

6. 자유자재로 데이터 가공하기



06-1. 데이터 전처리 - 원하는 형태로 데이터 가공하기

데이터 전처리(Preprocessing) - dplyr 패키지

함수	기능
filter()	행 추출
select()	열(변수) 추출
arrange()	정렬
mutate()	변수 추가
summarise()	통계치 산출
group_by()	집단별로 나누기
left_join()	데이터 합치기(열)
bind_rows()	데이터 합치기(행)

06-2. 조건에 맞는 데이터만 추출하기

class	english	science
2	98	50
1	97	60
2	86	78
1	98	58
1	80	65
2	89	98



class	english	science
1	97	60
1	98	58
1	80	65

조건에 맞는 데이터만 추출하기

dplyr 패키지 로드 & 데이터 준비

```
library(dplyr)
exam <- read.csv("csv_exam.csv")
exam
```

##		id	class	math	english	science
##	1	1	1	50	98	50
##	2	2	1	60	97	60
##	3	3	1	45	86	78
##	4	4	1	30	98	58
##	5	5	2	25	80	65
##	6	6	2	50	89	98
##	7	7	2	80	90	45
##	8	8	2	90	78	25
##	9	9	3	20	98	15
##	10	10	3	50	98	45
##	11	11	3	65	65	65
##	12	12	3	45	85	32
##	13	13	4	46	98	65
##	14	14	4	48	87	12
##	15	15	4	75	56	78
##	16	16	4	58	98	65
##	17	17	5	65	68	98
##	18	18	5	80	78	90

##	19	19	5	89	68	87
##	20	20	5	78	83	58

```
# exam 에서 class 가 1 인 경우만 추출하여 출력  
exam %>% filter(class == 1)
```

```
##      id class math english science  
## 1     1     1   50      98       50  
## 2     2     1   60      97       60  
## 3     3     1   45      86       78  
## 4     4     1   30      98       58
```

[참고] 단축키 [Ctrl+Shit+M]으로 %>% 기호 입력

2 반인 경우만 추출

```
exam %>% filter(class == 2)
```

```
##   id class math english science
## 1  5     2   25      80      65
## 2  6     2   50      89      98
## 3  7     2   80      90      45
## 4  8     2   90      78      25
```

1 반이 아닌 경우

```
exam %>% filter(class != 1)
```

##	id	class	math	english	science
## 1	5	2	25	80	65
## 2	6	2	50	89	98
## 3	7	2	80	90	45
## 4	8	2	90	78	25
## 5	9	3	20	98	15
## 6	10	3	50	98	45
## 7	11	3	65	65	65
## 8	12	3	45	85	32
## 9	13	4	46	98	65
## 10	14	4	48	87	12
## 11	15	4	75	56	78
## 12	16	4	58	98	65
## 13	17	5	65	68	98
## 14	18	5	80	78	90
## 15	19	5	89	68	87
## 16	20	5	78	83	58

3 반이 아닌 경우

```
exam %>% filter(class != 3)
```

##	id	class	math	english	science
## 1	1	1	50	98	50
## 2	2	1	60	97	60
## 3	3	1	45	86	78
## 4	4	1	30	98	58
## 5	5	2	25	80	65
## 6	6	2	50	89	98
## 7	7	2	80	90	45
## 8	8	2	90	78	25
## 9	13	4	46	98	65
## 10	14	4	48	87	12
## 11	15	4	75	56	78
## 12	16	4	58	98	65
## 13	17	5	65	68	98
## 14	18	5	80	78	90
## 15	19	5	89	68	87
## 16	20	5	78	83	58

초과, 미만, 이상, 이하 조건 걸기

수학 점수가 50 점을 초과한 경우

```
exam %>% filter(math > 50)
```

```
##      id class math english science
## 1     2     1   60      97       60
## 2     7     2   80      90       45
## 3     8     2   90      78       25
## 4    11     3   65      65       65
## 5    15     4   75      56       78
## 6    16     4   58      98       65
## 7    17     5   65      68       98
## 8    18     5   80      78       90
## 9    19     5   89      68       87
## 10   20     5   78      83       58
```

수학 점수가 50 점 미만인 경우

```
exam %>% filter(math < 50)
```

##	id	class	math	english	science	
##	1	3	1	45	86	78
##	2	4	1	30	98	58
##	3	5	2	25	80	65
##	4	9	3	20	98	15
##	5	12	3	45	85	32
##	6	13	4	46	98	65
##	7	14	4	48	87	12

영어점수가 80 점 이상인 경우

```
exam %>% filter(english >= 80)
```

##		id	class	math	english	science
##	1	1	1	50	98	50
##	2	2	1	60	97	60
##	3	3	1	45	86	78
##	4	4	1	30	98	58
##	5	5	2	25	80	65
##	6	6	2	50	89	98
##	7	7	2	80	90	45
##	8	9	3	20	98	15
##	9	10	3	50	98	45
##	10	12	3	45	85	32
##	11	13	4	46	98	65
##	12	14	4	48	87	12
##	13	16	4	58	98	65
##	14	20	5	78	83	58

영어점수가 80 점 이하인 경우

```
exam %>% filter(english <= 80)
```

##	id	class	math	english	science
## 1	5	2	25	80	65
## 2	8	2	90	78	25
## 3	11	3	65	65	65
## 4	15	4	75	56	78
## 5	17	5	65	68	98
## 6	18	5	80	78	90
## 7	19	5	89	68	87

여러 조건을 충족하는 행 추출하기

```
# 1 반 이면서 수학 점수가 50 점 이상인 경우  
exam %>% filter(class == 1 & math >= 50)
```

```
##      id class math english science  
## 1    1     1   50      98       50  
## 2    2     1   60      97       60
```

2 반 이면서 영어점수가 80 점 이상인 경우

```
exam %>% filter(class == 2 & english >= 80)
```

```
##   id class math english science
## 1  5     2   25      80       65
## 2  6     2   50      89       98
## 3  7     2   80      90       45
```

여러 조건 중 하나 이상 충족하는 행 추출하기

수학 점수가 90 점 이상이거나 영어점수가 90 점 이상인 경우

```
exam %>% filter(math >= 90 | english >= 90)
```

```
##   id class math english science
## 1   1     1   50      98      50
## 2   2     1   60      97      60
## 3   4     1   30      98      58
## 4   7     2   80      90      45
## 5   8     2   90      78      25
## 6   9     3   20      98      15
## 7  10     3   50      98      45
## 8  13     4   46      98      65
## 9  16     4   58      98      65
```


영어점수가 90 점 미만이거나 과학점수가 50 점 미만인 경우

```
exam %>% filter(english < 90 | science < 50)
```

##	id	class	math	english	science
## 1	3	1	45	86	78
## 2	5	2	25	80	65
## 3	6	2	50	89	98
## 4	7	2	80	90	45
## 5	8	2	90	78	25
## 6	9	3	20	98	15
## 7	10	3	50	98	45
## 8	11	3	65	65	65
## 9	12	3	45	85	32
## 10	14	4	48	87	12
## 11	15	4	75	56	78
## 12	17	5	65	68	98
## 13	18	5	80	78	90
## 14	19	5	89	68	87
## 15	20	5	78	83	58

목록에 해당되는 행 추출하기

```
exam %>% filter(class == 1 | class == 3 | class == 5) # 1, 3, 5 반에 해당되면 추출
```

##		id	class	math	english	science
##	1	1	1	50	98	50
##	2	2	1	60	97	60
##	3	3	1	45	86	78
##	4	4	1	30	98	58
##	5	9	3	20	98	15
##	6	10	3	50	98	45
##	7	11	3	65	65	65
##	8	12	3	45	85	32
##	9	17	5	65	68	98
##	10	18	5	80	78	90
##	11	19	5	89	68	87
##	12	20	5	78	83	58

%in% 기호 사용하기

```
exam %>% filter(class %in% c(1,3,5)) # 1, 3, 5 반에 해당하면 추출
```

##	id	class	math	english	science
## 1	1	1	50	98	50
## 2	2	1	60	97	60
## 3	3	1	45	86	78
## 4	4	1	30	98	58
## 5	9	3	20	98	15
## 6	10	3	50	98	45
## 7	11	3	65	65	65
## 8	12	3	45	85	32
## 9	17	5	65	68	98
## 10	18	5	80	78	90
## 11	19	5	89	68	87
## 12	20	5	78	83	58

추출한 행으로 데이터 만들기

```
class1 <- exam %>% filter(class == 1) # class 가 1 인 행 추출, class1 에 할당
class2 <- exam %>% filter(class == 2) # class 가 2 인 행 추출, class2 에 할당

mean(class1$math) # 1 반 수학 점수 평균 구하기

## [1] 46.25

mean(class2$math) # 2 반 수학 점수 평균 구하기

## [1] 61.25
```

R에서 사용하는 기호들

논리 연산자 기능

<	작다
<=	작거나 같다
>	크다
>=	크거나 같다
==	같다
!=	같지 않다
	또는
&	그리고
%in%	매칭 확인

R에서 사용하는 기호들

산술 연산자 기능

+	더하기
-	빼기
*	곱하기
/	나누기
^ , **	제곱
%/%	나눗셈의 몫
%%	나눗셈의 나머지

혼자서 해보기

mpg 데이터를 이용해 분석 문제를 해결해 보세요.

- Q1. 자동차 배기량에 따라 고속도로 연비가 다른지 알아보려고 합니다. `displ`(배기량)이 4 이하인 자동차와 5 이상인 자동차 중 어떤 자동차의 `hwy`(고속도로 연비)가 평균적으로 더 높은지 알아보세요.
- Q2. 자동차 제조 회사에 따라 도시 연비가 다른지 알아보려고 합니다. "audi"와 "toyota" 중 어느 `manufacturer`(자동차 제조 회사)의 `cty`(도시 연비)가 평균적으로 더 높은지 알아보세요.
- Q3. "chevrolet", "ford", "honda" 자동차의 고속도로 연비 평균을 알아보려고 합니다. 이 회사들의 자동차를 추출한 뒤 `hwy` 전체 평균을 구해보세요.

힌트

Q1. 특정 조건에 해당하는 데이터를 추출해서 평균을 구하면 해결할 수 있는 문제입니다. `filter()`를 이용해 `displ` 변수가 특정 값을 지닌 행을 추출해 새로운 변수에 할당한 다음 평균을 구해보세요.

Q2. 앞 문제와 동일한 절차로 해결하면 됩니다. 단, 변수의 값이 숫자가 아니라 문자라는 점이 다릅니다.

Q3. '여러 조건 중 하나 이상 충족'하면 추출하도록 `filter()` 함수를 구성해보세요. 이렇게 추출한 데이터로 평균을 구하면 됩니다. `%in%`를 이용하면 코드를 짧게 만들 수 있습니다.

정답

Q1. 자동차 배기량에 따라 고속도로 연비가 다른지 알아보려고 합니다. `displ`(배기량)이 4 이하인 자동차와 5 이상인 자동차 중 어떤 자동차의 `hwy`(고속도로 연비)가 평균적으로 더 높은지 알아보세요.

```
mpg <- as.data.frame(ggplot2::mpg)    # mpg 데이터 불러오기

mpg_a <- mpg %>% filter(displ <= 4)    # displ 4 이하 추출
mpg_b <- mpg %>% filter(displ >= 5)    # displ 5 이상 추출

mean(mpg_a$hwy)    # displ 4 이하 hwy 평균
## [1] 25.96319

mean(mpg_b$hwy)    # displ 5 이상 hwy 평균
## [1] 18.07895
```

Q2. 자동차 제조 회사에 따라 도시 연비가 다른지 알아보려고 합니다. "audi"와 "toyota" 중 어느 manufacturer(자동차 제조 회사)의 cty(도시 연비)가 평균적으로 더 높은지 알아보세요.

```
mpg_audi <- mpg %>% filter(manufacturer == "audi")      # audi 추출
mpg_toyota <- mpg %>% filter(manufacturer == "toyota")   # toyota 추출

mean(mpg_audi$cty)    # audi 의 cty 평균

## [1] 17.61111

mean(mpg_toyota$cty)  # toyota 의 cty 평균

## [1] 18.52941
```

Q3. "chevrolet", "ford", "honda" 자동차의 고속도로 연비 평균을 알아보려고 합니다. 이 회사들의 자동차를 추출한 뒤 hwy 전체 평균을 구해보세요.

```
# manufacturer 가 chevrolet, ford, honda 에 해당하면 추출  
mpg_new <- mpg %>% filter(manufacturer %in% c("chevrolet", "ford", "honda"))  
mean(mpg_new$hwy)  
## [1] 22.50943
```

06-3. 필요한 변수만 추출하기

id	class	english	science
1	2	98	50
2	1	97	60
3	2	86	78
4	1	98	58
5	1	80	65
6	2	89	98



class	english
2	98
1	97
2	86
1	98
1	80
2	89

```
exam %>% select(math) # math 추출
```

```
##      math
## 1      50
## 2      60
## 3      45
## 4      30
## 5      25
## 6      50
## 7      80
## 8      90
## 9      20
## 10     50
## 11     65
## 12     45
## 13     46
## 14     48
## 15     75
## 16     58
## 17     65
## 18     80
## 19     89
## 20     78
```

```
exam %>% select(english) # english 추출
```

```
##      english
## 1         98
## 2         97
## 3         86
## 4         98
## 5         80
## 6         89
## 7         90
## 8         78
## 9         98
## 10        98
## 11        65
## 12        85
## 13        98
## 14        87
## 15        56
## 16        98
## 17        68
## 18        78
## 19        68
## 20        83
```

여러 변수 추출하기

```
exam %>% select(class, math, english) # class, math, english 변수 추출
```

```
##      class math english
## 1         1   50      98
## 2         1   60      97
## 3         1   45      86
## 4         1   30      98
## 5         2   25      80
## 6         2   50      89
## 7         2   80      90
## 8         2   90      78
## 9         3   20      98
## 10        3   50      98
## 11        3   65      65
## 12        3   45      85
## 13        4   46      98
## 14        4   48      87
## 15        4   75      56
## 16        4   58      98
## 17        5   65      68
## 18        5   80      78
## 19        5   89      68
## 20        5   78      83
```

변수 제외하기

```
exam %>% select(-math)  # math 제외
```

##	id	class	english	science
## 1	1	1	98	50
## 2	2	1	97	60
## 3	3	1	86	78
## 4	4	1	98	58
## 5	5	2	80	65
## 6	6	2	89	98
## 7	7	2	90	45
## 8	8	2	78	25
## 9	9	3	98	15
## 10	10	3	98	45
## 11	11	3	65	65
## 12	12	3	85	32
## 13	13	4	98	65
## 14	14	4	87	12
## 15	15	4	56	78
## 16	16	4	98	65
## 17	17	5	68	98
## 18	18	5	78	90
## 19	19	5	68	87
## 20	20	5	83	58


```
exam %>% select(-math, -english) # math, english 제외
```

```
##      id class science
## 1     1     1      50
## 2     2     1      60
## 3     3     1      78
## 4     4     1      58
## 5     5     2      65
## 6     6     2      98
## 7     7     2      45
## 8     8     2      25
## 9     9     3      15
## 10    10    3      45
## 11    11    3      65
## 12    12    3      32
## 13    13    4      65
## 14    14    4      12
## 15    15    4      78
## 16    16    4      65
## 17    17    5      98
## 18    18    5      90
## 19    19    5      87
## 20    20    5      58
```

dplyr 함수 조합하기

```
# class 가 1 인 행만 추출한 다음 english 추출  
exam %>% filter(class == 1) %>% select(english)
```

```
##    english  
## 1      98  
## 2      97  
## 3      86  
## 4      98
```

가독성 있게 줄 바꾸기

```
exam %>%  
  filter(class == 1) %>%  # class 가 1 인 행 추출  
  select(english)         # english 추출
```

일부만 출력하기

```
exam %>%  
  select(id, math) %>% # id, math 추출  
  head                 # 앞부분 6 행까지 추출
```

```
##    id math  
## 1   1   50  
## 2   2   60  
## 3   3   45  
## 4   4   30  
## 5   5   25  
## 6   6   50
```

일부만 출력하기

```
exam %>%  
  select(id, math) %>% # id, math 추출  
  head(10)             # 앞부분 10 행까지 추출
```

```
##      id math  
## 1      1   50  
## 2      2   60  
## 3      3   45  
## 4      4   30  
## 5      5   25  
## 6      6   50  
## 7      7   80  
## 8      8   90  
## 9      9   20  
## 10     10   50
```

혼자서 해보기

mpg 데이터를 이용해서 분석 문제를 해결해보세요.

- Q1. mpg 데이터는 11 개 변수로 구성되어 있습니다. 이 중 일부만 추출해서 분석에 활용하려고 합니다. mpg 데이터에서 class(자동차 종류), cty(도시 연비) 변수를 추출해 새로운 데이터를 만드세요. 새로 만든 데이터의 일부를 출력해서 두 변수로만 구성되어 있는지 확인하세요.
- Q2. 자동차 종류에 따라 도시 연비가 다른지 알아보려고 합니다. 앞에서 추출한 데이터를 이용해서 class(자동차 종류)가 "suv"인 자동차와 "compact"인 자동차 중 어떤 자동차의 cty(도시 연비)가 더 높은지 알아보세요.

힌트

Q1. `select()`로 변수를 추출해서 새로운 데이터를 만들어 보세요.

Q2. `filter()`로 조건에 해당하는 데이터를 추출한 뒤 평균을 구하면 해결할 수 있습니다.

정답

Q1. mpg 데이터는 11개 변수로 구성되어 있습니다. 이 중 일부만 추출해서 분석에 활용하려고 합니다. mpg 데이터에서 class(자동차 종류), cty(도시 연비) 변수를 추출해 새로운 데이터를 만드세요. 새로 만든 데이터의 일부를 출력해서 두 변수로만 구성되어 있는지 확인하세요.

```
mpg <- as.data.frame(ggplot2::mpg)  # mpg 데이터 불러오기

df <- mpg %>% select(class, cty)      # class, cty 변수 추출
head(df)                             # df 일부 출력

##      class cty
## 1 compact  18
## 2 compact  21
## 3 compact  20
## 4 compact  21
## 5 compact  16
## 6 compact  18
```


Q2. 자동차 종류에 따라 도시 연비가 다른지 알아보려고 합니다. 앞에서 추출한 데이터를 이용해서 `class`(자동차 종류)가 "suv"인 자동차와 "compact"인 자동차 중 어떤 자동차의 `cty`(도시 연비)가 더 높은지 알아보세요.

```
df_suv <- df %>% filter(class == "suv") # class 가 suv 인 행 추출  
df_compact <- df %>% filter(class == "compact") # class 가 compact 인 행 추출  
  
mean(df_suv$cty) # suv 의 cty 평균  
## [1] 13.5  
  
mean(df_compact$cty) # compact 의 cty 평균  
## [1] 20.12766
```

06-4. 순서대로 정렬하기

id	english	science		id	english	science
1	98	50	➔	6	89	98
2	97	60		5	86	78
3	86	78		4	80	65
4	98	58		3	97	60
5	80	65		2	98	58
6	89	98		1	98	50

오름차순으로 정렬하기

```
exam %>% arrange(math) # math 오름차순 정렬
```

##	id	class	math	english	science
## 1	9	3	20	98	15
## 2	5	2	25	80	65
## 3	4	1	30	98	58
## 4	3	1	45	86	78
## 5	12	3	45	85	32
## 6	13	4	46	98	65
## 7	14	4	48	87	12
## 8	1	1	50	98	50
## 9	6	2	50	89	98
## 10	10	3	50	98	45
## 11	16	4	58	98	65
## 12	2	1	60	97	60
## 13	11	3	65	65	65
## 14	17	5	65	68	98
## 15	15	4	75	56	78
## 16	20	5	78	83	58
## 17	7	2	80	90	45
## 18	18	5	80	78	90
## 19	19	5	89	68	87
## 20	8	2	90	78	25

내림차순으로 정렬하기

```
exam %>% arrange(desc(math)) # math 내림차순 정렬
```

##	id	class	math	english	science
## 1	8	2	90	78	25
## 2	19	5	89	68	87
## 3	7	2	80	90	45
## 4	18	5	80	78	90
## 5	20	5	78	83	58
## 6	15	4	75	56	78
## 7	11	3	65	65	65
## 8	17	5	65	68	98
## 9	2	1	60	97	60
## 10	16	4	58	98	65
## 11	1	1	50	98	50
## 12	6	2	50	89	98
## 13	10	3	50	98	45
## 14	14	4	48	87	12
## 15	13	4	46	98	65
## 16	3	1	45	86	78
## 17	12	3	45	85	32
## 18	4	1	30	98	58
## 19	5	2	25	80	65
## 20	9	3	20	98	15

정렬 기준 변수 여러개 지정

```
exam %>% arrange(class, math)  # class 및 math 오름차순 정렬
```

##	id	class	math	english	science
## 1	4	1	30	98	58
## 2	3	1	45	86	78
## 3	1	1	50	98	50
## 4	2	1	60	97	60
## 5	5	2	25	80	65
## 6	6	2	50	89	98
## 7	7	2	80	90	45
## 8	8	2	90	78	25
## 9	9	3	20	98	15
## 10	12	3	45	85	32
## 11	10	3	50	98	45
## 12	11	3	65	65	65
## 13	13	4	46	98	65
## 14	14	4	48	87	12
## 15	16	4	58	98	65
## 16	15	4	75	56	78
## 17	17	5	65	68	98
## 18	20	5	78	83	58
## 19	18	5	80	78	90
## 20	19	5	89	68	87

혼자서 해보기

mpg 데이터를 이용해서 분석 문제를 해결해보세요.

- "audi"에서 생산한 자동차 중에 어떤 자동차 모델의 hwy(고속도로 연비)가 높은지 알아보려고 합니다.
"audi"에서 생산한 자동차 중 hwy가 1~5위에 해당하는 자동차의 데이터를 출력하세요.

힌트

`filter()`를 이용해 "audi"에서 생산한 자동차만 추출하고, `arrange()`로 `hwy`를 내림차순 정렬하면 됩니다. `head()`를 이용하면 이 중 특정 순위에 해당하는 자동차만 출력할 수 있습니다.

정답

```
mpg <- as.data.frame(ggplot2::mpg)           # mpg 데이터 불러오기

mpg %>% filter(manufacturer == "audi") %>%    # audi 추출
  arrange(desc(hwy)) %>%                     # hwy 내림차순 정렬
  head(5)                                    # 5 행까지 출력

##      manufacturer      model displ  year  cyl      trans drv  cty   hwy fl   class
## 1          audi          a4    2.0  2008    4 manual(m6)   f   20    31  p compact
## 2          audi          a4    2.0  2008    4   auto(av)   f   21    30  p compact
## 3          audi          a4    1.8  1999    4   auto(l5)   f   18    29  p compact
## 4          audi          a4    1.8  1999    4 manual(m5)   f   21    29  p compact
## 5      audi a4 quattro    2.0  2008    4 manual(m6)   4   20    28  p compact
```


06-5. 파생변수 추가하기

id	english	science		id	english	science	total
1	98	50		1	98	50	148
2	97	60		2	97	60	157
3	86	78	→	3	86	78	164
4	98	58		4	98	58	156
5	80	65		5	80	65	145
6	89	98		6	89	98	187

```
exam %>%
```

```
  mutate(total = math + english + science) %>% # 총합 변수 추가
```

```
  head # 일부 추출
```

```
##   id class math english science total
## 1  1     1   50      98      50   198
## 2  2     1   60      97      60   217
## 3  3     1   45      86      78   209
## 4  4     1   30      98      58   186
## 5  5     2   25      80      65   170
## 6  6     2   50      89      98   237
```

여러 파생변수 한 번에 추가하기

```
exam %>%  
  mutate(total = math + english + science,           # 총합 변수 추가  
          mean = (math + english + science)/3) %>%    # 총평균 변수 추가  
  head                                                # 일부 추출
```

##	id	class	math	english	science	total	mean
##	1	1	50	98	50	198	66.00000
##	2	1	60	97	60	217	72.33333
##	3	1	45	86	78	209	69.66667
##	4	1	30	98	58	186	62.00000
##	5	2	25	80	65	170	56.66667
##	6	2	50	89	98	237	79.00000

mutate()에 ifelse() 적용하기

```
exam %>%  
  mutate(test = ifelse(science >= 60, "pass", "fail")) %>%  
  head
```

##	id	class	math	english	science	test
## 1	1	1	50	98	50	fail
## 2	2	1	60	97	60	pass
## 3	3	1	45	86	78	pass
## 4	4	1	30	98	58	fail
## 5	5	2	25	80	65	pass
## 6	6	2	50	89	98	pass

추가한 변수를 dplyr 코드에 바로 활용하기

```
exam %>%  
  mutate(total = math + english + science) %>% # 총합 변수 추가  
  arrange(total) %>% # 총합 변수 기준 정렬  
  head # 일부 추출
```



```
##   id class math english science total  
## 1   9     3   20      98       15   133  
## 2  14     4   48      87       12   147  
## 3  12     3   45      85       32   162  
## 4   5     2   25      80       65   170  
## 5   4     1   30      98       58   186  
## 6   8     2   90      78       25   193
```

혼자서 해보기

mpg 데이터를 이용해서 분석 문제를 해결해보세요.

mpg 데이터는 연비를 나타내는 변수가 hwy(고속도로 연비), cty(도시 연비) 두 종류로 분리되어 있습니다. 두 변수를 각각 활용하는 대신 하나의 통합 연비 변수를 만들어 분석하려고 합니다.

- Q1. mpg 데이터 복사본을 만들고, cty 와 hwy 를 더한 '합산 연비 변수'를 추가하세요.
- Q2. 앞에서 만든 '합산 연비 변수'를 2 로 나눠 '평균 연비 변수'를 추가세요.
- Q3. '평균 연비 변수'가 가장 높은 자동차 3 종의 데이터를 출력하세요.
- Q4. 1~3 번 문제를 해결할 수 있는 하나로 연결된 dplyr 구문을 만들어 출력하세요. 데이터는 복사본 대신 mpg 원본을 이용하세요.

힌트

Q1. `mutate()`를 적용한 결과를 `<-`를 이용해 데이터 프레임에 할당하는 형태로 코드를 작성하면 기존 데이터 프레임에 변수가 추가됩니다.

Q3. `arrange()`와 `head()`를 조합하면 됩니다.

Q4. 앞에서 만든 코드들을 `%>%`를 이용해 연결하면 됩니다. 변수를 추가하는 작업을 하나의 `mutate()` 구성하면 코드를 더 간결하게 만들 수 있습니다.

정답

Q1. mpg 데이터 복사본을 만들고, cty와 hwy를 더한 '합산 연비 변수'를 추가하세요.

```
mpg <- as.data.frame(ggplot2::mpg)           # mpg 데이터 불러오기  
mpg_new <- mpg                                # 복사본 만들기  
  
mpg_new <- mpg_new %>% mutate(total = cty + hwy) # 합산 변수 만들기
```


Q2. 앞에서 만든 '합산 연비 변수'를 2로 나눠 '평균 연비 변수'를 추가세요.

```
mpg_new <- mpg_new %>% mutate(mean = total/2)      # 평균 변수 만들기
```

Q3. '평균 연비 변수'가 가장 높은 자동차 3종의 데이터를 출력하세요.

```
mpg_new %>%  
  arrange(desc(mean)) %>% # 내림차순 정렬  
  head(3)                 # 상위 3행 출력
```



```
##   manufacturer      model displ  year  cyl      trans drv  cty  hwy fl  
## 1  volkswagen new beetle   1.9 1999   4 manual(m5)  f   35  44  d  
## 2  volkswagen      jetta   1.9 1999   4 manual(m5)  f   33  44  d  
## 3  volkswagen new beetle   1.9 1999   4   auto(l4)  f   29  41  d  
##           class total mean  
## 1 subcompact      79 39.5  
## 2   compact      77 38.5  
## 3 subcompact      70 35.0
```

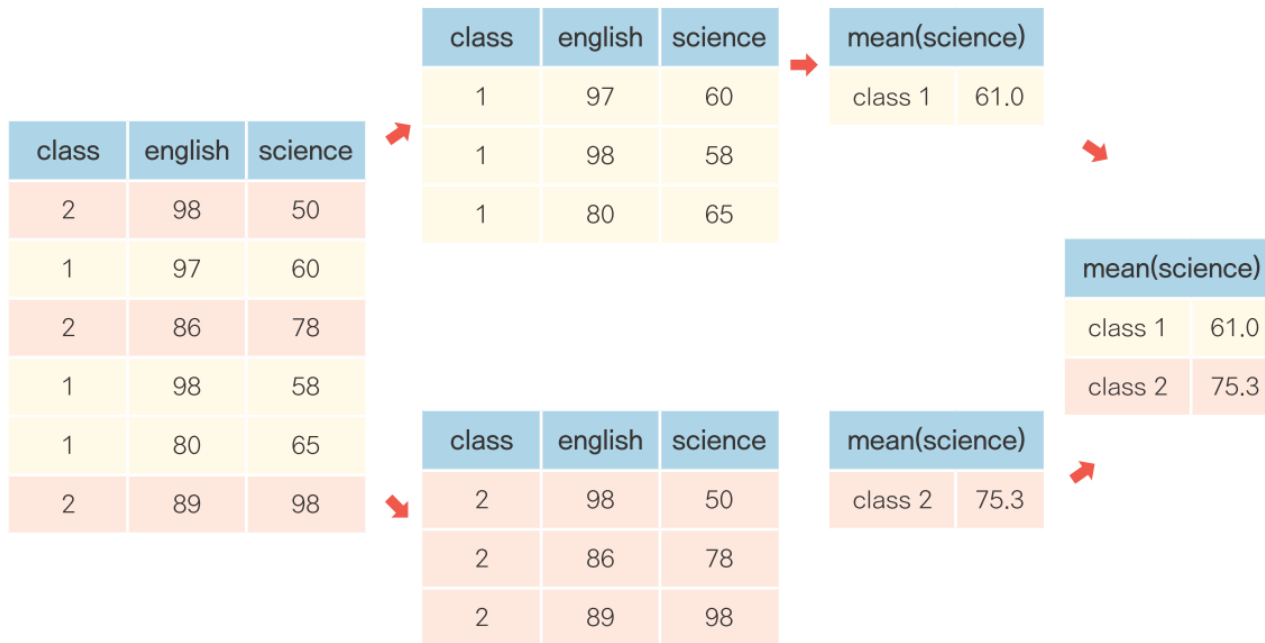
Q4. 1~3번 문제를 해결할 수 있는 하나로 연결된 `dplyr` 구문을 만들어 출력하세요. 데이터는 복사본 대신 `mpg` 원본을 이용하세요.

```
mpg %>%  
  mutate(total = cty + hwy,      # 합산 변수 만들기  
         mean = total/2) %>%    # 평균 변수 만들기  
  arrange(desc(mean)) %>%      # 내림차순 정렬  
  head(3)                      # 상위 3 행 출력
```



```
##   manufacturer      model displ  year  cyl      trans  drv  cty  hwy  fl  
## 1  volkswagen new beetle   1.9 1999   4 manual(m5)  f   35   44  d  
## 2  volkswagen      jetta   1.9 1999   4 manual(m5)  f   33   44  d  
## 3  volkswagen new beetle   1.9 1999   4  auto(l4)   f   29   41  d  
##           class total mean  
## 1 subcompact     79 39.5  
## 2   compact     77 38.5  
## 3 subcompact     70 35.0
```

06-6. 집단별로 요약하기



집단별로 요약하기

요약하기

```
exam %>% summarise(mean_math = mean(math)) # math 평균 산출
```

```
##    mean_math  
## 1        57.45
```

집단별로 요약하기

```
exam %>%  
  group_by(class) %>%  
  summarise(mean_math = mean(math))  
  
## # A tibble: 5 x 2  
##   class mean_math  
##   <int>     <dbl>  
## 1     1      46.25  
## 2     2      61.25  
## 3     3      45.00  
## 4     4      56.75  
## 5     5      78.00
```

여러 요약통계량 한 번에 산출하기

```
exam %>%  
  group_by(class) %>%  
  summarise(mean_math = mean(math),      # class 별로 분리  
             sum_math = sum(math),       # math 평균  
             median_math = median(math), # math 합계  
             n = n())                   # math 중앙값  
                                         # 학생 수
```

A tibble: 5 x 5

##	class	mean_math	sum_math	median_math	n
##	<int>	<dbl>	<int>	<dbl>	<int>
## 1	1	46.25	185	47.5	4
## 2	2	61.25	245	65.0	4
## 3	3	45.00	180	47.5	4
## 4	4	56.75	227	53.0	4
## 5	5	78.00	312	79.0	4

자주 사용하는 요약통계량 함수

함수	의미
mean()	평균
sd()	표준편차
sum()	합계
median()	중앙값
min()	최솟값
max()	최댓값
n()	빈도

각 집단별로 다시 집단 나누기

```
mpg %>%  
  group_by(manufacturer, drv) %>%      # 회사별, 구방방식별 분리  
  summarise(mean_cty = mean(cty)) %>%  # cty 평균 산출  
  head(10)                             # 일부 출력
```

```
## # A tibble: 10 x 3  
## # Groups:   manufacturer [5]  
##   manufacturer    drv mean_cty  
##   <chr>    <chr>    <dbl>  
## 1      audi      4 16.81818  
## 2      audi      f 18.85714  
## 3  chevrolet     4 12.50000  
## 4  chevrolet     f 18.80000  
## 5  chevrolet     r 14.10000  
## 6    dodge      4 12.00000  
## 7    dodge      f 15.81818  
## 8     ford      4 13.30769  
## 9     ford      r 14.75000  
## 10   honda      f 24.44444
```

dplyr 조합하기

문제) 회사별로 "suv" 자동차의 도시 및 고속도로 통합 연비 평균을 구해 내림차순으로 정렬하고, 1~5위까지 출력하기

분석 절차 생각해보기

절차	기능	dplyr 함수
1	회사별로 분리	group_by()
2	suv 추출	filter()
3	통합 연비 변수 생성	mutate()
4	통합 연비 평균 산출	summarise()
5	내림차순 정렬	arrange()
6	1~5위까지 출력	head()

dplyr 조합하기

```
mpg %>%  
  group_by(manufacturer) %>%           # 회사별로 분리  
  filter(class == "suv") %>%          # suv 추출  
  mutate(tot = (cty+hwy)/2) %>%       # 통합 연비 변수 생성  
  summarise(mean_tot = mean(tot)) %>% # 통합 연비 평균 산출  
  arrange(desc(mean_tot)) %>%        # 내림차순 정렬  
  head(5)                             # 1~5 위까지 출력  
  
## # A tibble: 5 x 2  
##   manufacturer mean_tot  
##   <chr>      <dbl>  
## 1      subaru 21.91667  
## 2      toyota 16.31250  
## 3      nissan 15.87500  
## 4    mercury 15.62500  
## 5       jeep 15.56250
```

혼자서 해보기

mpg 데이터를 이용해서 분석 문제를 해결해 보세요.

- Q1. mpg 데이터의 class 는 "suv", "compact" 등 자동차를 특징에 따라 일곱 종류로 분류한 변수입니다. 어떤 차종의 연비가 높은지 비교해보려고 합니다. class 별 cty 평균을 구해보세요.
- Q2. 앞 문제의 출력 결과는 class 값 알파벳 순으로 정렬되어 있습니다. 어떤 차종의 도시 연비가 높은지 쉽게 알아볼 수 있도록 cty 평균이 높은 순으로 정렬해 출력하세요.
- Q3. 어떤 회사 자동차의 hwy(고속도로 연비)가 가장 높은지 알아보려고 합니다. hwy 평균이 가장 높은 회사 세 곳을 출력하세요.
- Q4. 어떤 회사에서 "compact"(경차) 차종을 가장 많이 생산하는지 알아보려고 합니다. 각 회사별 "compact" 차종 수를 내림차순으로 정렬해 출력하세요.

힌트

- Q1. `group_by()`를 이용해 `class` 별로 나눈 뒤 `summarise()`를 이용해 `cty` 평균을 구하면 됩니다.
- Q2. 앞에서 만든 코드를 `%>%`로 연결하고 내림차순으로 정렬하는 코드를 추가하면 됩니다.
- Q3. 2번 문제와 같은 절차로 코드를 구성하고, 일부만 출력하도록 `head()`를 추가하면 됩니다.
- Q4. `filter()`를 이용해 `"compact"` 차종만 남긴 후 회사별 자동차 수를 구하면 됩니다. 자동차 수는 데이터가 몇 행으로 구성되는지 빈도를 구하면 알 수 있습니다. 빈도는 `n()`을 이용해 구할 수 있습니다.

정답

Q1. mpg 데이터의 class는 "suv", "compact" 등 자동차를 특징에 따라 일곱 종류로 분류한 변수입니다. 어떤 차종의 연비가 높은지 비교해보려고 합니다. class별 cty 평균을 구해보세요.

```
mpg <- as.data.frame(ggplot2::mpg)  # mpg 데이터 불러오기
```

```
mpg %>%
```

```
  group_by(class) %>%                # class 별 분리
```

```
  summarise(mean_cty = mean(cty))    # cty 평균 구하기
```

```
## # A tibble: 7 x 2
```

```
##       class mean_cty
```

```
##       <chr>   <dbl>
```

```
## 1 2seater 15.40000
```

```
## 2 compact 20.12766
```

```
## 3 midsize 18.75610
```

```
## 4 minivan 15.81818
```

```
## 5 pickup 13.00000
```

```
## 6 subcompact 20.37143
```

```
## 7 suv 13.50000
```

Q2. 앞 문제의 출력 결과는 `class` 값 알파벳 순으로 정렬되어 있습니다. 어떤 차종의 도시 연비가 높은지 쉽게 알아볼 수 있도록 `cty` 평균이 높은 순으로 정렬해 출력하세요.

```
mpg %>%  
  group_by(class) %>%                                # class 별 분리  
  summarise(mean_cty = mean(cty)) %>%                 # cty 평균 구하기  
  arrange(desc(mean_cty))                             # 내림차순 정렬하기  
  
## # A tibble: 7 x 2  
##       class mean_cty  
##   <chr>     <dbl>  
## 1 subcompact 20.37143  
## 2 compact    20.12766  
## 3 midsize    18.75610  
## 4 minivan    15.81818  
## 5 2seater    15.40000  
## 6 suv        13.50000  
## 7 pickup     13.00000
```

Q3. 어떤 회사 자동차의 hwy(고속도로 연비)가 가장 높은지 알아보려고 합니다. hwy 평균이 가장 높은 회사 세 곳을 출력하세요.

```
mpg %>%  
  group_by(manufacturer) %>%  
  summarise(mean_hwy = mean(hwy)) %>%  
  arrange(desc(mean_hwy)) %>%  
  head(3)  
  
## # A tibble: 3 x 2  
##   manufacturer mean_hwy  
##   <chr>      <dbl>  
## 1      honda 32.55556  
## 2 volkswagen 29.22222  
## 3    hyundai 26.85714
```


Q4. 어떤 회사에서 "compact"(경차) 차종을 가장 많이 생산하는지 알아보려고 합니다. 각 회사별 "compact" 차종 수를 내림차순으로 정렬해 출력하세요.

```
mpg %>%  
  filter(class == "compact") %>% # compact 추출  
  group_by(manufacturer) %>%     # manufacturer 별 분리  
  summarise(count = n()) %>%    # 빈도 구하기  
  arrange(desc(count))          # 내림차순 정렬  
  
## # A tibble: 5 x 2  
##   manufacturer count  
##   <chr> <int>  
## 1      audi      15  
## 2 volkswagen    14  
## 3     toyota     12  
## 4     subaru      4  
## 5     nissan      2
```

06-7. 데이터 합치기

가로로 합치기

id	midterm		id	final		id	midterm	final
1	60	+	1	70	=	1	60	70
2	80		2	83		2	80	83
3	70		3	65		3	70	65

가로로 합치기

세로로 합치기

id	test		id	test		id	test
1	60	+	4	70	=	1	60
2	80		5	83		2	80
3	70		6	65		3	70
						4	70
						5	83
						6	65

세로로 합치기

가로로 합치기

데이터 생성

```
# 중간고사 데이터 생성
test1 <- data.frame(id = c(1, 2, 3, 4, 5),
                    midterm = c(60, 80, 70, 90, 85))

# 기말고사 데이터 생성
test2 <- data.frame(id = c(1, 2, 3, 4, 5),
                    final = c(70, 83, 65, 95, 80))
```

test1 # test1 출력

##	id	midterm
## 1	1	60
## 2	2	80
## 3	3	70
## 4	4	90
## 5	5	85

test2 # test2 출력

##	id	final
## 1	1	70
## 2	2	83
## 3	3	65
## 4	4	95
## 5	5	80

id 기준으로 합치기

```
total <- left_join(test1, test2, by = "id") # id 기준으로 합쳐 total 에 할당  
total                                     # total 출력
```

```
##   id midterm final  
## 1   1      60    70  
## 2   2      80    83  
## 3   3      70    65  
## 4   4      90    95  
## 5   5      85    80
```

[주의] by에 변수명을 지정할 때 변수명 앞 뒤에 겹따옴표 입력

다른 데이터 활용해 변수 추가하기

반별 담임교사 명단 생성

```
name <- data.frame(class = c(1, 2, 3, 4, 5),  
                    teacher = c("kim", "lee", "park", "choi", "jung"))
```

name

```
##   class teacher  
## 1     1     kim  
## 2     2     lee  
## 3     3    park  
## 4     4    choi  
## 5     5    jung
```

class 기준 합치기

```
exam_new <- left_join(exam, name, by = "class")
```

```
exam_new
```

```
##      id class math english science teacher
##  1     1     1   50      98      50      kim
##  2     2     1   60      97      60      kim
##  3     3     1   45      86      78      kim
##  4     4     1   30      98      58      kim
##  5     5     2   25      80      65      lee
##  6     6     2   50      89      98      lee
##  7     7     2   80      90      45      lee
##  8     8     2   90      78      25      lee
##  9     9     3   20      98      15      park
## 10    10     3   50      98      45      park
## 11    11     3   65      65      65      park
## 12    12     3   45      85      32      park
## 13    13     4   46      98      65      choi
## 14    14     4   48      87      12      choi
## 15    15     4   75      56      78      choi
## 16    16     4   58      98      65      choi
## 17    17     5   65      68      98      jung
## 18    18     5   80      78      90      jung
## 19    19     5   89      68      87      jung
## 20    20     5   78      83      58      jung
```

세로로 합치기

데이터 생성

학생 1~5 번 시험 데이터 생성

```
group_a <- data.frame(id = c(1, 2, 3, 4, 5),  
                      test = c(60, 80, 70, 90, 85))
```

학생 6~10 번 시험 데이터 생성

```
group_b <- data.frame(id = c(6, 7, 8, 9, 10),  
                      test = c(70, 83, 65, 95, 80))
```


group_a *# group_a 출력*

##	id	test
----	----	------

## 1	1	60
------	---	----

## 2	2	80
------	---	----

## 3	3	70
------	---	----

## 4	4	90
------	---	----

## 5	5	85
------	---	----

group_b *# group_b 출력*

##	id	test
----	----	------

## 1	6	70
------	---	----

## 2	7	83
------	---	----

## 3	8	65
------	---	----

## 4	9	95
------	---	----

## 5	10	80
------	----	----

세로로 합치기

```
group_all <- bind_rows(group_a, group_b)  # 데이터 합쳐서 group_all 에 할당  
group_all                                # group_all 출력
```

```
##      id test  
## 1     1   60  
## 2     2   80  
## 3     3   70  
## 4     4   90  
## 5     5   85  
## 6     6   70  
## 7     7   83  
## 8     8   65  
## 9     9   95  
## 10    10   80
```

혼자서 해보기

mpg 데이터를 이용해서 분석 문제를 해결해 보세요.

mpg 데이터의 f1 변수는 자동차에 사용하는 연료(fuel)를 의미합니다. 아래는 자동차 연료별 가격을 나타낸 표입니다.

f1	연료 종류	가격(갤런당 USD)
c	CNG	2.35
d	diesel	2.38
e	ethanol E85	2.11
p	premium	2.76
r	regular	2.22

우선 이 정보를 이용해서 연료와 가격으로 구성된 데이터 프레임을 만들어 보세요.

```
fuel <- data.frame(f1 = c("c", "d", "e", "p", "r"),  
                  price_f1 = c(2.35, 2.38, 2.11, 2.76, 2.22),  
                  stringsAsFactors = F)  
  
fuel # 출력
```

```
##    f1 price_f1
## 1  c      2.35
## 2  d      2.38
## 3  e      2.11
## 4  p      2.76
## 5  r      2.22
```

- Q1. mpg 데이터에는 연료 종류를 나타낸 f1 변수는 있지만 연료 가격을 나타낸 변수는 없습니다. 위에서 만든 fue1 데이터를 이용해서 mpg 데이터에 price_f1(연료 가격) 변수를 추가하세요.
- Q2. 연료 가격 변수가 잘 추가됐는지 확인하기 위해서 model, f1, price_f1 변수를 추출해 앞부분 5 행을 출력해 보세요.

힌트

Q1. `left_join()`을 이용해서 `mpg` 데이터에 `fue1` 데이터를 합치면 됩니다. 두 데이터에 공통으로 들어있는 변수를 기준으로 삼아야 합니다.

Q2. `select()`와 `head()`를 조합하면 됩니다.

정답

Q1. mpg 데이터에는 연료 종류를 나타낸 f1 변수는 있지만 연료 가격을 나타낸 변수는 없습니다. 위에서 만든 fuel 데이터를 이용해서 mpg 데이터에 price_f1(연료 가격) 변수를 추가하세요.

```
mpg <- as.data.frame(ggplot2::mpg)      # mpg 데이터 불러오기  
mpg <- left_join(mpg, fuel, by = "f1")   # mpg 에 연료 가격 변수 추가
```

Q2. 연료 가격 변수가 잘 추가됐는지 확인하기 위해서 `model`, `fl`, `price_fl` 변수를 추출해 앞부분 5행을 출력해 보세요.

```
mpg %>%  
  select(model, fl, price_fl) %>%      # model, fl, price_fl 추출  
  head(5)  
  
##   model fl price_fl  
## 1    a4  p      2.76  
## 2    a4  p      2.76  
## 3    a4  p      2.76  
## 4    a4  p      2.76  
## 5    a4  p      2.76
```

정리하기

1. 조건에 맞는 데이터만 추출하기

```
exam %>% filter(english >= 80)
```

여러 조건 동시 충족

```
exam %>% filter(class == 1 & math >= 50)
```

여러 조건 중 하나 이상 충족

```
exam %>% filter(math >= 90 | english >= 90)
```

```
exam %>% filter(class %in% c(1,3,5))
```

2. 필요한 변수만 추출하기

```
exam %>% select(math)
```

```
exam %>% select(class, math, english)
```

3. 함수 조합하기, 일부만 출력하기

```
exam %>%
```

```
  select(id, math) %>%
```

```
  head(10)
```


4. 순서대로 정렬하기

```
exam %>% arrange(math)           # 오름차순 정렬
exam %>% arrange(desc(math))      # 내림차순 정렬
exam %>% arrange(class, math)     # 여러 변수 기준 오름차순 정렬
```

5. 파생변수 추가하기

```
exam %>% mutate(total = math + english + science)
```

여러 파생변수 한 번에 추가하기

```
exam %>%
  mutate(total = math + english + science,
         mean = (math + english + science)/3)
```

mutate()에 ifelse() 적용하기

```
exam %>% mutate(test = ifelse(science >= 60, "pass", "fail"))
```

추가한 변수를 dplyr 코드에 바로 활용하기

```
exam %>%
  mutate(total = math + english + science) %>%
  arrange(total)
```

6. 집단별로 요약하기

```
exam %>%  
  group_by(class) %>%  
  summarise(mean_math = mean(math))
```

각 집단별로 다시 집단 나누기

```
mpg %>%  
  group_by(manufacturer, drv) %>%  
  summarise(mean_cty = mean(cty))
```

7. 데이터 합치기

가로로 합치기

```
total <- left_join(test1, test2, by = "id")
```

세로로 합치기

```
group_all <- bind_rows(group_a, group_b)
```

분석 도전

미국 동북중부 437개 지역의 인구통계 정보를 담고 있는 `midwest` 데이터를 사용해 데이터 분석 문제를 해결해 보세요. `midwest`는 `ggplot2` 패키지에 들어 있습니다.

- 문제 1. `popadults` 는 해당 지역의 성인 인구, `poptotal` 은 전체 인구를 나타냅니다. `midwest` 데이터에 '전체 인구 대비 미성년 인구 백분율' 변수를 추가하세요.
- 문제 2. 미성년 인구 백분율이 가장 높은 상위 5 개 `county`(지역)의 미성년 인구 백분율을 출력하세요.
- 문제 3. 분류표의 기준에 따라 미성년 비율 등급 변수를 추가하고, 각 등급에 몇 개의 지역이 있는지 알아보세요.

분류	기준
large	40% 이상
middle	30% ~ 40% 미만
small	30% 미만

- 문제4. `popasian`은 해당 지역의 아시아인 인구를 나타냅니다. '전체 인구 대비 아시아인 인구 백분율' 변수를 추가하고, 하위 10개 지역의 `state`(주), `county`(지역명), 아시아인 인구 백분율을 출력하세요.

정답

문제1. `popadults`는 해당 지역의 성인 인구, `poptotal`은 전체 인구를 나타냅니다. `midwest` 데이터에 '전체 인구 대비 미성년 인구 백분율' 변수를 추가하세요.

```
# midwest 불러오기
midwest <- as.data.frame(ggplot2::midwest)

# midwest 에 백분율 변수 추가
midwest <- midwest %>%
  mutate(ratio_child = (poptotal-popadults)/poptotal*100)
```

문제2. 미성년 인구 백분율이 가장 높은 상위 5개 county(지역)의 미성년 인구 백분율을 출력하세요.

```
midwest %>%  
  arrange(desc(ratio_child)) %>% # ratio_child 내림차순 정렬  
  select(county, ratio_child) %>% # county, ratio_child 추출  
  head(5) # 상위 5 행 출력
```

```
##      county ratio_child  
## 1  ISABELLA    51.50117  
## 2 MENOMINEE    50.59126  
## 3   ATHENS     49.32073  
## 4  MECOSTA     49.05918  
## 5  MONROE      47.35818
```

문제3. 분류표의 기준에 따라 미성년 비율 등급 변수를 추가하고, 각 등급에 몇 개의 지역이 있는지 알아보세요.

```
# midwest 에 grade 변수 추가
```

```
midwest <- midwest %>%
```

```
  mutate(grade = ifelse(ratio_child >= 40, "large",  
                        ifelse(ratio_child >= 30, "middle", "small")))
```

```
# 미성년 비율 등급 빈도표
```

```
table(midwest$grade)
```

```
##
```

```
##  large middle  small
```

```
##    32    396     9
```

문제4. popasian은 해당 지역의 아시아인 인구를 나타냅니다. '전체 인구 대비 아시아인 인구 백분율' 변수를 추가하고, 하위 10개 지역의 state(주), county(지역명), 아시아인 인구 백분율을 출력하세요.

```
midwest %>%  
  mutate(ratio_asian = (popasian/poptotal)*100) %>% # 백분율 변수 추가  
  arrange(ratio_asian) %>% # 내림차순 정렬  
  select(state, county, ratio_asian) %>% # 변수 추출  
  head(10) # 상위 10 행 출력
```

```
##      state      county ratio_asian  
## 1      WI  MENOMINEE  0.00000000  
## 2      IN    BENTON  0.01059210  
## 3      IN   CARROLL  0.01594981  
## 4      OH    VINTON  0.02703190  
## 5      WI      IRON  0.03250447  
## 6      IL    SCOTT  0.05315379  
## 7      IN     CLAY  0.06071645  
## 8      MI   OSCODA  0.06375925  
## 9      OH    PERRY  0.06654625  
## 10     IL    PIATT  0.07074865
```