Pandas

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

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

mydataset = {"cars": ["BMW", "Volvo", "Ford"], "passings": [3, 7, 2]}

mydf = pd.DataFrame(mydataset)

print(mydf)
```

```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

In [2]:
```
mydf["passings"]
```

Out[2]:
```
0    3
1    7
2    2
Name: passings, dtype: int64
```

A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.

In [3]:
```
a = [1, 7, 2]
myvar = pd.Series(a)
print(myvar)
```

```
0    1
1    7
2    2
dtype: int64
```

In [4]:
```
# access the first element
print(myvar[0])
```

```
1
```

In [5]:
```
# with defined index
a = [1, 7, 2]
myvar = pd.Series(a, index=["x", "y", "z"])
print(myvar)
```

```
x    1
y    7
z    2
dtype: int64
```

In [ ]:
```
# access the element with index label
print(myvar["y"])
```

In [6]:
```
# create a series from a dictionary where the keys will be used as index
calories = {"day1": 420, "day2": 380, "day3": 390}
myvar = pd.Series(calories)
print(myvar)
```

```
day1    420
day2    380
day3    390
dtype: int64
```

In [7]:
```
# create a Series using only data from "day1" and "day2"
calories = {"day1": 420, "day2": 380, "day3": 390}
myvar = pd.Series(calories, index=["day1", "day2"])
print(myvar)
```

```
day1    420
day2    380
dtype: int64
```

In [10]:
```
data = {"calories": [420, 380, 390], "duration": [50, 40, 45]}
# df = pd.DataFrame(data)
df = pd.DataFrame(data, index=["day1", "day2", "day3"])
print(df)
```

```
      calories  duration
day1       420        50
day2       380        40
day3       390        45
```

In [12]:
```
# loc uses named index
# print(df.loc[0])
print(df.loc["day1"])
```

```
calories    420
duration     50
Name: day1, dtype: int64
```

In [13]:
```
# to include multiple rows
# print(df.loc[[0, 1]])
print(df.loc[["day1", "day2"]])
```

```
      calories  duration
day1       420        50
day2       380        40
```

In [14]:
```
# iloc for using indexes instead of labels
df = pd.DataFrame(data, index=["day1", "day2", "day3"])
print(df.iloc[0:2])
```

```
      calories  duration
day1       420        50
day2       380        40
```

In [15]:
```
import numpy as np


data = {"calories": [420, 380, 390], "duration": [50, 40, 45]}
df = pd.DataFrame(data)

selected_df1 = df.loc[[0, 1]]
selected_df2 = df.iloc[0:2]

print(
    f"The df and selected_df1 share memory using 'loc' method: {np.shares_memory(selec
)
print(
    f"The df and selected_df2 share memory using 'iloc' method: {np.shares_memory(sel
)
```

```
print(f"The selected_df1 is a view: {selected_df1._is_view}")
print(f"The selected_df2 is a view: {selected_df2._is_view}")
```

```
The df and selected_df1 share memory using 'loc' method: False
The df and selected_df2 share memory using 'iloc' method: True
The selected_df1 is a view: False
The selected_df2 is a view: True
```

In [17]:
```
# this behaviour is not guaranteed for mixed data types
data = {
    "calories": [420, 380, 390],
    "duration": ["50", "40", "45"],
}

df = pd.DataFrame(data)

selected_df1 = df.loc[[0, 1]]
selected_df2 = df.iloc[0:2]

print(
    f"The df and selected_df1 share memory using 'loc' method: {np.shares_memory(selec
)
print(
    f"The df and selected_df2 share memory using 'iloc' method: {np.shares_memory(sel
)
print(f"The selected_df1 is a view: {selected_df1._is_view}")
print(f"The selected_df2 is a view: {selected_df2._is_view}")
```

```
The df and selected_df1 share memory using 'loc' method: False
The df and selected_df2 share memory using 'iloc' method: False
The selected_df1 is a view: False
The selected_df2 is a view: False
```

In [18]:
```
# sample data with nan values
df = pd.read_csv(
    "https://github.com/atsfc/jupyterlite/raw/refs/heads/main/content/CompMath/sample_
)
df
```

Out[18]:

	name	age	state	point	other
0	Alice	24.0	NY	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	Charlie	NaN	CA	NaN	NaN
3	Dave	68.0	TX	70.0	NaN
4	Ellen	NaN	CA	88.0	NaN
5	Frank	30.0	NaN	NaN	NaN

In [19]:
```
# check for missing values
print(df.isnull())
```

```
    name    age  state  point  other
0  False  False  False   True   True
1   True   True   True   True   True
2  False   True  False   True   True
3  False  False  False  False   True
4  False   True  False  False   True
5  False  False   True   True   True
```

In [20]:
```
# check for the number of missing values
np.sum(df.isnull(), axis=0)
```

Out[20]:
```
name     1
age      3
state    2
point    4
other    6
dtype: int64
```

In [21]:
```
df.isnull().sum(axis=0)
```

Out[21]:
```
name     1
age      3
state    2
point    4
other    6
dtype: int64
```

In [22]:
```
print(df["point"].isnull())
```

```
0     True
1     True
2     True
3    False
4    False
5     True
Name: point, dtype: bool
```

In [23]:
```
np.sum(df["point"].isnull())
```

Out[23]:
```
np.int64(4)
```

In [24]:
```
new_df = df.dropna()
new_df
```

Out[24]:

	name	age	state	point	other

In [28]:
```
df[["name", "age", "state", "point"]].dropna()
```

Out[28]:

	name	age	state	point
3	Dave	68.0	TX	70.0

In [29]:
```
df
```

Out[29]:

	name	age	state	point	other
0	Alice	24.0	NY	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	Charlie	NaN	CA	NaN	NaN
3	Dave	68.0	TX	70.0	NaN
4	Ellen	NaN	CA	88.0	NaN
5	Frank	30.0	NaN	NaN	NaN

In [31]:
```
# inplace=True will change the original dataframe
df.dropna(inplace=True)
df
```

Out[31]:

	name	age	state	point	other

In [32]:
```
df = pd.read_csv(
```

```
    "https://github.com/atsfc/jupyterlite/raw/refs/heads/main/content/CompMath/sample_
)
df
```

Out[32]:

	name	age	state	point	other
0	Alice	24.0	NY	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	Charlie	NaN	CA	NaN	NaN
3	Dave	68.0	TX	70.0	NaN
4	Ellen	NaN	CA	88.0	NaN
5	Frank	30.0	NaN	NaN	NaN

In [33]:

```
# fill all missing values with 0
df.fillna(0, inplace=True)
df
```

Out[33]:

	name	age	state	point	other
0	Alice	24.0	NY	0.0	0.0
1	0	0.0	0	0.0	0.0
2	Charlie	0.0	CA	0.0	0.0
3	Dave	68.0	TX	70.0	0.0
4	Ellen	0.0	CA	88.0	0.0
5	Frank	30.0	0	0.0	0.0

In [34]:

```
# fill missing values specifically for each column
df = pd.read_csv(
    "https://github.com/atsfc/jupyterlite/raw/refs/heads/main/content/CompMath/sample_
)
df.fillna({"name": "No Name", "point": 0.0, "age": 30.0, "state": "DC"}, inplace=True
newdf = df[["name", "point", "age", "state"]].copy()
newdf
```

Out[34]:

	name	point	age	state
0	Alice	0.0	24.0	NY
1	No Name	0.0	30.0	DC
2	Charlie	0.0	30.0	CA
3	Dave	70.0	68.0	TX
4	Ellen	88.0	30.0	CA
5	Frank	0.0	30.0	DC

In [35]:

```
# load data from url
url = "https://github.com/YBI-Foundation/Dataset/raw/refs/heads/main/Diabetes%20Missi
df = pd.read_csv(url)
df
```

Out[35]:

	Pregnant	Glucose	Diastolic_BP	Skin_Fold	Serum_Insulin	BMI	Diabetes_Pedigree	Age	Class
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.0	32.9	0.171	63	0
764	2	122.0	70.0	27.0	NaN	36.8	0.340	27	0
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0
766	1	126.0	60.0	NaN	NaN	30.1	0.349	47	1
767	1	93.0	70.0	31.0	NaN	30.4	0.315	23	0

768 rows × 9 columns

In [36]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Pregnant           768 non-null    int64  
 1   Glucose            763 non-null    float64
 2   Diastolic_BP       733 non-null    float64
 3   Skin_Fold          541 non-null    float64
 4   Serum_Insulin      394 non-null    float64
 5   BMI                757 non-null    float64
 6   Diabetes_Pedigree  768 non-null    float64
 7   Age                768 non-null    int64  
 8   Class              768 non-null    int64  
dtypes: float64(6), int64(3)
memory usage: 54.1 KB
```

In [37]:
```
df.isnull()
```

Out[37]:

	Pregnant	Glucose	Diastolic_BP	Skin_Fold	Serum_Insulin	BMI	Diabetes_Pedigree	Age	Class
0	False	False	False	False	True	False	False	False	False
1	False	False	False	False	True	False	False	False	False
2	False	False	False	True	True	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
763	False	False	False	False	False	False	False	False	False
764	False	False	False	False	True	False	False	False	False
765	False	False	False	False	False	False	False	False	False
766	False	False	False	True	True	False	False	False	False
767	False	False	False	False	True	False	False	False	False

768 rows × 9 columns

In [38]:
```
df.isnull().sum(axis=0)
```

Out[38]:
```
Pregnant              0
Glucose               5
Diastolic_BP         35
Skin_Fold           227
Serum_Insulin       374
BMI                  11
Diabetes_Pedigree     0
Age                   0
Class                 0
dtype: int64
```

In [39]:
```
df.dropna()
```

Out[39]:

	Pregnant	Glucose	Diastolic_BP	Skin_Fold	Serum_Insulin	BMI	Diabetes_Pedigree	Age	Class
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53	1
13	1	189.0	60.0	23.0	846.0	30.1	0.398	59	1
...	...	...	...	...	...	...	...	...	...
753	0	181.0	88.0	44.0	510.0	43.3	0.222	26	1
755	1	128.0	88.0	39.0	110.0	36.5	1.057	37	1
760	2	88.0	58.0	26.0	16.0	28.4	0.766	22	0
763	10	101.0	76.0	48.0	180.0	32.9	0.171	63	0
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0

392 rows × 9 columns

In [40]:
```
# fill missing values with mean
SI_mean = df["Serum_Insulin"].mean()
df.fillna({"Serum_Insulin": SI_mean}, inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Pregnant           768 non-null    int64  
 1   Glucose            763 non-null    float64
 2   Diastolic_BP       733 non-null    float64
 3   Skin_Fold          541 non-null    float64
 4   Serum_Insulin      768 non-null    float64
 5   BMI                757 non-null    float64
 6   Diabetes_Pedigree  768 non-null    float64
 7   Age                768 non-null    int64  
 8   Class              768 non-null    int64  
dtypes: float64(6), int64(3)
memory usage: 54.1 KB
```

In [41]:
```
# fill missing values with median
SF_median = df["Skin_Fold"].median()
df.fillna({"Skin_Fold": SF_median}, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Pregnant           768 non-null    int64  
 1   Glucose            763 non-null    float64
 2   Diastolic_BP       733 non-null    float64
 3   Skin_Fold          768 non-null    float64
 4   Serum_Insulin      768 non-null    float64
 5   BMI                757 non-null    float64
 6   Diabetes_Pedigree  768 non-null    float64
 7   Age                768 non-null    int64  
 8   Class              768 non-null    int64  
dtypes: float64(6), int64(3)
memory usage: 54.1 KB
```

In [42]:
```
# fill missing values with mode
# mode is the value that appears most frequently
BMI_mode = df["BMI"].mode()[0]
print(BMI_mode)
df.fillna({"BMI": BMI_mode}, inplace=True)
df.info()
```

```
32.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Pregnant           768 non-null    int64  
 1   Glucose            763 non-null    float64
 2   Diastolic_BP       733 non-null    float64
 3   Skin_Fold          768 non-null    float64
 4   Serum_Insulin      768 non-null    float64
 5   BMI                768 non-null    float64
 6   Diabetes_Pedigree  768 non-null    float64
 7   Age                768 non-null    int64  
 8   Class              768 non-null    int64  
dtypes: float64(6), int64(3)
memory usage: 54.1 KB
```

In [43]:
```
df
```

Out[43]:

	Pregnant	Glucose	Diastolic_BP	Skin_Fold	Serum_Insulin	BMI	Diabetes_Pedigree	Age	Class
0	6	148.0	72.0	35.0	155.548223	33.6	0.627	50	1
1	1	85.0	66.0	29.0	155.548223	26.6	0.351	31	0
2	8	183.0	64.0	29.0	155.548223	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.000000	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.000000	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.000000	32.9	0.171	63	0
764	2	122.0	70.0	27.0	155.548223	36.8	0.340	27	0
765	5	121.0	72.0	23.0	112.000000	26.2	0.245	30	0
766	1	126.0	60.0	29.0	155.548223	30.1	0.349	47	1
767	1	93.0	70.0	31.0	155.548223	30.4	0.315	23	0

768 rows × 9 columns

In [44]:
```
# you can see all mode values
df.mode()
```

Out[44]:

	Pregnant	Glucose	Diastolic_BP	Skin_Fold	Serum_Insulin	BMI	Diabetes_Pedigree	Age	Class
0	1.0	99.0	70.0	29.0	155.548223	32.0	0.254	22.0	0.0
1	NaN	100.0	NaN	NaN	NaN	NaN	0.258	NaN	NaN

In [45]:
```
df["Diabetes_Pedigree"].loc[df["Diabetes_Pedigree"] > 1].info()
```

```
<class 'pandas.core.series.Series'>
Index: 51 entries, 4 to 755
Series name: Diabetes_Pedigree
Non-Null Count  Dtype
--------------  -----
51 non-null     float64
dtypes: float64(1)
memory usage: 816.0 bytes
```

In [46]:
```
# replace all values in "Diabetes_Pedigree" column that are greater than 1 with 1
df.loc[df["Diabetes_Pedigree"] > 1, "Diabetes_Pedigree"] = 1
df
```

Out[46]:

	Pregnant	Glucose	Diastolic_BP	Skin_Fold	Serum_Insulin	BMI	Diabetes_Pedigree	Age	Class
0	6	148.0	72.0	35.0	155.548223	33.6	0.627	50	1
1	1	85.0	66.0	29.0	155.548223	26.6	0.351	31	0
2	8	183.0	64.0	29.0	155.548223	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.000000	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.000000	43.1	1.000	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.000000	32.9	0.171	63	0
764	2	122.0	70.0	27.0	155.548223	36.8	0.340	27	0
765	5	121.0	72.0	23.0	112.000000	26.2	0.245	30	0
766	1	126.0	60.0	29.0	155.548223	30.1	0.349	47	1
767	1	93.0	70.0	31.0	155.548223	30.4	0.315	23	0

768 rows × 9 columns

In [47]:
```
# duplicate a row
newdf = pd.concat([df, df.iloc[0:1]], ignore_index=True)
newdf
```

Out[47]:

	Pregnant	Glucose	Diastolic_BP	Skin_Fold	Serum_Insulin	BMI	Diabetes_Pedigree	Age	Class
0	6	148.0	72.0	35.0	155.548223	33.6	0.627	50	1
1	1	85.0	66.0	29.0	155.548223	26.6	0.351	31	0
2	8	183.0	64.0	29.0	155.548223	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.000000	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.000000	43.1	1.000	33	1
...	...	...	...	...	...	...	...	...	...
764	2	122.0	70.0	27.0	155.548223	36.8	0.340	27	0
765	5	121.0	72.0	23.0	112.000000	26.2	0.245	30	0
766	1	126.0	60.0	29.0	155.548223	30.1	0.349	47	1
767	1	93.0	70.0	31.0	155.548223	30.4	0.315	23	0
768	6	148.0	72.0	35.0	155.548223	33.6	0.627	50	1

769 rows × 9 columns

In [49]:
```
# to check for duplicates
# duplicated shows the duplicate rows only
# newdf.duplicated()
np.sum(newdf.duplicated())
```

Out[49]: np.int64(1)

In [50]:
```
# load data from url
url = "https://github.com/Opensourcefordatascience/Data-sets/raw/refs/heads/master/au
df = pd.read_csv(url)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          205 non-null    int64  
 1   normalized-losses  205 non-null    object 
 2   make               205 non-null    object 
 3   fuel-type          205 non-null    object 
 4   aspiration         205 non-null    object 
 5   num-of-doors       205 non-null    object 
 6   body-style         205 non-null    object 
 7   drive-wheels       205 non-null    object 
 8   engine-location    205 non-null    object 
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64  
 14  engine-type        205 non-null    object 
 15  num-of-cylinders   205 non-null    object 
 16  engine-size        205 non-null    int64  
 17  fuel-system        205 non-null    object 
 18  bore               205 non-null    object 
 19  stroke             205 non-null    object 
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    object 
 22  peak-rpm           205 non-null    object 
 23  city-mpg           205 non-null    int64  
 24  highway-mpg        205 non-null    int64  
 25  price              205 non-null    object 
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
```

In [51]:
```
np.sum(df.duplicated())
```

Out[51]:
```
np.int64(0)
```

In [56]:
```
# number of unique values in each column
df.nunique()
```

Out[56]:
```
symboling             6
normalized-losses    52
make                 22
fuel-type             2
aspiration            2
num-of-doors          3
body-style            5
drive-wheels          3
engine-location       2
wheel-base           53
length               75
width                44
height               49
curb-weight         171
engine-type           7
num-of-cylinders      7
engine-size          44
fuel-system           8
bore                 39
stroke               37
compression-ratio    32
horsepower           60
peak-rpm             24
city-mpg             29
highway-mpg          30
price               187
dtype: int64
```

In [57]:
```
# unique values in a column
df["price"].unique()
```

Out[57]:
```
array(['13495', '16500', '13950', '17450', '15250', '17710', '18920',
       '23875', '?', '16430', '16925', '20970', '21105', '24565', '30760',
       '41315', '36880', '5151', '6295', '6575', '5572', '6377', '7957',
       '6229', '6692', '7609', '8558', '8921', '12964', '6479', '6855',
       '5399', '6529', '7129', '7295', '7895', '9095', '8845', '10295',
       '12945', '10345', '6785', '11048', '32250', '35550', '36000',
       '5195', '6095', '6795', '6695', '7395', '10945', '11845', '13645',
       '15645', '8495', '10595', '10245', '10795', '11245', '18280',
       '18344', '25552', '28248', '28176', '31600', '34184', '35056',
       '40960', '45400', '16503', '5389', '6189', '6669', '7689', '9959',
       '8499', '12629', '14869', '14489', '6989', '8189', '9279', '5499',
       '7099', '6649', '6849', '7349', '7299', '7799', '7499', '7999',
       '8249', '8949', '9549', '13499', '14399', '17199', '19699',
       '18399', '11900', '13200', '12440', '13860', '15580', '16900',
       '16695', '17075', '16630', '17950', '18150', '12764', '22018',
       '32528', '34028', '37028', '9295', '9895', '11850', '12170',
       '15040', '15510', '18620', '5118', '7053', '7603', '7126', '7775',
       '9960', '9233', '11259', '7463', '10198', '8013', '11694', '5348',
       '6338', '6488', '6918', '7898', '8778', '6938', '7198', '7788',
       '7738', '8358', '9258', '8058', '8238', '9298', '9538', '8449',
       '9639', '9989', '11199', '11549', '17669', '8948', '10698', '9988',
       '10898', '11248', '16558', '15998', '15690', '15750', '7975',
       '7995', '8195', '9495', '9995', '11595', '9980', '13295', '13845',
       '12290', '12940', '13415', '15985', '16515', '18420', '18950',
       '16845', '19045', '21485', '22470', '22625'], dtype=object)
```

In [58]:
```
# replace all "?" with "NaN"
df.replace("?", np.nan, inplace=True)
df.dropna(inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 159 entries, 3 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          159 non-null    int64
 1   normalized-losses  159 non-null    object
 2   make               159 non-null    object
 3   fuel-type          159 non-null    object
 4   aspiration         159 non-null    object
 5   num-of-doors       159 non-null    object
 6   body-style         159 non-null    object
 7   drive-wheels       159 non-null    object
 8   engine-location    159 non-null    object
 9   wheel-base         159 non-null    float64
 10  length             159 non-null    float64
 11  width              159 non-null    float64
 12  height             159 non-null    float64
 13  curb-weight        159 non-null    int64
 14  engine-type        159 non-null    object
 15  num-of-cylinders   159 non-null    object
 16  engine-size        159 non-null    int64
 17  fuel-system        159 non-null    object
 18  bore               159 non-null    object
 19  stroke             159 non-null    object
 20  compression-ratio  159 non-null    float64
 21  horsepower         159 non-null    object
 22  peak-rpm           159 non-null    object
 23  city-mpg           159 non-null    int64
 24  highway-mpg        159 non-null    int64
 25  price              159 non-null    object
dtypes: float64(5), int64(5), object(16)
memory usage: 33.5+ KB
```

In [59]:
```
df
```

Out[59]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	eng
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	
6	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	
8	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	
10	2	192	bmw	gas	std	two	sedan	rwd	front	101.2	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	

159 rows × 26 columns

In [60]:
```
# convert the problematic columns to numeric
df["price"] = pd.to_numeric(df["price"])
df["horsepower"] = pd.to_numeric(df["horsepower"])
df["peak-rpm"] = pd.to_numeric(df["peak-rpm"])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 159 entries, 3 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          159 non-null    int64  
 1   normalized-losses  159 non-null    object 
 2   make               159 non-null    object 
 3   fuel-type          159 non-null    object 
 4   aspiration         159 non-null    object 
 5   num-of-doors       159 non-null    object 
 6   body-style         159 non-null    object 
 7   drive-wheels       159 non-null    object 
 8   engine-location    159 non-null    object 
 9   wheel-base         159 non-null    float64
 10  length             159 non-null    float64
 11  width              159 non-null    float64
 12  height             159 non-null    float64
 13  curb-weight        159 non-null    int64  
 14  engine-type        159 non-null    object 
 15  num-of-cylinders   159 non-null    object 
 16  engine-size        159 non-null    int64  
 17  fuel-system        159 non-null    object 
 18  bore               159 non-null    object 
 19  stroke             159 non-null    object 
 20  compression-ratio  159 non-null    float64
 21  horsepower         159 non-null    int64  
 22  peak-rpm           159 non-null    int64  
 23  city-mpg           159 non-null    int64  
 24  highway-mpg        159 non-null    int64  
 25  price              159 non-null    int64  
dtypes: float64(5), int64(8), object(13)
memory usage: 33.5+ KB
```

In [61]:
```
# if you need to discard non-numeric columns
newdf = df._get_numeric_data()
newdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 159 entries, 3 to 204
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          159 non-null    int64  
 1   wheel-base         159 non-null    float64
 2   length             159 non-null    float64
 3   width              159 non-null    float64
 4   height             159 non-null    float64
 5   curb-weight        159 non-null    int64  
 6   engine-size        159 non-null    int64  
 7   compression-ratio  159 non-null    float64
 8   horsepower         159 non-null    int64  
 9   peak-rpm           159 non-null    int64  
 10  city-mpg           159 non-null    int64  
 11  highway-mpg        159 non-null    int64  
 12  price              159 non-null    int64  
dtypes: float64(5), int64(8)
memory usage: 17.4 KB
```

In [66]:
```
newdf.corr().style.background_gradient(cmap="coolwarm")
```

Out[66]:

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio
symboling	1.000000	-0.520591	-0.336257	-0.219186	-0.475185	-0.251880	-0.109453	-0.138316
wheel-base	-0.520591	1.000000	0.871534	0.814991	0.555767	0.810181	0.649206	0.291431
length	-0.336257	0.871534	1.000000	0.838338	0.499251	0.871291	0.725953	0.184814
width	-0.219186	0.814991	0.838338	1.000000	0.292706	0.870595	0.779253	0.258752
height	-0.475185	0.555767	0.499251	0.292706	1.000000	0.367052	0.111083	0.233308
curb-weight	-0.251880	0.810181	0.871291	0.870595	0.367052	1.000000	0.888626	0.224724
engine-size	-0.109453	0.649206	0.725953	0.779253	0.111083	0.888626	1.000000	0.141097
compression-ratio	-0.138316	0.291431	0.184814	0.258752	0.233308	0.224724	0.141097	1.000000
horsepower	-0.003949	0.516948	0.672063	0.681872	0.034317	0.790095	0.812073	-0.162305
peak-rpm	0.199106	-0.289234	-0.234074	-0.232216	-0.245864	-0.259988	-0.284686	-0.416769
city-mpg	0.089550	-0.580657	-0.724544	-0.666684	-0.199737	-0.762155	-0.699139	0.278332
highway-mpg	0.149830	-0.611750	-0.724599	-0.693339	-0.226136	-0.789338	-0.714095	0.221483
price	-0.162794	0.734419	0.760952	0.843371	0.244836	0.893639	0.841496	0.209361

The Result of the `corr()` method is a table with a lot of numbers that represents how well the relationship is between two columns.

- The number varies from -1 to 1.
- 1 means that there is a 1 to 1 relationship (a perfect correlation), and for this data set, each time a value went up in the first column, the other one went up as well.
- 0.9 is also a good relationship, and if you increase one value, the other will probably increase as well.
- -0.9 would be just as good relationship as 0.9, but if you increase one value, the other will probably go down.
- 0.2 means NOT a good relationship, meaning that if one value goes up does not mean that the other will.