa       1
oldsmobile starfire sx     23.8  4          151.0         85.0        2855    1
7.6         78          usa       1

..
datsun b210 gx             39.4  4          85.0          70.0        2070    1
8.6         78          japan     1
datsun b210                31.0  4          79.0          67.0        1950    1
9.0         74          japan     1
datsun b-210               32.0  4          85.0          70.0        1990    1
7.0         76          japan     1
datsun 810 maxima          24.2  6          146.0         120.0       2930    1
3.8         81          japan     1
vw rabbit custom           31.9  4          89.0          71.0        1925    1
4.0         79          europe    1
Name: count, Length: 392, dtype: int64
```

b) Reshaping, Filtering, Scaling, Merging the data and Handling the missing values in datasets.

Merging

In [22]:
```
car_origin=pd.DataFrame()
car_weight=pd.DataFrame()
```

In [23]:
```
car_origin=pd.read_csv('Automobile.csv')
car_origin=car_origin[['origin','model_year']]
car_origin
```

Out[23]:

	origin	model_year
0	usa	70
1	usa	70
2	usa	70
3	usa	70
4	usa	70
...	...	...
393	usa	82
394	europe	82
395	usa	82
396	usa	82
397	usa	82

398 rows × 2 columns

In [25]:
```
car_weight=pd.read_csv('Automobile.csv')
car_weight=car_weight[['weight','model_year']]
car_weight
```

Out[25]:

	weight	model_year
0	3504	70
1	3693	70
2	3436	70
3	3433	70
4	3449	70
...	...	...
393	2790	82
394	2130	82
395	2295	82
396	2625	82
397	2720	82

398 rows × 2 columns

In [26]:
```
# Merging the dataframe
car_data=car_origin.merge(car_origin, how='inner',on='model_year')
car_data
```

Out[26]:

	origin_x	model_year	origin_y
0	usa	70	usa
1	usa	70	usa
2	usa	70	usa
3	usa	70	usa
4	usa	70	usa
...	...	...	...
12353	usa	82	usa
12354	usa	82	europe
12355	usa	82	usa
12356	usa	82	usa
12357	usa	82	usa

12358 rows × 3 columns

Handling Missing Values

In [27]:
```
df.isnull().sum()
```

Out[27]:
```
Restaurant Name          0
Cuisine                 27
Rating                   0
Number of Ratings    14542
Average Price            0
Number of Offers         0
Offer Name            1808
Area                     2
Pure Veg                 0
Location                 0
dtype: int64
```

In [38]:
```
from sklearn.impute import SimpleImputer
import numpy as np
```

In [39]:
```
imputer = SimpleImputer(missing_values=np.nan, strategy='mode')
```

In [ ]:
```
df_trans1 = imputer.fit_transform(df)
# Not able to Handle missing values as dataframe contans categorical data
```

Label Encoding

In [41]:
```
from sklearn.preprocessing import LabelEncoder
```

In [42]:
```
df1.head()
```

Out[42]:

	name	mpg	cylinders	displacement	horsepower	weight	acceleration	model_y
0	chevrolet chevelle malibu	18.0	8	307.0	130.0	3504	12.0	
1	buick skylark 320	15.0	8	350.0	165.0	3693	11.5	
2	plymouth satellite	18.0	8	318.0	150.0	3436	11.0	
3	amc rebel sst	16.0	8	304.0	150.0	3433	12.0	
4	ford torino	17.0	8	302.0	140.0	3449	10.5	

In [43]:
```
encoder = LabelEncoder()
```

In [45]:
```
new_car_data=pd.DataFrame()
new_car_data
```

Out[45]: —

In [47]:
```
new_car_data['weight'] = encoder.fit_transform(df1['weight'])
```

In [48]:
```
new_car_data
```

Out[48]:

	weight
0	247
1	265
2	241
3	240
4	244
...	...
393	160
394	54
395	89
396	136
397	152

398 rows × 1 columns

In [49]:
```
new_car_data['displacement'] = df1['displacement']
new_car_data['horsepower'] = df1['horsepower']
new_car_data['acceleration'] = encoder.fit_transform(df1['acceleration'])
new_car_data['cylinders'] = df1['cylinders']
```

In [50]:
```
new_car_data
```

Out[50]:

	weight	displacement	horsepower	acceleration	cylinders
0	247	307.0	130.0	13	8
1	265	350.0	165.0	11	8
2	241	318.0	150.0	6	8
3	240	304.0	150.0	13	8
4	244	302.0	140.0	5	8
...	...	...	...	...	...
393	160	140.0	86.0	43	4
394	54	97.0	52.0	93	4
395	89	135.0	84.0	12	4
396	136	120.0	79.0	71	4
397	152	119.0	82.0	76	4

398 rows × 5 columns

In [68]:
```
# Now all column data is in numner now we can handle missing data.
```

In [51]:
```
from sklearn.impute import SimpleImputer
import numpy as np
```

In [52]:
```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
```

In [55]:
```
car_NaN_handle = imputer.fit_transform(new_car_data)
```

In [56]:
```
car_NaN_handle
```

Out[56]:
```
array([[247., 307., 130.,  13.,   8.],
       [265., 350., 165.,  11.,   8.],
       [241., 318., 150.,   6.,   8.],
       ...,
       [ 89., 135.,  84.,  12.,   4.],
       [136., 120.,  79.,  71.,   4.],
       [152., 119.,  82.,  76.,   4.]])
```

In [57]:
```
car_NaN_handle=pd.DataFrame(car_NaN_handle)
```

In [58]:
```
car_NaN_handle
```

Out[58]:

	0	1	2	3	4
0	247.0	307.0	130.0	13.0	8.0
1	265.0	350.0	165.0	11.0	8.0
2	241.0	318.0	150.0	6.0	8.0
3	240.0	304.0	150.0	13.0	8.0
4	244.0	302.0	140.0	5.0	8.0
...	...	...	...	...	...
393	160.0	140.0	86.0	43.0	4.0
394	54.0	97.0	52.0	93.0	4.0
395	89.0	135.0	84.0	12.0	4.0
396	136.0	120.0	79.0	71.0	4.0
397	152.0	119.0	82.0	76.0	4.0

398 rows × 5 columns

In [59]:
```
car_NaN_handle.isnull().sum()
```

Out[59]:
```
0    0
1    0
2    0
3    0
4    0
dtype: int64
```

Creating Dependent and Indepent columns : X and Y

In [60]:
```
cardata=pd.DataFrame(car_NaN_handle)
```

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

Out[61]: (398, 5)

In [62]:
```
X = pd.DataFrame(cardata.iloc[:, 0:4].values)
#cardata.iloc[:, 0:4].values extracts the values from the selected rows and colu
```

In [63]:
```
X
```

Out[63]:

	0	1	2	3
0	247.0	307.0	130.0	13.0
1	265.0	350.0	165.0	11.0
2	241.0	318.0	150.0	6.0
3	240.0	304.0	150.0	13.0
4	244.0	302.0	140.0	5.0
...	...	...	...	...
393	160.0	140.0	86.0	43.0
394	54.0	97.0	52.0	93.0
395	89.0	135.0	84.0	12.0
396	136.0	120.0	79.0	71.0
397	152.0	119.0	82.0	76.0

398 rows × 4 columns

In [79]:
```
Y = pd.DataFrame(cardata.iloc[:, -1].values)
```

In [64]:
```
Y = pd.DataFrame(cardata.iloc[:, 4:].values)
```

In [65]:
```
Y
```

Out[65]:

	0
0	8.0
1	8.0
2	8.0
3	8.0
4	8.0
...	...
393	4.0
394	4.0
395	4.0
396	4.0
397	4.0

398 rows × 1 columns

Feature Scaling of DataSet- MinMaxScalar

In [66]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [67]:
```
scalar = MinMaxScaler()
```

In [68]:
```
X_scaled = pd.DataFrame(scalar.fit_transform(X))
```

In [69]:
```
X_scaled
```

Out[69]:

	0	1	2	3
0	0.705714	0.617571	0.456522	0.138298
1	0.757143	0.728682	0.646739	0.117021
2	0.688571	0.645995	0.565217	0.063830
3	0.685714	0.609819	0.565217	0.138298
4	0.697143	0.604651	0.510870	0.053191
...	...	...	...	...
393	0.457143	0.186047	0.217391	0.457447
394	0.154286	0.074935	0.032609	0.989362
395	0.254286	0.173127	0.206522	0.127660
396	0.388571	0.134367	0.179348	0.755319
397	0.434286	0.131783	0.195652	0.808511

398 rows × 4 columns

Train Test Split

In [70]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_
```

In [71]:
```
X_train
```

Out[71]:

	0	1	2	3
112	91.0	122.0	85.0	70.0
283	220.0	232.0	90.0	68.0
379	52.0	98.0	70.0	59.0
306	131.0	173.0	115.0	9.0
343	2.0	79.0	58.0	55.0
...	...	...	...	...
299	209.0	141.0	71.0	94.0
22	98.0	104.0	95.0	61.0
72	281.0	304.0	150.0	16.0
15	166.0	198.0	95.0	42.0
168	139.0	140.0	83.0	56.0

278 rows × 4 columns

In [72]:
```
X_train.shape
```

Out[72]:
```
(278, 4)
```

In [73]:
```
X_test
```

Out[73]:

	0	1	2	3
94	343.0	440.0	215.000000	6.0
32	38.0	98.0	104.469388	74.0
279	55.0	98.0	68.000000	52.0
178	184.0	120.0	88.000000	56.0
354	92.0	100.0	104.469388	45.0
...	...	...	...	...
253	205.0	200.0	95.000000	68.0
99	182.0	232.0	100.000000	47.0
217	59.0	111.0	80.000000	35.0
391	96.0	135.0	84.000000	20.0
10	252.0	383.0	170.000000	4.0

120 rows × 4 columns

In [74]:
```
X_test.shape
```

Out[74]:
```
(120, 4)
```

In [75]:
```
Y_train
```

Out[75]:

	0
112	4.0
283	6.0
379	4.0
306	6.0
343	4.0
...	...
299	4.0
22	4.0
72	8.0
15	6.0
168	4.0

278 rows × 1 columns

In [76]:
```
Y_train.shape
```

Out[76]: (278, 1)

In [77]:
```
Y_test
```

Out[77]:

	0
94	8.0
32	4.0
279	4.0
178	4.0
354	4.0
...	...
253	6.0
99	6.0
217	4.0
391	4.0
10	8.0

120 rows × 1 columns

In [78]:
```
Y_test.shape
```

Out[78]: (120, 1)