DB에 데이터를 넣어, 이를 활용하여 머신러닝을 구현한다.

학습 목표

- DB의 데이터를 넣고, sql로 불러와 시각화를 해 본다.
- 데이터를 불러오고 이를 활용하여 머신러닝을 수행한다.

In [83]:

```
import sqlite3 as sql
import pandas as pd
import matplotlib.pyplot as plt
```

01 데이터 가져오기

- https://github.com/alanjones2/dataviz/raw/master/londonweather.csv (https://github.com/alanjones2/dataviz/raw/master/londonweather.csv)
- 런던 날씨 데이터 셋

In [84]:

```
data_url = 'https://github.com/alanjones2/dataviz/raw/master/londonweather.csv'
weather = pd.read_csv(data_url)
print(weather.shape)
print(weather.info())
weather.head()
```

```
(748, 6)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 748 entries, 0 to 747
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Year    748 non-null    int64  
 1   Month   748 non-null    int64  
 2   Tmax    748 non-null    float64
 3   Tmin    748 non-null    float64
 4   Rain    748 non-null    float64
 5   Sun     748 non-null    float64
dtypes: float64(4), int64(2)
memory usage: 35.2 KB
None
```

Out[84]:

	Year	Month	Tmax	Tmin	Rain	Sun
0	1957	1	8.7	2.7	39.5	53.0
1	1957	2	9.0	2.9	69.8	64.9
2	1957	3	13.9	5.7	25.4	96.7
3	1957	4	14.2	5.2	5.7	169.6
4	1957	5	16.2	6.5	21.3	195.0

- 매년 매월의 섭씨(최고, 최소), 강우량(Rain), 한달의 총 일조시간(Sun)

02 DataFrame를 db로 이전

In [85]:

```
conn = sql.connect('weather.db')
```

In [86]:

```
# 테이블 존재한다면 삭제
conn.execute("DROP TABLE weather")
print("Table dropped... ")

# 데이터 베이스 commit
conn.commit()
```

```
Table dropped...
```

In [87]:

```
weather.to_sql('weather', conn)
conn.close()
```

02 DB로부터 데이터 가져오기

In [88]:

```
conn = sql.connect('weather.db')
weather = pd.read_sql('SELECT * FROM weather', conn)
weather.head()
```

Out[88]:

	index	Year	Month	Tmax	Tmin	Rain	Sun
0	0	1957	1	8.7	2.7	39.5	53.0
1	1	1957	2	9.0	2.9	69.8	64.9
2	2	1957	3	13.9	5.7	25.4	96.7
3	3	1957	4	14.2	5.2	5.7	169.6
4	4	1957	5	16.2	6.5	21.3	195.0

03 내가 원하는 조건을 만족하는 것 가져오기

In [89]:

```
weather.Year.describe()
```

Out[89]:

```
count     748.000000
mean     1987.668449
std        18.006907
min      1957.000000
25%      1972.000000
50%      1988.000000
75%      2003.000000
max      2019.000000
Name: Year, dtype: float64
```

In [90]:

```
y2015 = pd.read_sql('SELECT * FROM weather WHERE Year == 2015', conn)
y2015.head()
```

Out[90]:

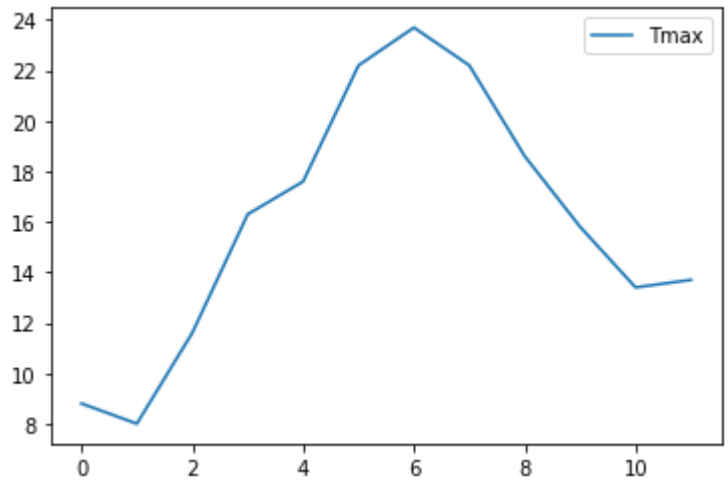
	index	Year	Month	Tmax	Tmin	Rain	Sun
0	696	2015	1	8.8	1.6	63.4	62.0
1	697	2015	2	8.0	1.8	38.6	63.9
2	698	2015	3	11.6	4.1	24.0	140.7
3	699	2015	4	16.3	6.0	16.2	212.1
4	700	2015	5	17.6	8.8	41.6	189.0

In [91]:

```
y2015.plot(y='Tmax')
```

Out[91]:

<AxesSubplot:>



In [92]:

```
y1960 = pd.read_sql('SELECT * FROM weather WHERE Year == 1960', conn)
y1960.head()
```

Out[92]:

	index	Year	Month	Tmax	Tmin	Rain	Sun
0	36	1960	1	6.9	1.8	47.9	34.4
1	37	1960	2	7.9	1.6	48.0	80.1
2	38	1960	3	10.2	4.5	33.9	65.0
3	39	1960	4	14.3	4.6	12.4	156.1
4	40	1960	5	18.4	9.3	45.6	181.7

In [93]:

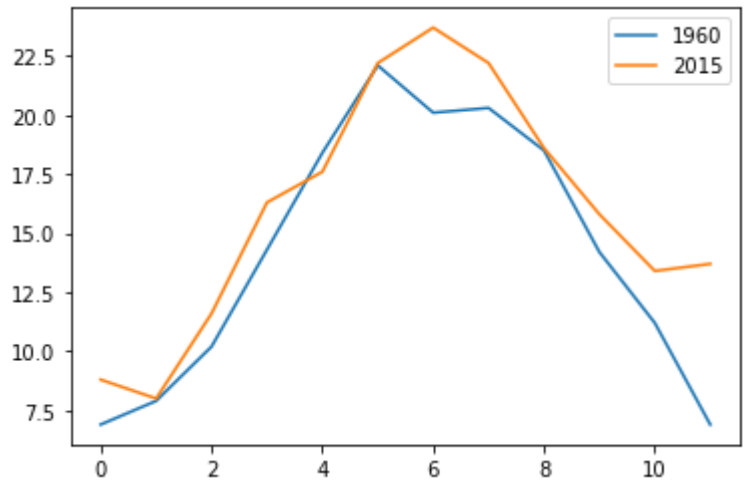
```
import matplotlib.pyplot as plt
```

In [94]:

```
plt.plot(y1960.index, y1960['Tmax'])
plt.plot(y2015.index, y2015['Tmax'])
plt.legend(['1960', '2015'])
```

Out[94]:

```
<matplotlib.legend.Legend at 0x17b21ded1c0>
```



04 일부 열 선택

In [95]:

```
high = pd.read_sql('SELECT Year,Month,Tmax FROM weather WHERE Tmax > 25', conn)
high
```

Out[95]:

	Year	Month	Tmax
0	1975	8	25.9
1	1976	6	25.5
2	1976	7	26.6
3	1976	8	25.1
4	1983	7	27.6
5	1989	7	25.8
6	1990	8	26.0
7	1994	7	26.2
8	1995	7	26.3
9	1995	8	27.0
10	1997	8	25.8
11	2003	8	26.4
12	2006	7	28.2
13	2013	7	27.0
14	2014	7	25.8
15	2018	7	28.3

In [96]:

```
query = 'SELECT Year,Month,Tmax FROM weather WHERE Tmax > 25 ORDER BY Tmax DESC'
high = pd.read_sql(query, conn)
high
```

Out[96]:

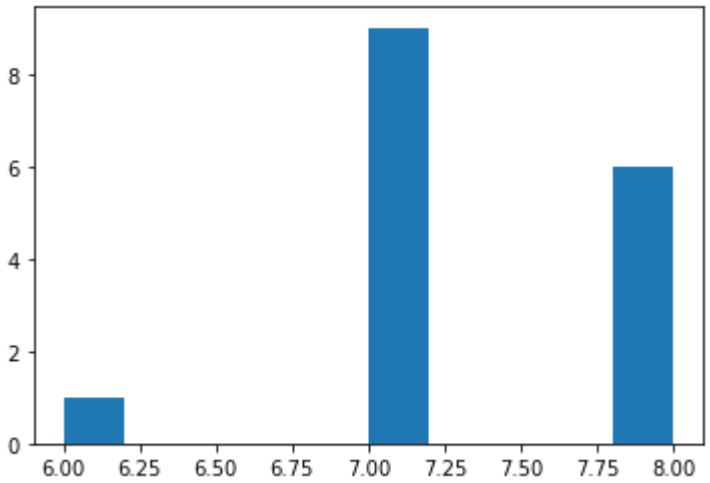
	Year	Month	Tmax
0	2018	7	28.3
1	2006	7	28.2
2	1983	7	27.6
3	1995	8	27.0
4	2013	7	27.0
5	1976	7	26.6
6	2003	8	26.4
7	1995	7	26.3
8	1994	7	26.2
9	1990	8	26.0
10	1975	8	25.9
11	1989	7	25.8
12	1997	8	25.8
13	2014	7	25.8
14	1976	6	25.5
15	1976	8	25.1

In [97]:

```
plt.hist(high['Month'])
```

Out[97]:

```
(array([1., 0., 0., 0., 0., 9., 0., 0., 0., 6.]),
 array([6. , 6.2, 6.4, 6.6, 6.8, 7. , 7.2, 7.4, 7.6, 7.8, 8. ]),
 <BarContainer object of 10 artists>)
```



지금까지의 6월달의 변화를 확인해 보자.

In [98]:

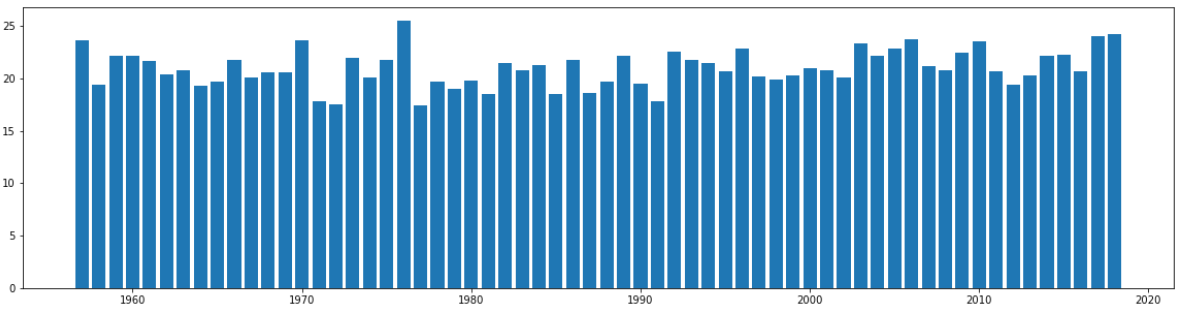
```
july = pd.read_sql('SELECT Year,Month,Tmax FROM weather WHERE month == 6', conn)
```

In [99]:

```
plt.figure(figsize=(20,5))
plt.bar(july['Year'], july['Tmax'])
```

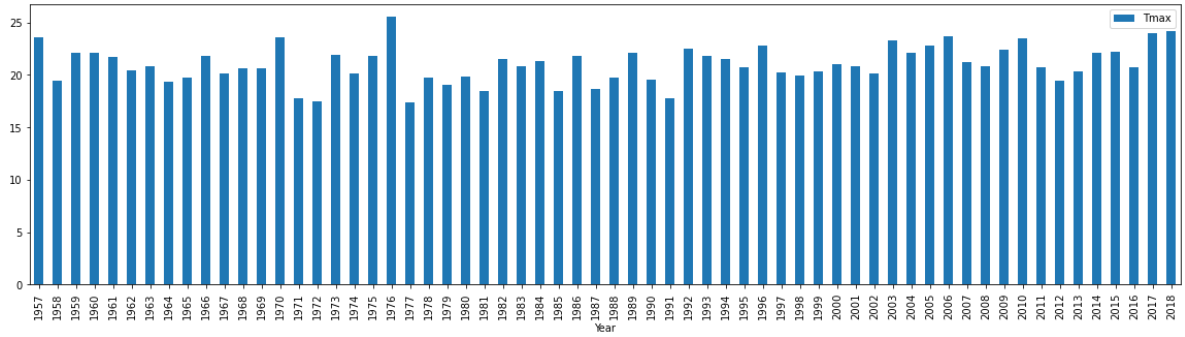
Out[99]:

```
<BarContainer object of 62 artists>
```



In [100]:

```
## 같은 내용 다른 시각화
july.plot.bar(x='Year', y='Tmax', figsize=(20,5));
```



일조량을 예측하는 머신러닝 모델 만들고 평가

In [104]:

```
weather = pd.read_sql('SELECT * FROM weather', conn)
conn.close()
weather.head()
```

Out[104]:

	index	Year	Month	Tmax	Tmin	Rain	Sun
0	0	1957	1	8.7	2.7	39.5	53.0
1	1	1957	2	9.0	2.9	69.8	64.9
2	2	1957	3	13.9	5.7	25.4	96.7
3	3	1957	4	14.2	5.2	5.7	169.6
4	4	1957	5	16.2	6.5	21.3	195.0

In [105]:

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
```

In [106]:

```
X = weather[  ['Year', 'Month', 'Tmax', 'Tmin', 'Rain']]
y = weather['Sun']

X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=17)
```

In [107]:

```
model = LinearRegression()
model.fit(X_train , y_train)

scores = cross_val_score(model, X_test, y_test, cv=5)
scores
```

Out[107]:

```
array([0.84088252, 0.89164575, 0.905941  , 0.87718712, 0.8429997 ])
```

In [108]:

```
model = RandomForestRegressor(n_estimators=300, random_state=0)
model.fit(X_train , y_train)

scores = cross_val_score(model, X_test, y_test, cv=5)
scores
```

Out[108]:

```
array([0.81373059, 0.75499834, 0.7193892 , 0.77203937, 0.81947544])
```

In [82]:

```
model = LinearRegression()
model.fit(X_train , y_train)
pred = model.predict(X_test)
pred[0:10]
```

Out[82]:

```
array([113.67270119, 185.51382423,  56.96164471,  44.83492875,
       157.45931622, 151.10205592, 179.98653517, 209.1477698 ,
       197.22508764,  34.40415088])
```

REF

https://towardsdatascience.com/python-pandas-and-sqlite-a0e2c052456f