# Pandas를 활용한 UCI 자동차 연비 데이터 분석

## ● pandas DataFrame으로 'auto-mpg' 파일 가져와서 분석¶

## 1) csv 파일을 활용하여 분석하기

csv 파일을 pandas는 read_csv로 읽습니다.
옵션 중에서 header는 레이블을 의미하며,
첫번째 행(row)이 레이블명이 아닌 데이터일 경우 None를 사용합니다.
만약 첫번째 행(row)이 레이블명이면 header=0,
두번째 행(row)이 레이블이면 header=1 이런식으로 사용합니다.

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



file_path = 'D:/AI/auto-mpg.csv'

# csv 파일의 구분자가 탭일 경우는  sep = "₩t" 옵션을 사용
df_csv = pd.read_csv(file_path, sep = "₩t", header=None)
```

판다스에서 제공하는 함수 중에서 head()는 읽어들인 데이터 파일 중 처음 5개 0~4 행(row)만 출력합니다.
데이터가 많을 경우 사용하면 편리합니다.

In [2]:
```
df_csv.head()
```

Out[2]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **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 |

데이터 파일 중 처음 시작하는 부분 중 n개를 출력하고 싶으면 df_csv.head(n)과 같은 형식으로 사용하면 됩니다.

In [3]:
```
# 데이터 파일 중 10개의 row (0~9 라인)을 출력합니다.
df_csv.head(10)
```

Out[3]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **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 |
| **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 |

iloc[x, y]은 integer-location[x,y]를 의미하며, x,y는 [row, column]인덱스 위치를 의미한다.

In [4]:
```
df_csv.iloc[[2, 5], 5:]
```

Out[4]:

| | 5 | 6 | 7 | 8 |
|---|---|---|---|---|
| **2** | 11.0 | 70 | 1 | plymouth satellite |
| **5** | 10.0 | 70 | 1 | ford galaxie 500 |

In [5]:
```
df_csv.iloc[[2, 5], [1, 8]]
```

Out[5]:

| | 1 | 8 |
|---|---|---|
| **2** | 8 | plymouth satellite |
| **5** | 8 | ford galaxie 500 |

판다스에서 제공하는 함수 중에서 tail()은
읽어들인 데이터 파일 중 마지막 5개 0~4 행(row)만 출력합니다. 데이터가 많을 경우 사용하면 편리합니다.

In [6]:
```
1 df_csv.tail()
```

Out[6]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **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 |

In [7]:
```
1 #df_csv  # 데이터 파일 모두를 출력합니다.
```

In [8]:
```
1 df_csv.columns    # 맨 첫 행에 있는 column부분... 레이블명에 해당할 부분을 출력합니다.
```

Out[8]:
```
Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8], dtype='int64')
```

In [9]:
```
1 df_csv.index
```

Out[9]:
```
RangeIndex(start=0, stop=398, step=1)
```

In [10]:
```
1 df_csv.shape  # 데이터 파일 중 row, column 구조를 출력합니다.
```

Out[10]:
```
(398, 9)
```

In [11]:
```
1 type(df_csv.shape)
```

Out[11]:
```
tuple
```

리스트는 [ ], 튜플은 ( )을 사용합니다.
리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 그 값을 바꿀 수 없습니다.
얼핏 보면 튜플과 리스트는 비슷한 역할을 하지만 프로그래밍을 할 때 튜플과 리스트는 구별해서 사용하는 것이 좋습니다.
즉 리스트의 항목 값은 변화가 가능하고 튜플의 항목 값은 변화가 불가능합니다.

따라서 프로그램이 실행되는 동안 그 값이 항상 변하지 않기를 바란다거나 값이 바뀔까 걱정하고 싶지 않다면
주저하지 말고 튜플을 사용해야 합니다. 이와는 반대로 수시로 그 값을 변화시켜야할 경우라면 리스트를 사용해야 합니다.

# 2) excel 파일을 활용하여 분석하기

In [12]:
```
# 엑셀 파일도 csv와 같은 형식으로 불러오면 됩니다.

file_path = 'D:/AI/auto-mpg.xlsx'

df_excel = pd.read_excel(file_path, 'Sheet1', header=None)
```

In [13]:
```
# 데이터 파일 중 5개의 row (0~4 라인)만 출력합니다. 데이터가 많을 경우 사용하면 편리합니다.
df_excel.head()
```

Out[13]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **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 |

In [14]:
```
# 데이터 파일 중 끝부분 5개의 라인만 출력합니다. 데이터가 많을 경우 사용하면 편리합니다.
df_excel.tail()
```

Out[14]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **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 |

In [15]:
```
df_excel.shape  # 데이터 파일 중 row, column 구조를 출력합니다.
```

Out[15]: (398, 9)

In [16]:
```
df_excel.columns    # 맨 처음 라인에 있는 column부분... 레이블명에 해당할 부분을 출력합니다.
```

Out[16]:
```
Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8], dtype='int64')
```

In [17]:
```
df_excel.index
```

Out[17]:
```
RangeIndex(start=0, stop=398, step=1)
```

In [18]:
```
df_excel[8].head()

# df_excel[n].head()  n은 레이블명에 해당하며, 현재 레이블명은 0,1,2,3,4,5,6,7,8이므로
# 8이라는 레이블명에 해당하는 데이터의 head 부분 5개를 출력하게 됩니다.
```

Out[18]:
```
0    chevrolet chevelle malibu
1            buick skylark 320
2           plymouth satellite
3                amc rebel sst
4                  ford torino
Name: 8, dtype: object
```

In [19]:
```
#df_excel[8]  # 8이라는 레이블명에 해당하는 데이터를 모두 출력하게 됩니다.
```

In [20]:
```
# df_excel[n:m]  n 부터 m-1 번째에 해당하는 row 데이터값을 출력합니다. 레이블명이 아니라 row 인덱스 입니다.
df_excel[8:10]
```

Out[20]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **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 |

In [21]:
```
# df_excel[n:m]  n 부터 m-1 번째에 해당하는 row 데이터값을 출력합니다. 레이블명이 아니라 row 인덱스 입니다.
df_excel[20:25]
```

Out[21]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **20** | 25.0 | 4 | 110.0 | 87 | 2672 | 17.5 | 70 | 2 | peugeot 504 |
| **21** | 24.0 | 4 | 107.0 | 90 | 2430 | 14.5 | 70 | 2 | audi 100 ls |
| **22** | 25.0 | 4 | 104.0 | 95 | 2375 | 17.5 | 70 | 2 | saab 99e |
| **23** | 26.0 | 4 | 121.0 | 113 | 2234 | 12.5 | 70 | 2 | bmw 2002 |
| **24** | 21.0 | 6 | 199.0 | 90 | 2648 | 15.0 | 70 | 1 | amc gremlin |

In [22]:
```
df_excel.iloc[3:5, 5:9]

# df_excel[n:m, i:j]  # n 부터 m-1 번째에 해당하는 row 데이터값 그리고 i 부터 j-1 번째에 해당하는 column을 출력합니다.
```

Out[22]:

| | 5 | 6 | 7 | 8 |
|---|---|---|---|---|
| **3** | 12.0 | 70 | 1 | amc rebel sst |
| **4** | 10.5 | 70 | 1 | ford torino |

### excel로 읽어들인 데이터 프레임에 열 레이블명 지정

In [23]:
```
# column 레이블명을 지정할 수 있습니다.

df_excel.columns = ['연비','실린더수','배기량','출력','차중', '가속능력','출시년도','제조국','모델명']

#df_excel.columns = ['mpg','cylinders','displacement','horsepower','weight','acceleration','model year','origin','name']

```

In [24]:
```
df_excel.columns  # 맨 처음 라인에 있는 column부분... 레이블명에 해당할 부분을 출력합니다.
```

Out[24]:
```
Index(['연비', '실린더수', '배기량', '출력', '차중', '가속능력', '출시년도', '제조국', '모델명'], dtype='object')
```

In [25]:
```
type(df_excel.columns)
```

Out[25]:
```
pandas.core.indexes.base.Index
```

In [26]:
```
df_excel.index
```

Out[26]:
```
RangeIndex(start=0, stop=398, step=1)
```

In [27]:
```
# 데이터프레임 df의 내용을 일부 확인
print("● df_excel 데이터프레임의 행 :",len(df_excel))
print(type(df_excel))

df_excel.head()     # column ... 레이블명까지 확인할 수 있습니다.
```

```
● df_excel 데이터프레임의 행 : 398
<class 'pandas.core.frame.DataFrame'>
```

Out[27]:

| | 연비 | 실린더수 | 배기량 | 출력 | 차중 | 가속능력 | 출시년도 | 제조국 | 모델명 |
|---|---|---|---|---|---|---|---|---|---|
| **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 |

In [28]:
```
df_excel.tail()    # column 레이블명까지 확인할 수 있습니다.
```

Out[28]:

| | 연비 | 실린더수 | 배기량 | 출력 | 차중 | 가속능력 | 출시년도 | 제조국 | 모델명 |
|---|---|---|---|---|---|---|---|---|---|
| **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 |

In [29]:
```
1 df_excel['모델명'].head()  # '모델명'이라는 레이블명에 해당하는 데이터를 출력하게 됩니다.
```

Out[29]:
```
0    chevrolet chevelle malibu
1            buick skylark 320
2           plymouth satellite
3                amc rebel sst
4                  ford torino
Name: 모델명, dtype: object
```

In [30]:
```
1 df_excel[8]  # 이제 8이라는 레이블명이 없으므로 에러가 출력됩니다.
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~Wanaconda3WLibWsite-packagesWpandasWcoreWindexesWbase.py:3802, in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File ~Wanaconda3WLibWsite-packagesWpandasW_libsWindex.pyx:138, in pandas._libs.index.IndexEngine.get_loc()

File ~Wanaconda3WLibWsite-packagesWpandasW_libsWindex.pyx:165, in pandas._libs.index.IndexEngine.get_loc()

File pandasW_libsWhashtable_class_helper.pxi:5745, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandasW_libsWhashtable_class_helper.pxi:5753, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 8

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

KeyError                                  Traceback (most recent call last)
Cell In[30], line 1
----> 1 df_excel[8]

File ~Wanaconda3WLibWsite-packagesWpandasWcoreWframe.py:3807, in DataFrame.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File ~Wanaconda3WLibWsite-packagesWpandasWcoreWindexesWbase.py:3804, in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 8
```

In [31]:
```
1 df_excel.info()  # 데이터프레임 df_excel에 관한 정보를 출력합니다.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   연비      398 non-null    float64
 1   실린더수    398 non-null    int64
 2   배기량     398 non-null    float64
 3   출력      398 non-null    object
 4   차중      398 non-null    int64
 5   가속능력    398 non-null    float64
 6   출시년도    398 non-null    int64
 7   제조국     398 non-null    int64
 8   모델명     398 non-null    object
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [32]:
```
1 df_excel.dtypes  # 데이터프레임 df_excel에 사용된 자료형을 출력합니다.
```

Out[32]:
```
연비      float64
실린더수      int64
배기량     float64
출력       object
차중        int64
가속능력    float64
출시년도      int64
제조국       int64
모델명      object
dtype: object
```

df_excel.describe()

데이터프레임 df_excel의 산술(숫자)데이터를 갖는 column에 대해서 기술통계 정보 출력합니다.

count 개수, mean 평균, std 표준편차, min 최소값, max 최대값을 의미하며
25%, 50%, 75%는 4분위수(25%, 50%, 75%)를 의미합니다.

텍스트가 포함되어 있는 '출력', '모델명' 부분은 생략되어 있습니다.
생략된 이유는 파일에서 '출력'의 경우 중간에 숫자가 아닌 ?가 포함되어 있기 때문입니다.

In [33]:
```
1 df_excel.describe()
```

Out[33]:

| | 연비 | 실린더수 | 배기량 | 차중 | 가속능력 | 출시년도 | 제조국 |
|---|---|---|---|---|---|---|---|
| **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 [34]:
```
1 type(df_excel)
```

Out[34]:
```
pandas.core.frame.DataFrame
```

아래와 같이 df_excel.describe(include='all')를 실행했을 때

● unique는 유일한 데이터의 개수,

● top은 가장 많은 빈도수를 갖는 변수
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html)
If multiple object values have the highest count, then the count and top results will be arbitrarily chosen from among those with the highest count.
top결과는 개수가 가장 높은 값이 여러개일 경우, 여러개 중에서 임의로 선택됩니다.

● freq는 top에 해당하는 변수의 빈도수를 의미합니다.

● 수치형 데이터의 요약값과 object 타입의 요약값이 다른데 이를 하나의 테이블에 출력하다보니
'출력'과 '모델명'을 제외한 나머지 object타입의 경우
unique, top, freq와 같은 부분에 NaN(Not a Number)으로 출력됩니다.

● 하지만 파일에서 '출력'에는 ? 그리고 '모델명'에는 텍스트가 포함되어 있으므로
unique, top, freq와 같은 부분에 값이 출력되었습니다.
대신 mean부터 max까지 수치형 데이터 요약값 부분에는 NaN(Not a Number)이 출력됩니다.

In [35]:
```
df_excel.describe(include='all')
```

Out[35]:

| | 연비 | 실린더수 | 배기량 | 출력 | 차중 | 가속능력 | 출시년도 | 제조국 | 모델명 |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 398.000000 | 398.000000 | 398.000000 | 398.0 | 398.000000 | 398.000000 | 398.000000 | 398.000000 | 398 |
| **unique** | NaN | NaN | NaN | 94.0 | NaN | NaN | NaN | NaN | 305 |
| **top** | NaN | NaN | NaN | 150.0 | NaN | NaN | NaN | NaN | ford pinto |
| **freq** | NaN | NaN | NaN | 22.0 | NaN | NaN | NaN | NaN | 6 |
| **mean** | 23.514573 | 5.454774 | 193.425879 | NaN | 2970.424623 | 15.568090 | 76.010050 | 1.572864 | NaN |
| **std** | 7.815984 | 1.701004 | 104.269838 | NaN | 846.841774 | 2.757689 | 3.697627 | 0.802055 | NaN |
| **min** | 9.000000 | 3.000000 | 68.000000 | NaN | 1613.000000 | 8.000000 | 70.000000 | 1.000000 | NaN |
| **25%** | 17.500000 | 4.000000 | 104.250000 | NaN | 2223.750000 | 13.825000 | 73.000000 | 1.000000 | NaN |
| **50%** | 23.000000 | 4.000000 | 148.500000 | NaN | 2803.500000 | 15.500000 | 76.000000 | 1.000000 | NaN |
| **75%** | 29.000000 | 8.000000 | 262.000000 | NaN | 3608.000000 | 17.175000 | 79.000000 | 2.000000 | NaN |
| **max** | 46.600000 | 8.000000 | 455.000000 | NaN | 5140.000000 | 24.800000 | 82.000000 | 3.000000 | NaN |

## 3) 10행으로 구성된 excel 파일을 활용하여 다시 분석하기

10행으로 구성된 excel 파일 활용하여 분석 < 파일명 : auto-mpg(10 lines) >
데이터가 너무 많으므로 10행으로 구성된 데이터로 다시 분석해보자.

In [36]:
```
# 엑셀 파일도 csv와 같은 형식으로 불러오면 됩니다.

file_path = 'D:/AI/auto-mpg(10 lines).xlsx'

df_excel_ten = pd.read_excel(file_path, 'Sheet1', header=None)
```

In [37]:
```
df_excel_ten
```

Out[37]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 13.0 | 8 | 360 | 175 | 3821 | 11.0 | 73 | 1 | amc ambassador brougham |
| **1** | 15.0 | 8 | 390 | 190 | 3850 | 8.5 | 70 | 2 | amc ambassador dpl |
| **2** | 17.0 | 8 | 304 | 150 | 3672 | 11.5 | 72 | 1 | amc ambassador sst |
| **3** | 18.0 | 6 | 232 | 100 | 2789 | 15.0 | 73 | 1 | amc gremlin |
| **4** | 18.1 | 6 | 258 | 120 | 3410 | 15.1 | 78 | 1 | amc concord d/l |
| **5** | 19.4 | 6 | 232 | 90 | 3210 | 17.2 | 78 | 1 | amc concord |
| **6** | 20.2 | 6 | 232 | 90 | 3265 | 18.2 | 79 | 1 | amc concord dl 6 |
| **7** | 21.0 | 6 | 199 | 90 | 2648 | 15.0 | 70 | 3 | amc gremlin |
| **8** | 23.0 | 4 | 151 | 80 | 3035 | 20.5 | 82 | 1 | amc concord dl |
| **9** | 24.3 | 4 | 151 | 90 | 3003 | 20.1 | 80 | 1 | amc concord |

In [38]:
```
df_excel_ten.columns = ['연비','실린더수','배기량','출력','차중', '가속능력','출시년도','제조국','모델명']
```

In [39]:
```
df_excel_ten  #총 9개의 항목이 출력
```

Out[39]:

| | 연비 | 실린더수 | 배기량 | 출력 | 차중 | 가속능력 | 출시년도 | 제조국 | 모델명 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 13.0 | 8 | 360 | 175 | 3821 | 11.0 | 73 | 1 | amc ambassador brougham |
| **1** | 15.0 | 8 | 390 | 190 | 3850 | 8.5 | 70 | 2 | amc ambassador dpl |
| **2** | 17.0 | 8 | 304 | 150 | 3672 | 11.5 | 72 | 1 | amc ambassador sst |
| **3** | 18.0 | 6 | 232 | 100 | 2789 | 15.0 | 73 | 1 | amc gremlin |
| **4** | 18.1 | 6 | 258 | 120 | 3410 | 15.1 | 78 | 1 | amc concord d/l |
| **5** | 19.4 | 6 | 232 | 90 | 3210 | 17.2 | 78 | 1 | amc concord |
| **6** | 20.2 | 6 | 232 | 90 | 3265 | 18.2 | 79 | 1 | amc concord dl 6 |
| **7** | 21.0 | 6 | 199 | 90 | 2648 | 15.0 | 70 | 3 | amc gremlin |
| **8** | 23.0 | 4 | 151 | 80 | 3035 | 20.5 | 82 | 1 | amc concord dl |
| **9** | 24.3 | 4 | 151 | 90 | 3003 | 20.1 | 80 | 1 | amc concord |

describe() : 수치 데이터에 해당하는 데이터를 기준으로 통계처리

In [40]:
```
df_excel_ten.describe()

# 수치 데이터에 해당하는 데이터를 기준으로 통계 처리됩니다.
# 총 9개의 항목중에서'모델명'에 해당하는 데이터의 통계값은 출력되지 않고 나머지 8개의 항목만 출력됩니다.
```

Out[40]:

| | 연비 | 실린더수 | 배기량 | 출력 | 차중 | 가속능력 | 출시년도 | 제조국 |
|---|---|---|---|---|---|---|---|---|
| **count** | 10.000000 | 10.00000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 |
| **mean** | 18.900000 | 6.20000 | 250.900000 | 117.500000 | 3270.300000 | 15.210000 | 75.500000 | 1.300000 |
| **std** | 3.454466 | 1.47573 | 80.171275 | 39.947883 | 417.610677 | 3.963851 | 4.377975 | 0.674949 |
| **min** | 13.000000 | 4.00000 | 151.000000 | 80.000000 | 2648.000000 | 8.500000 | 70.000000 | 1.000000 |
| **25%** | 17.250000 | 6.00000 | 207.250000 | 90.000000 | 3011.000000 | 12.375000 | 72.250000 | 1.000000 |
| **50%** | 18.750000 | 6.00000 | 232.000000 | 95.000000 | 3237.500000 | 15.050000 | 75.500000 | 1.000000 |
| **75%** | 20.800000 | 7.50000 | 292.500000 | 142.500000 | 3606.500000 | 17.950000 | 78.750000 | 1.000000 |
| **max** | 24.300000 | 8.00000 | 390.000000 | 190.000000 | 3850.000000 | 20.500000 | 82.000000 | 3.000000 |

In [41]:
```
df_excel_ten.describe(include='all')
```

Out[41]:

| | 연비 | 실린더수 | 배기량 | 출력 | 차중 | 가속능력 | 출시년도 | 제조국 | 모델명 |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 10.000000 | 10.00000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 | 10 |
| **unique** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 8 |
| **top** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | amc gremlin |
| **freq** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 2 |
| **mean** | 18.900000 | 6.20000 | 250.900000 | 117.500000 | 3270.300000 | 15.210000 | 75.500000 | 1.300000 | NaN |
| **std** | 3.454466 | 1.47573 | 80.171275 | 39.947883 | 417.610677 | 3.963851 | 4.377975 | 0.674949 | NaN |
| **min** | 13.000000 | 4.00000 | 151.000000 | 80.000000 | 2648.000000 | 8.500000 | 70.000000 | 1.000000 | NaN |
| **25%** | 17.250000 | 6.00000 | 207.250000 | 90.000000 | 3011.000000 | 12.375000 | 72.250000 | 1.000000 | NaN |
| **50%** | 18.750000 | 6.00000 | 232.000000 | 95.000000 | 3237.500000 | 15.050000 | 75.500000 | 1.000000 | NaN |
| **75%** | 20.800000 | 7.50000 | 292.500000 | 142.500000 | 3606.500000 | 17.950000 | 78.750000 | 1.000000 | NaN |
| **max** | 24.300000 | 8.00000 | 390.000000 | 190.000000 | 3850.000000 | 20.500000 | 82.000000 | 3.000000 | NaN |

count() : 각 항목에 대해 데이터가 몇 개 들어있는지 카운트 값 출력

In [42]:
```
1 df_excel_ten.count()
```

Out[42]:
```
연비        10
실린더수      10
배기량       10
출력        10
차중        10
가속능력      10
출시년도      10
제조국       10
모델명       10
dtype: int64
```

In [43]:
```
1 type(df_excel_ten.count())
```

Out[43]:
```
pandas.core.series.Series
```

value_counts() : 각 항목에 대해 데이터 값이 몇 번 들어있는지 카운트한 값을 내림차순 정렬하여 출력

In [44]:
```
1 model_count = df_excel_ten['모델명'].value_counts()
2
3 model_count  # 내림 차순 정렬 포함
```

Out[44]:
```
amc gremlin               2
amc concord               2
amc ambassador brougham   1
amc ambassador dpl        1
amc ambassador sst        1
amc concord d/l           1
amc concord dl 6          1
amc concord dl            1
Name: 모델명, dtype: int64
```

In [45]:
```
df_excel_ten
```

Out[45]:

| | 연비 | 실린더수 | 배기량 | 출력 | 차중 | 가속능력 | 출시년도 | 제조국 | 모델명 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 13.0 | 8 | 360 | 175 | 3821 | 11.0 | 73 | 1 | amc ambassador brougham |
| **1** | 15.0 | 8 | 390 | 190 | 3850 | 8.5 | 70 | 2 | amc ambassador dpl |
| **2** | 17.0 | 8 | 304 | 150 | 3672 | 11.5 | 72 | 1 | amc ambassador sst |
| **3** | 18.0 | 6 | 232 | 100 | 2789 | 15.0 | 73 | 1 | amc gremlin |
| **4** | 18.1 | 6 | 258 | 120 | 3410 | 15.1 | 78 | 1 | amc concord d/l |
| **5** | 19.4 | 6 | 232 | 90 | 3210 | 17.2 | 78 | 1 | amc concord |
| **6** | 20.2 | 6 | 232 | 90 | 3265 | 18.2 | 79 | 1 | amc concord dl 6 |
| **7** | 21.0 | 6 | 199 | 90 | 2648 | 15.0 | 70 | 3 | amc gremlin |
| **8** | 23.0 | 4 | 151 | 80 | 3035 | 20.5 | 82 | 1 | amc concord dl |
| **9** | 24.3 | 4 | 151 | 90 | 3003 | 20.1 | 80 | 1 | amc concord |

In [46]:
```
country_count = df_excel_ten['제조국'].value_counts()

country_count   # 내림 차순 정렬 포함
```

Out[46]:
```
1    8
2    1
3    1
Name: 제조국, dtype: int64
```

In [47]:
```
type(country_count)
```

Out[47]:
```
pandas.core.series.Series
```

mean() : 평균 값 출력

In [48]:
```
df_excel_ten.mean(numeric_only=True)  #numeric_only=True 수치형 데이터만 사용하겠다는 의미
```

Out[48]:
```
연비          18.90
실린더수         6.20
배기량        250.90
출력         117.50
차중        3270.30
가속능력        15.21
출시년도        75.50
제조국          1.30
dtype: float64
```

In [49]:
```
#DataFrame에서 숫자 데이터가 아닌 부분은 삭제가 필요함

re_df_excel_ten = df_excel_ten[['연비','실린더수','배기량','출력','차중','가속능력','출시년도','제조국']]
```

In [50]:
```
re_df_excel_ten
```

Out[50]:

| | 연비 | 실린더수 | 배기량 | 출력 | 차중 | 가속능력 | 출시년도 | 제조국 |
|---|---|---|---|---|---|---|---|---|
| **0** | 13.0 | 8 | 360 | 175 | 3821 | 11.0 | 73 | 1 |
| **1** | 15.0 | 8 | 390 | 190 | 3850 | 8.5 | 70 | 2 |
| **2** | 17.0 | 8 | 304 | 150 | 3672 | 11.5 | 72 | 1 |
| **3** | 18.0 | 6 | 232 | 100 | 2789 | 15.0 | 73 | 1 |
| **4** | 18.1 | 6 | 258 | 120 | 3410 | 15.1 | 78 | 1 |
| **5** | 19.4 | 6 | 232 | 90 | 3210 | 17.2 | 78 | 1 |
| **6** | 20.2 | 6 | 232 | 90 | 3265 | 18.2 | 79 | 1 |
| **7** | 21.0 | 6 | 199 | 90 | 2648 | 15.0 | 70 | 3 |
| **8** | 23.0 | 4 | 151 | 80 | 3035 | 20.5 | 82 | 1 |
| **9** | 24.3 | 4 | 151 | 90 | 3003 | 20.1 | 80 | 1 |

In [51]:
```
re_df_excel_ten.mean()
```

Out[51]:
```
연비          18.90
실린더수         6.20
배기량        250.90
출력         117.50
차중        3270.30
가속능력        15.21
출시년도        75.50
제조국          1.30
dtype: float64
```

In [52]:
```
print("연비 평균 : ",df_excel_ten['연비'].mean())
print("연비 평균 : ",df_excel_ten.연비.mean())  # 바로 위의 코드와 같은 의미입니다.
print('='*80)

print("차중 평균 : ",df_excel_ten['차중'].mean())
print("차중 평균 : ",df_excel_ten.차중.mean())  # 바로 위의 코드와 같은 의미입니다.
print('='*80)

print("연비와 차중 평균")
print(df_excel_ten[['연비','차중']].mean())
```

```
연비 평균 :  18.9
연비 평균 :  18.9
================================================================================
차중 평균 :  3270.3
차중 평균 :  3270.3
================================================================================
연비와 차중 평균
연비      18.9
차중    3270.3
dtype: float64
```

median() : 중앙값을 출력

In [53]:
```
df_excel_ten.median(numeric_only=True)  #numeric_only=True 수치형 데이터만 사용하겠다는 의미
```

Out[53]:
```
연비          18.75
실린더수         6.00
배기량        232.00
출력          95.00
차중        3237.50
가속능력        15.05
출시년도        75.50
제조국          1.00
dtype: float64
```

In [54]:
```
re_df_excel_ten.median()
```

Out[54]:
```
연비          18.75
실린더수         6.00
배기량        232.00
출력          95.00
차중        3237.50
가속능력        15.05
출시년도        75.50
제조국          1.00
dtype: float64
```

In [55]:
```
df_excel_ten['연비'].median()  # 연비의 중앙값을 출력합니다.
```

Out[55]:
```
18.75
```

max() : 최대값을 출력

In [56]:
```
df_excel_ten.max()
```

Out[56]:
```
연비              24.3
실린더수               8
배기량               390
출력                190
차중               3850
가속능력             20.5
출시년도               82
제조국                3
모델명      amc gremlin
dtype: object
```

In [57]:
```
df_excel_ten['연비'].max()  # 연비의 최대값을 출력합니다.
```

Out[57]:
```
24.3
```

min() : 최소값을 출력

In [58]:
```
df_excel_ten.min()
```

Out[58]:
```
연비                          13.0
실린더수                         4
배기량                         151
출력                           80
차중                         2648
가속능력                        8.5
출시년도                         70
제조국                           1
모델명      amc ambassador brougham
dtype: object
```

In [59]:
```
df_excel_ten['연비'].min()  # 연비의 최대값을 출력합니다.
```

Out[59]:
```
13.0
```

std() : 표준편차를 출력

In [60]:
```
df_excel_ten.std(numeric_only=True)  #numeric_only=True 수치형 데이터만 사용하겠다는 의미
```

Out[60]:
```
연비          3.454466
실린더수        1.475730
배기량        80.171275
출력         39.947883
차중        417.610677
가속능력        3.963851
출시년도        4.377975
제조국         0.674949
dtype: float64
```

In [61]:
```
1 re_df_excel_ten.std()
```

Out[61]:
```
연비          3.454466
실린더수        1.475730
배기량        80.171275
출력         39.947883
차중        417.610677
가속능력        3.963851
출시년도        4.377975
제조국         0.674949
dtype: float64
```

In [62]:
```
1 df_excel_ten['연비'].std()     # 연비의 표준편차를 출력합니다.
```

Out[62]:
```
3.4544657088084305
```

## 4) csv 파일을 읽어들인 데이터 프레임 df_csv 다시 살펴보기

In [63]:
```
1 df_csv
```

Out[63]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **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 [64]:
```
df_csv.columns = ['연비','실린더수','배기량','출력','차중', '가속능력','출시년도','제조국','모델명']
```

In [65]:
```
df_csv
```

Out[65]:

| | 연비 | 실린더수 | 배기량 | 출력 | 차중 | 가속능력 | 출시년도 | 제조국 | 모델명 |
|---|---|---|---|---|---|---|---|---|---|
| **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 [66]:
```
df_csv.min()

# 데이터프레임의 산술(숫자)데이터를 갖는 column에 대해서 최소값을 출력합니다.
```

Out[66]:
```
연비                         9.0
실린더수                       3
배기량                      68.0
출력                        100
차중                       1613
가속능력                      8.0
출시년도                       70
제조국                         1
모델명      amc ambassador brougham
dtype: object
```

In [67]:
```
df_csv['연비'].min()   # 연비의 최소값을 출력합니다.
```

Out[67]:
```
9.0
```

In [68]:
```
df_csv.std(numeric_only=True)  #numeric_only=True 수치형 데이터만 사용하겠다는 의미

# 데이터프레임의 산술(숫자)데이터를 갖는 column에 대해서 표준편차를 출력합니다.
```

Out[68]:
```
연비          7.815984
실린더수        1.701004
배기량       104.269838
차중        846.841774
가속능력        2.757689
출시년도        3.697627
제조국         0.802055
dtype: float64
```

In [69]:
```
# DataFrame에서 숫자 데이터가 아닌 부분은 삭제가 필요함
# '모델명'은 숫자가 아니므로 삭제
# 더불어'출력', '모델명' 부분에 "?" 텍스트가 포함되어 있기에 삭제

re_df_csv = df_csv[['연비','실린더수','배기량','차중','가속능력','출시년도','제조국']]
```

In [70]:
```
1 re_df_csv
```

Out[70]:

| | 연비 | 실린더수 | 배기량 | 차중 | 가속능력 | 출시년도 | 제조국 |
|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 3504 | 12.0 | 70 | 1 |
| **1** | 15.0 | 8 | 350.0 | 3693 | 11.5 | 70 | 1 |
| **2** | 18.0 | 8 | 318.0 | 3436 | 11.0 | 70 | 1 |
| **3** | 16.0 | 8 | 304.0 | 3433 | 12.0 | 70 | 1 |
| **4** | 17.0 | 8 | 302.0 | 3449 | 10.5 | 70 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **393** | 27.0 | 4 | 140.0 | 2790 | 15.6 | 82 | 1 |
| **394** | 44.0 | 4 | 97.0 | 2130 | 24.6 | 82 | 2 |
| **395** | 32.0 | 4 | 135.0 | 2295 | 11.6 | 82 | 1 |
| **396** | 28.0 | 4 | 120.0 | 2625 | 18.6 | 82 | 1 |
| **397** | 31.0 | 4 | 119.0 | 2720 | 19.4 | 82 | 1 |

398 rows × 7 columns

In [71]:
```
1 re_df_csv.std()   # 숫자 데이터 사이에 ?에 해당하는 데이터가 있기 때문에
```

Out[71]:
```
연비          7.815984
실린더수        1.701004
배기량       104.269838
차중        846.841774
가속능력        2.757689
출시년도        3.697627
제조국         0.802055
dtype: float64
```

In [72]:
```
1 df_csv['연비'].std()  # 연비의 표준편차를 출력합니다.
```

Out[72]:
```
7.815984312565782
```

# 참고

## • 피어슨 상관 계수 (Pearson Correlation Coefficient)

corr() : 상관계수를 출력

In [73]:
```
df_csv.corr(numeric_only=True)  #numeric_only=True 수치형 데이터만 사용하겠다는 의미

# 데이터프레임 df_csv의 산술(숫자)데이터를 갖는 column에 대해서 상관계수를 출력합니다.
```

Out[73]:

| | 연비 | 실린더수 | 배기량 | 차중 | 가속능력 | 출시년도 | 제조국 |
|---|---|---|---|---|---|---|---|
| **연비** | 1.000000 | -0.775396 | -0.804203 | -0.831741 | 0.420289 | 0.579267 | 0.563450 |
| **실린더수** | -0.775396 | 1.000000 | 0.950721 | 0.896017 | -0.505419 | -0.348746 | -0.562543 |
| **배기량** | -0.804203 | 0.950721 | 1.000000 | 0.932824 | -0.543684 | -0.370164 | -0.609409 |
| **차중** | -0.831741 | 0.896017 | 0.932824 | 1.000000 | -0.417457 | -0.306564 | -0.581024 |
| **가속능력** | 0.420289 | -0.505419 | -0.543684 | -0.417457 | 1.000000 | 0.288137 | 0.205873 |
| **출시년도** | 0.579267 | -0.348746 | -0.370164 | -0.306564 | 0.288137 | 1.000000 | 0.180662 |
| **제조국** | 0.563450 | -0.562543 | -0.609409 | -0.581024 | 0.205873 | 0.180662 | 1.000000 |

In [74]:
```
df_excel.corr(numeric_only=True)  #numeric_only=True 수치형 데이터만 사용하겠다는 의미

# 데이터프레임 df_excel의 산술(숫자)데이터를 갖는 column에 대해서 상관계수를 출력합니다.
```

Out[74]:

| | 연비 | 실린더수 | 배기량 | 차중 | 가속능력 | 출시년도 | 제조국 |
|---|---|---|---|---|---|---|---|
| **연비** | 1.000000 | -0.775396 | -0.804203 | -0.831741 | 0.420289 | 0.579267 | 0.563450 |
| **실린더수** | -0.775396 | 1.000000 | 0.950721 | 0.896017 | -0.505419 | -0.348746 | -0.562543 |
| **배기량** | -0.804203 | 0.950721 | 1.000000 | 0.932824 | -0.543684 | -0.370164 | -0.609409 |
| **차중** | -0.831741 | 0.896017 | 0.932824 | 1.000000 | -0.417457 | -0.306564 | -0.581024 |
| **가속능력** | 0.420289 | -0.505419 | -0.543684 | -0.417457 | 1.000000 | 0.288137 | 0.205873 |
| **출시년도** | 0.579267 | -0.348746 | -0.370164 | -0.306564 | 0.288137 | 1.000000 | 0.180662 |
| **제조국** | 0.563450 | -0.562543 | -0.609409 | -0.581024 | 0.205873 | 0.180662 | 1.000000 |

In [75]:
```
df_excel_ten.corr(numeric_only=True)  #numeric_only=True 수치형 데이터만 사용하겠다는 의미

# 데이터프레임 df_excel_ten의 산술(숫자)데이터를 갖는 column에 대해서 상관계수를 출력합니다.
```

Out[75]:

| | 연비 | 실린더수 | 배기량 | 출력 | 차중 | 가속능력 | 출시년도 | 제조국 |
|---|---|---|---|---|---|---|---|---|
| **연비** | 1.000000 | -0.924136 | -0.949955 | -0.869172 | -0.747389 | 0.899650 | 0.648730 | 0.014296 |
| **실린더수** | -0.924136 | 1.000000 | 0.941210 | 0.857568 | 0.733685 | -0.942519 | -0.773909 | 0.156174 |
| **배기량** | -0.949955 | 0.941210 | 1.000000 | 0.950335 | 0.849198 | -0.939375 | -0.660200 | 0.072484 |
| **출력** | -0.869172 | 0.857568 | 0.950335 | 1.000000 | 0.869915 | -0.919041 | -0.627375 | 0.072116 |
| **차중** | -0.747389 | 0.733685 | 0.849198 | 0.869915 | 1.000000 | -0.682625 | -0.236864 | -0.262103 |
| **가속능력** | 0.899650 | -0.942519 | -0.939375 | -0.919041 | -0.682625 | 1.000000 | 0.855728 | -0.296114 |
| **출시년도** | 0.648730 | -0.773909 | -0.660200 | -0.627375 | -0.236864 | 0.855728 | 1.000000 | -0.620437 |
| **제조국** | 0.014296 | 0.156174 | 0.072484 | 0.072116 | -0.262103 | -0.296114 | -0.620437 | 1.000000 |

In [76]:
```
print("● df_csv 데이터 프레임에서 연비와 차중의 상관관계 ")
df_csv[['연비','차중']].corr()
```

● df_csv 데이터 프레임에서 연비와 차중의 상관관계

Out[76]:

| | 연비 | 차중 |
|---|---|---|
| **연비** | 1.000000 | -0.831741 |
| **차중** | -0.831741 | 1.000000 |

In [77]:
```
print("● df_excel 데이터 프레임에서 연비와 차중의 상관관계 ")
df_excel[['연비','차중']].corr()
```

● df_excel 데이터 프레임에서 연비와 차중의 상관관계

Out[77]:

| | 연비 | 차중 |
|---|---|---|
| **연비** | 1.000000 | -0.831741 |
| **차중** | -0.831741 | 1.000000 |

In [78]:
```
print("● df_excel_ten 데이터 프레임에서 연비와 차중의 상관관계 ")
df_excel_ten[['연비','차중']].corr()
```

● df_excel_ten 데이터 프레임에서 연비와 차중의 상관관계

Out[78]:

| | 연비 | 차중 |
|---|---|---|
| **연비** | 1.000000 | -0.747389 |
| **차중** | -0.747389 | 1.000000 |

In [ ]:
```
```