

# 주피터 노트북 테마 설정하기

1. cmd로 console창으로 이동
2. `pip install jupyterthemes`
3. `jt -l` (소문자 L)
4. `jt -t grade3`

```
chesterish  
grade3  
gruvboxd  
gruvboxl  
monokai  
oceans16  
onedork  
solarizedd  
solarizedl
```

## **05-1 분석하기 좋은 데이터**

# 분석하기 좋은 데이터란?



- ① 데이터 분석 목적에 맞는 데이터 모으기
- ② 측정한 값은 행(row)를 구성
- ③ 변수는 열(column)을 구성

분석하기 좋은 데이터 = 깔끔한 데이터(Tidy Data)

## **05-2 데이터 연결 기초**

# 데이터 연결하기(1)



## Concat 메서드로 데이터 연결(인덱스 유지되는 점 주의)

```
import pandas as pd

df1 = pd.read_csv('../data/concat_1.csv')
df2 = pd.read_csv('../data/concat_2.csv')
df3 = pd.read_csv('../data/concat_3.csv')
```

```
row_concat = pd.concat([df1, df2, df3])
print(row_concat)
```

	A	B	C	D
0	a0	b0	c0	d0
1	a1	b1	c1	d1
2	a2	b2	c2	d2
...				
2	a10	b10	c10	d10
3	a11	b11	c11	d11



인덱스도 그대로 유지됩니다.

# 데이터 연결하기(2)



**append 메서드로 데이터 연결**

**(연결할 데이터 프레임이 한 개일 때만!!)**

**일치하는 column 에 맞게 추가됨**

```
new_row_df = pd.DataFrame([['n1', 'n2', 'n3', 'n4']], columns=['A', 'B', 'C', 'D'])  
print(new_row_df)
```

```
df1.append(new_row_df)
```

# 데이터 연결하기(3)



## ignore\_index 인자 사용

원래의 인덱스를 무시하고 0부터 다시 지정

```
row_concat_i = pd.concat([df1, df2, df3], ignore_index=True)
print(row_concat_i)
```

	A	B	C	D
0	a0	b0	c0	d0
1	a1	b1	c1	d1
2	a2	b2	c2	d2
3	a3	b3	c3	d3
4	a4	b4	c4	d4
5	a5	b5	c5	d5

# 데이터 연결하기(4)



## 열 방향으로 데이터 연결하기

인자 axis = 1

# default는 axis = 0이고 행으로 연결된다.

```
row_concat_i = pd.concat([df1, df2, df3], ignore_index=True)
print(row_concat_i)
```

	A	B	C	D
0	a0	b0	c0	d0
1	a1	b1	c1	d1
2	a2	b2	c2	d2
3	a3	b3	c3	d3
4	a4	b4	c4	d4
5	a5	b5	c5	d5



# 데이터 연결하기(5)



## 공통 열만 연결하기

```
row_concat = pd.concat([df1, df2, df3])  
print(row_concat)
```

	A	B	C	D	E	F	G	H
0	a0	b0	c0	d0	NaN	NaN	NaN	NaN
1	a1	b1	c1	d1	NaN	NaN	NaN	NaN
2	a2	b2	c2	d2	NaN	NaN	NaN	NaN
3	a3	b3	c3	d3	NaN	NaN	NaN	NaN
0	NaN	NaN	NaN	NaN	a4	b4	c4	d4
1	NaN	NaN	NaN	NaN	a5	b5	c5	d5
2	NaN	NaN	NaN	NaN	a6	b6	c6	d6
3	NaN	NaN	NaN	NaN	a7	b7	c7	d7
0	a8	NaN	b8	NaN	NaN	c8	NaN	d8
1	a9	NaN	b9	NaN	NaN	c9	NaN	d9
2	a10	NaN	b10	NaN	NaN	c10	NaN	d10
3	a11	NaN	b11	NaN	NaN	c11	NaN	d11

# 데이터 연결하기(5)



## 공통 열만 연결하기

#inner join? 내부조인은 둘 이상의 데이터프레임에서 조건에 맞는 행을 연결하는 것입니다.

```
print(pd.concat([df1,df3], ignore_index=False, join='inner'))
```

	A	C
0	a0	c0
1	a1	c1
2	a2	c2
3	a3	c3
0	a8	b8
1	a9	b9
2	a10	b10
3	a11	b11

# 데이터 연결하기(5)



## 공통 인덱스만 연결하기

```
col_concat = pd.concat([df1, df2, df3], axis=1)
print(col_concat)
```

	A	B	C	D	E	F	G	H	A	C	F	H
0	a0	b0	c0	d0	NaN	NaN	NaN	NaN	a8	b8	c8	d8
1	a1	b1	c1	d1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	a2	b2	c2	d2	NaN	NaN	NaN	NaN	a9	b9	c9	d9
3	a3	b3	c3	d3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	a4	b4	c4	d4	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	a5	b5	c5	d5	a10	b10	c10	d10
6	NaN	NaN	NaN	NaN	a6	b6	c6	d6	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	a7	b7	c7	d7	a11	b11	c11	d11

```
print(pd.concat([df1, df3], axis=1, join='inner'))
```

	A	B	C	D	A	C	F	H
0	a0	b0	c0	d0	a8	b8	c8	d8
2	a2	b2	c2	d2	a9	b9	c9	d9

## **05-3 데이터 연결 마무리**

# merge 메서드 사용하기 (1)



## 1. 데이터 불러오기

```
person = pd.read_csv('../data/survey_person.csv')
site = pd.read_csv('../data/survey_site.csv')
survey = pd.read_csv('../data/survey_survey.csv')
visited = pd.read_csv('../data/survey_visited.csv')
```

# merge 메서드 사용하기 (2)



2. merge 메서드 => default: 내부 조인  
메서드를 사용한 데이터프레임 (site)이 merge된 DF의 왼쪽에 옴

```
o2o_merge = site.merge(visited_subset, left_on='name', right_on='site')
```

	name	lat	long
0	DR-1	-49.85	-128.57
1	DR-3	-47.15	-126.72
2	MSK-4	-48.87	-123.40

	ident	site	dated
0	619	DR-1	1927-02-08
2	734	DR-3	1939-01-07
6	837	MSK-4	1932-01-14

	name	lat	long	ident	site	dated
0	DR-1	-49.85	-128.57	619	DR-1	1927-02-08
1	DR-3	-47.15	-126.72	734	DR-3	1939-01-07
2	MSK-4	-48.87	-123.40	837	MSK-4	1932-01-14

# merge 메서드 사용하기 (3)



## 3. left\_on 과 right\_on에 여러 개의 값 전달 가능

```
ps_vs = ps.merge(vs, left_on=['ident', 'taken', 'quant', 'reading'],  
                 right_on=['person', 'ident', 'quant', 'reading'])
```

indent – person  
taken – indent

.

.

다음과 같이 대응

## **07-1 열과 피벗**





데이터의 열 이름이 어떤 값을 의미하면 열의 폭의 넓은 경우가 많음

	religion	<\$10k	\$10-20k	\$20-30k	\$30-40k	\$40-50k	\$50-75k	#
0	Agnostic	27	34	60	81	76	137	
1	Atheist	12	27	37	52	35	70	
2	Buddhist	27	21	30	34	33	58	
3	Catholic	418	617	732	670	638	1116	
4	Don' t know/refused	15	14	15	11	10	35	
		\$75-100k	\$100-150k	>150k	Don't know/refused			
0		122	109	84		96		
1		73	59	74		76		
2		62	39	53		54		
3		949	792	633		1489		
4		21	17	18		116		



## 데이터 재구조화 (Reshaping data by **melt**)



**pd.melt(data, id\_vars, var\_name, value\_name)**

*Original data*

cust_ID	prd_CD	pch_amt	pch_cnt
C_001	P_001	100	1
C_001	P_002	200	2
C_002	P_001	300	3
C_002	P_002	400	4

*melt*

*Melted data*

cust_ID	prd_CD	variable	value
C_001	P_001	pch_amt	100
C_001	P_002	pch_amt	200
C_002	P_001	pch_amt	300
C_002	P_002	pch_amt	400
C_001	P_001	pch_cnt	1
C_001	P_002	pch_cnt	2
C_002	P_001	pch_cnt	3
C_002	P_002	pch_cnt	4

# melt 메서드(1)



## 1개의 열 고정하고 나머지 열을 행으로 바꾸기(Pivot)

```
import pandas as pd
pew = pd.read_csv('../data/pew.csv')
print(pew.head( ))
```

	religion	<\$10k	\$10-20k	\$20-30k	\$30-40k	\$40-50k	\$50-75k	\
0	Agnostic	27	34	60	81	76	137	
1	Atheist	12	27	37	52	35	70	
2	Buddhist	27	21	30	34	33	58	
3	Catholic	418	617	732	670	638	1116	
4	Don't know/refused	15	14	15	11	10	35	

```
pew_long = pd.melt(pew, id_vars='religion')
print(pew_long.head( ))
```

	religion	variable	value
0	Agnostic	<\$10k	27
1	Atheist	<\$10k	12
2	Buddhist	<\$10k	27
3	Catholic	<\$10k	418
4	Don't know/refused	<\$10k	15

# melt 메서드(2)



2개 이상의 열을 고정하고 나머지 열을 행으로 바꾸기  
year, artist, track, time, date.entered 컬럼을 제외한 부분을 행으로 바꾸기

```
billboard = pd.read_csv('../data/billboard.csv')  
print billboard.iloc[0:5, 0:16]
```

	year	artist	track	time	date.entered	wk1	wk2	₩
0	2000	2 Pac	Baby Don't Cry (Keep...	4:22	2000-02-26	87	82.0	
1	2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	91	87.0	
2	2000	3 Doors Down	Kryptonite	3:53	2000-04-08	81	70.0	
3	2000	3 Doors Down	Loser	4:24	2000-10-21	76	76.0	
4	2000	504 Boyz	Wobble Wobble	3:35	2000-04-15	57	34.0	

	wk3	wk4	wk5	wk6	wk7	wk8	wk9	wk10	wk11
0	72.0	77.0	87.0	94.0	99.0	NaN	NaN	NaN	NaN
1	92.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	68.0	67.0	66.0	57.0	54.0	53.0	51.0	51.0	51.0
3	72.0	69.0	67.0	65.0	55.0	59.0	62.0	61.0	61.0
4	25.0	17.0	17.0	31.0	36.0	49.0	53.0	57.0	64.0

# melt 메서드(2)



`id_vars` 인자에는 고정할 column들을 넣는다.

`var_name` 인자에는 column들의 이름들이 들어갈 새 column의 이름을 넣는다.

`value_name` 인자에는 원래 value였던 자료들이 들어갈 새 column의 이름을 넣는다.

고정 column

```
billboard_long = pd.melt(billboard, id_vars=['year', 'artist', 'track', 'time', 'date.entered'],  
                        var_name='week', value_name='rating')  
print(billboard_long.head())
```

	year	artist	track	time	date.entered	week	rating
0	2000	2 Pac	Baby Don't Cry (Keep...	4:22	2000-02-26	wk1	87.0
1	2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	wk1	91.0
2	2000	3 Doors Down	Kryptonite	3:53	2000-04-08	wk1	81.0
3	2000	3 Doors Down	Loser	4:24	2000-10-21	wk1	76.0
4	2000	504 Boyz	Wobble Wobble	3:35	2000-04-15	wk1	57.0

## **07-2 열 이름 관리하기**

# ebola 데이터 집합 살펴보기



Cases : 발병 / Deaths : 죽음

```
ebola = pd.read_csv('../data/country_timeseries.csv')  
print(ebola.columns)
```

```
Index(['Date', 'Day', 'Cases_Guinea', 'Cases_Liberia', 'Cases_SierraLeone',  
      'Cases_Nigeria', 'Cases_Senegal', 'Cases_UnitedStates', 'Cases_Spain',  
      'Cases_Mali', 'Deaths_Guinea', 'Deaths_Liberia', 'Deaths_SierraLeone',  
      'Deaths_Nigeria', 'Deaths_Senegal', 'Deaths_UnitedStates',  
      'Deaths_Spain', 'Deaths_Mali'],  
      dtype='object')
```



# ebola 데이터 집합 살펴보기



‘Date’ 와 ‘Day’ Column을 제외하고 녹여버리기~!

```
ebola_long = pd.melt(ebola, id_vars=['Date', 'Day'])  
print(ebola_long.head())
```

variable이라는 하나의 열이 여러 의미를 가지게 되었다!!!

	Date	Day	<u>variable</u>	<u>value</u>
0	1/5/2015	289	Cases_Guinea	2776.0
1	1/4/2015	288	Cases_Guinea	2775.0
2	1/3/2015	287	Cases_Guinea	2769.0
3	1/2/2015	286	Cases_Guinea	NaN
4	12/31/2014	284	Cases_Guinea	2730.0



# split 메서드로 열 이름 분리하기



## < variable\_split : 자료형은 Series >

0	[Cases, Guinea]
1	[Cases, Guinea]
2	[Cases, Guinea]
3	[Cases, Guinea]
4	[Cases, Guinea]
...	...
1947	[Deaths, Mali]
1948	[Deaths, Mali]
1949	[Deaths, Mali]
1950	[Deaths, Mali]
1951	[Deaths, Mali]

```
status_values = variable_split.str.get(0)
country_values = variable_split.str.get(1)
print(status_values[:5])
print(country_values[:5])
```

```
0    Cases
1    Cases
2    Cases
3    Cases
4    Cases
Name: variable, dtype: object
0    Guinea
1    Guinea
2    Guinea
3    Guinea
4    Guinea
Name: variable, dtype: object
```

# split 메서드로 열 이름 분리하기



각 component의 위치에 해당하는 원소를 추출한다.

component : lists, tuples or strings

```
pandas.Series.str.get
```

```
Series.str.get(self, i) ¶
```

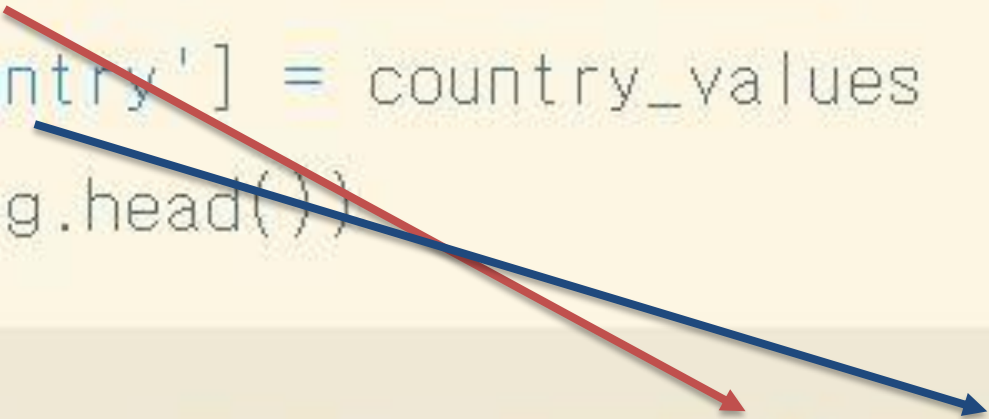
Extract element from each component at specified position.

Extract element from lists, tuples, or strings in each element in the Series/Index.

# 정돈된 열을 다시 DF에 추가하기



```
ebola_long['status'] = status_values  
ebola_long['country'] = country_values  
print(ebola_long.head())
```



A diagram with two arrows originates from the code above. A red arrow points from the 'status' column in the code to the 'status' column in the table. A blue arrow points from the 'country' column in the code to the 'country' column in the table.

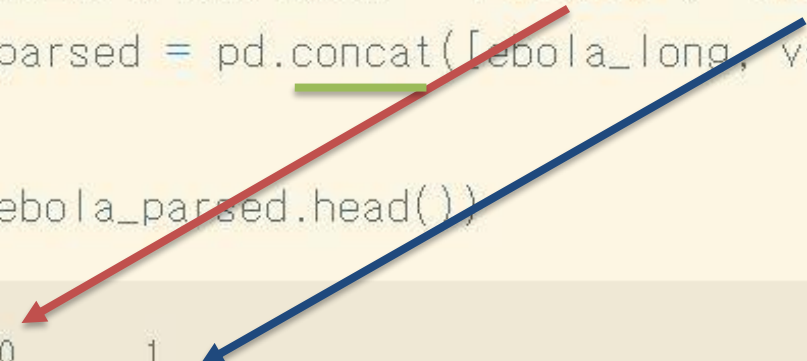
	Date	Day	variable	value	status	country
0	1/5/2015	289	Cases_Guinea	2776.0	Cases	Guinea
1	1/4/2015	288	Cases_Guinea	2775.0	Cases	Guinea
2	1/3/2015	287	Cases_Guinea	2769.0	Cases	Guinea
3	1/2/2015	286	Cases_Guinea	NaN	Cases	Guinea
4	12/31/2014	284	Cases_Guinea	2730.0	Cases	Guinea

# #다른 방법



```
variable_split = ebola_long.variable.str.split('_', expand=True)
print(variable_split.head())
variable_split.columns = ['status', 'country'] #column 추가
ebola_parsed = pd.concat([ebola_long, variable_split], axis=1)
print(ebola_parsed.head())
```

열 방향 추가!!



	0	1
0	Cases	Guinea
1	Cases	Guinea
2	Cases	Guinea
3	Cases	Guinea
4	Cases	Guinea

	Date	Day	variable	value	status	country	status	country
0	1/5/2015	289	Cases_Guinea	2776.0	Cases	Guinea	Cases	Guinea
1	1/4/2015	288	Cases_Guinea	2775.0	Cases	Guinea	Cases	Guinea
2	1/3/2015	287	Cases_Guinea	2769.0	Cases	Guinea	Cases	Guinea
3	1/2/2015	286	Cases_Guinea	NaN	Cases	Guinea	Cases	Guinea
4	12/31/2014	284	Cases_Guinea	2720.0	Cases	Guinea	Cases	Guinea

## **07-3 여러 열을 하나로 정리하기**

# 기상 데이터 집합 살펴보기



```
weather = pd.read_csv('../data/weather.csv')
print(weather.iloc[:5, :11])
```

	id	year	month	element	d1	d2	d3	d4	d5	d6	d7
0	MX17004	2010	1	tmax	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	MX17004	2010	1	tmin	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	MX17004	2010	2	tmax	NaN	27.3	24.1	NaN	NaN	NaN	NaN
3	MX17004	2010	2	tmin	NaN	14.4	14.4	NaN	NaN	NaN	NaN
4	MX17004	2010	3	tmax	NaN	NaN	NaN	NaN	32.1	NaN	NaN

날짜 열에 각 월별  
최고, 최저 온도 데이터 저장

```
weather_melt = pd.melt(weather, id_vars=['id', 'year', 'month', 'element'],
                        var_name='day', value_name='temp')
print(weather_melt.head())
```

	id	year	month	element	day	temp
0	MX17004	2010	1	tmax	d1	NaN
1	MX17004	2010	1	tmin	d1	NaN
2	MX17004	2010	2	tmax	d1	NaN
3	MX17004	2010	2	tmin	d1	NaN



# 기상 데이터의 여러 열을 하나로 정리하기



## pivot\_table 메소드

```
weather_tidy = weather_melt.pivot_table(  
    index=['id', 'year', 'month', 'day'],  
    columns='element',  
    values='temp'  
)  
  
weather_tidy
```

id	year	month	day	element	tmax	tmin
MX17004	2010	1	d30		27.8	14.5
			2	d11	29.7	13.4
				d2	27.3	14.4
				d23	29.9	10.7
		3		d3	24.1	14.4
			d10		34.5	16.8
				d16	31.1	17.6
				d5	32.1	14.2
		4	d27		36.3	16.7
		5	d27		33.2	18.2
		6	d17		28.0	17.5
				d29	30.1	18.0
		7	d3		28.6	17.5
				d14	29.9	16.5
		8	d23		26.4	15.0
				d5	29.6	15.8
				d29	28.0	15.3

# 기상 데이터의 여러 열을 하나로 정리하기



## reset\_index 메소드

```
1 weather_tidy_flat = weather_tidy.reset_index()  
2 print(weather_tidy_flat.head())
```

element	id	year	month	day	tmax	tmin
0	MX17004	2010	1	d30	27.8	14.5
1	MX17004	2010	2	d11	29.7	13.4
2	MX17004	2010	2	d2	27.3	14.4
3	MX17004	2010	2	d23	29.9	10.7
4	MX17004	2010	2	d3	24.1	14.4



## **07-4 중복 데이터 처리하기**

# 빌보드 차트 데이터 집합 살펴보기



```
1 import pandas as pd
2
3 billboard = pd.read_csv('../data/billboard.csv')
4 billboard_long = pd.melt(billboard, id_vars=['year', 'artist', 'track', 'time', 'date.entered'],
5                           var_name='week', value_name='rating')
6
7 print(billboard_long.shape)
8 billboard_long.head(10)
```

(24092, 7)

	year	artist	track	time	date.entered	week	rating
0	2000	2 Pac	Baby Don't Cry (Keep...	4:22	2000-02-26	wk1	87.0
1	2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	wk1	91.0
2	2000	3 Doors Down	Kryptonite	3:53	2000-04-08	wk1	81.0
3	2000	3 Doors Down	Loser	4:24	2000-10-21	wk1	76.0
4	2000	504 Boyz	Wobble Wobble	3:35	2000-04-15	wk1	57.0
5	2000	98^0	Give Me Just One Nig...	3:24	2000-08-19	wk1	51.0
6	2000	A*Teens	Dancing Queen	3:44	2000-07-08	wk1	97.0
7	2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	wk1	84.0
8	2000	Aalivah	Trv Again	4:03	2000-03-18	wk1	59.0

# 빌보드 차트 데이터 집합 살펴보기



# year, artist, track, time에 중복이 많다는 것을 알 수 있다.

```
1 billboard_long[billboard_long.track == 'Loser'].head()
```

	year	artist	track	time	date.entered	week	rating
3	2000	3 Doors Down	Loser	4:24	2000-10-21	wk1	76.0
320	2000	3 Doors Down	Loser	4:24	2000-10-21	wk2	76.0
637	2000	3 Doors Down	Loser	4:24	2000-10-21	wk3	72.0
954	2000	3 Doors Down	Loser	4:24	2000-10-21	wk4	69.0
1271	2000	3 Doors Down	Loser	4:24	2000-10-21	wk5	67.0

# 빌보드 차트 – 중복 데이터 처리하기



중복이 많은 year, artist, track, time 열을 추출한다.

```
1 billboard_songs = billboard_long[['year', 'artist', 'track', 'time']]
2 print(billboard_songs.shape)
```

(24092, 4)

**중복된 행을 제거!**

```
1 billboard_songs = billboard_songs.drop_duplicates()
2 print(billboard_songs[billboard_songs.track == 'Loser'].head())
3 print(billboard_songs.shape)
```

	year		artist	track	time
3	2000	3	Doors Down	Loser	4:24

(317, 4)

# 빌보드 차트 – 중복 데이터 처리하기



‘id’라는 새로운 column을 만들!

```
1 billboard_songs['id'] = range(len(billboard_songs))  
2 print(billboard_songs.head(n=10))
```

	year	artist	track	time	id
0	2000	2 Pac	Baby Don't Cry (Keep...	4:22	0
1	2000	2Gether	The Hardest Part Of ...	3:15	1
2	2000	3 Doors Down	Kryptonite	3:53	2
3	2000	3 Doors Down	Loser	4:24	3
4	2000	504 Boyz	Wobble Wobble	3:35	4
5	2000	98^0	Give Me Just One Nig...	3:24	5
6	2000	A*Teens	Dancing Queen	3:44	6
7	2000	Aaliyah	I Don't Wanna	4:15	7
8	2000	Aaliyah	Try Again	4:03	8
9	2000	Adams, Yolanda	Open My Heart	5:30	9

# 빌보드 차트 - 중복 데이터 처리하기



```
1 print billboard_ratings.shape
2
3 billboard_ratings = billboard_long.merge( billboard_songs,
4                                           on=['year', 'artist', 'track', 'time'])
5
```

billboard\_long에 billboard\_songs를 merge한다.

↳ 인자 on을 기준으로 merge한다.

(24092, 8)

```
1 print billboard_ratings.shape
2 billboard_ratings.head(20)
```

(24092, 8)

	year	artist	track	time	date.entered	week	rating	id
0	2000	2 Pac	Baby Don't Cry (Keep...	4:22	2000-02-26	wk1	87.0	0
1	2000	2 Pac	Baby Don't Cry (Keep...	4:22	2000-02-26	wk2	82.0	0
2	2000	2 Pac	Baby Don't Cry (Keep...	4:22	2000-02-26	wk3	72.0	0
3	2000	2 Pac	Baby Don't Cry (Keep...	4:22	2000-02-26	wk4	77.0	0
4	2000	2 Pac	Baby Don't Cry (Keep...	4:22	2000-02-26	wk5	87.0	0

## **08-1 자료형 다루기**

# 자료형 변환 — astype 메서드



카테고리 자료형인 tips['sex']를 astype 메서드를 이용하여 문자열로 변환

```
tips['sex_str'] = tips['sex'].astype(str)
```

```
print(tips.dtypes)
```

total_bill	float64
tip	float64
sex	category
smoker	category
day	category
time	category
size	int64
sex_str	object

dtype: object



# 잘못 입력한 데이터 처리하기



missing 처리된 1, 3, 5, 7행의 데이터 처리하기

```
tips_sub_miss = tips.head(10)
tips_sub_miss.loc[[1, 3, 5, 7], 'total_bill'] = 'missing'
print(tips_sub_miss)
```

	total_bill	tip	sex	smoker	day	time	size	sex_str
0	16.99	1.01	Female	No	Sun	Dinner	2	Female
1	missing	1.66	Male	No	Sun	Dinner	3	Male
2	21.01	3.50	Male	No	Sun	Dinner	3	Male

# 잘못 입력한 데이터



missing으로 인하여 total\_bill(float)의 값이 문자열로 인식

```
print(tips_sub_miss.dtypes)
```

total_bill	object
------------	--------

tip	float64
-----	---------

sex	category
-----	----------

smoker	category
--------	----------

day	category
-----	----------

time	category
------	----------

size	int64
------	-------

sex_str	object
---------	--------

dtype: object

# 잘못 입력한 데이터 처리하기



total\_bill column을 float type으로 바꾸어보자.

```
1 | tips_sub_miss['total_bill'].astype(float)
```

# 잘못 입력한 데이터 처리하기 — to\_numeric 메서드



errors 인자에 설정할 수 있는 값

- raise : 숫자로 변환할 수 없는 값이 있으면 오류 발생
- coerce : 숫자로 변환할 수 없는 값을 누락값으로 지정
- ignore : 아무 작업도 하지 않음

```
1 tips_sub_miss['total_bill'] = pd.to_numeric(  
2     tips_sub_miss['total_bill'],  
3     errors='ignore')  
4  
5 print(tips_sub_miss.dtypes)
```

total_bill	object
tip	float64
sex	category
smoker	category
day	category

# 잘못 입력한 데이터 처리하기 — to\_numeric 메서드



errors = 'coerce'      숫자로 변환할 수 없는 값을 누락값으로 지정

```
1 tips_sub_miss['total_bill'] = pd.to_numeric(  
2     tips_sub_miss['total_bill'],  
3     errors='coerce')  
4  
5 print(tips_sub_miss.dtypes)
```

total_bill	float64
tip	float64

downcast float 64 → float32

```
1 tips_sub_miss['total_bill'] = pd.to_numeric( tips_sub_miss['total_bill'],  
2                                             errors='coerce',  
3                                             downcast='float')  
4  
5 print(tips_sub_miss.dtypes)
```

total_bill	float32
tip	float64

## **08-2 카테고리 자료형**

# 카테고리 자료형?



‘카테고리’ : 유한한 범위의 값만 가질 수 특수한 자료형

- 용량과 속도 면에서 매우 효율적
- 주로 동일한 문자열이 반복되어 데이터를 구성할 때 사용

```
1 tips['sex'] = tips['sex'].astype('str')
2 print(tips.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 8 columns):
total_bill    244 non-null float64
tip           244 non-null float64
sex           244 non-null object
smoker        244 non-null category
day           244 non-null category
time          244 non-null category
size          244 non-null int64
sex_str       244 non-null object
dtypes: category(3), float64(2), int64(1), object(2)
memory usage: 10.7+ KB
None
```

```
1 tips['sex'] = tips['sex'].astype('category')
2 print(tips.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 8 columns):
total_bill    244 non-null float64
tip           244 non-null float64
sex           244 non-null category
smoker        244 non-null category
day           244 non-null category
time          244 non-null category
size          244 non-null int64
sex_str       244 non-null object
dtypes: category(4), float64(2), int64(1), object(1)
memory usage: 9.1+ KB
None
```