

# Exploratory Data Analysis (EDA) 및 데이 터전처리

Day

07

# Exploratory Data Analysis (EDA) 및 데이터전처리

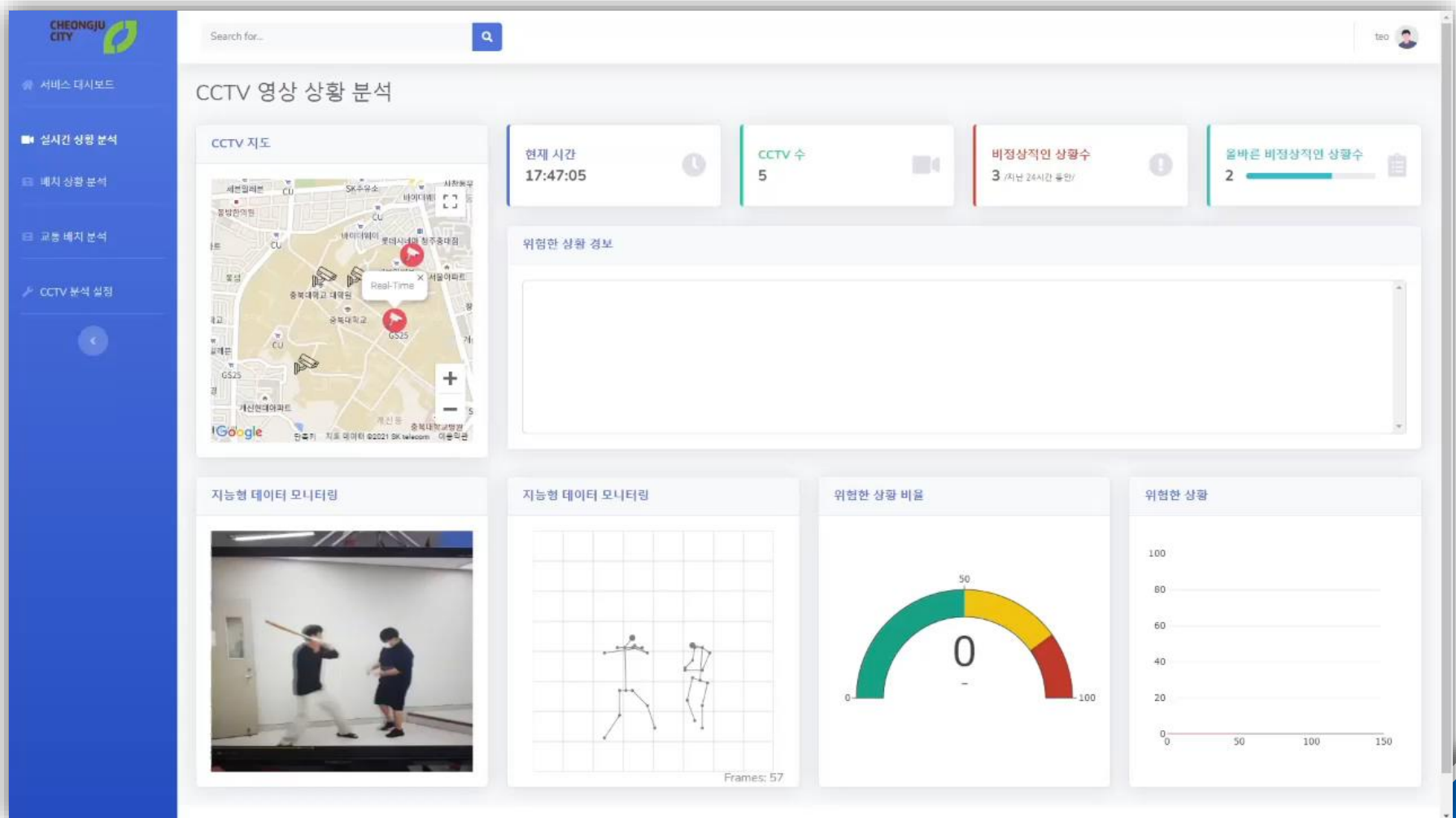


## CONTENTS

- A. EDA 필요성 및 역할
- B. 데이터 전처리
- C. 데이터 정제
- D. 데이터 변화
- E. 데이터 정규화

# Plotly Express

## ❖ Usage example



---



A

## EDA 필요성 및 역할

# EDA 필요성 및 역할

---

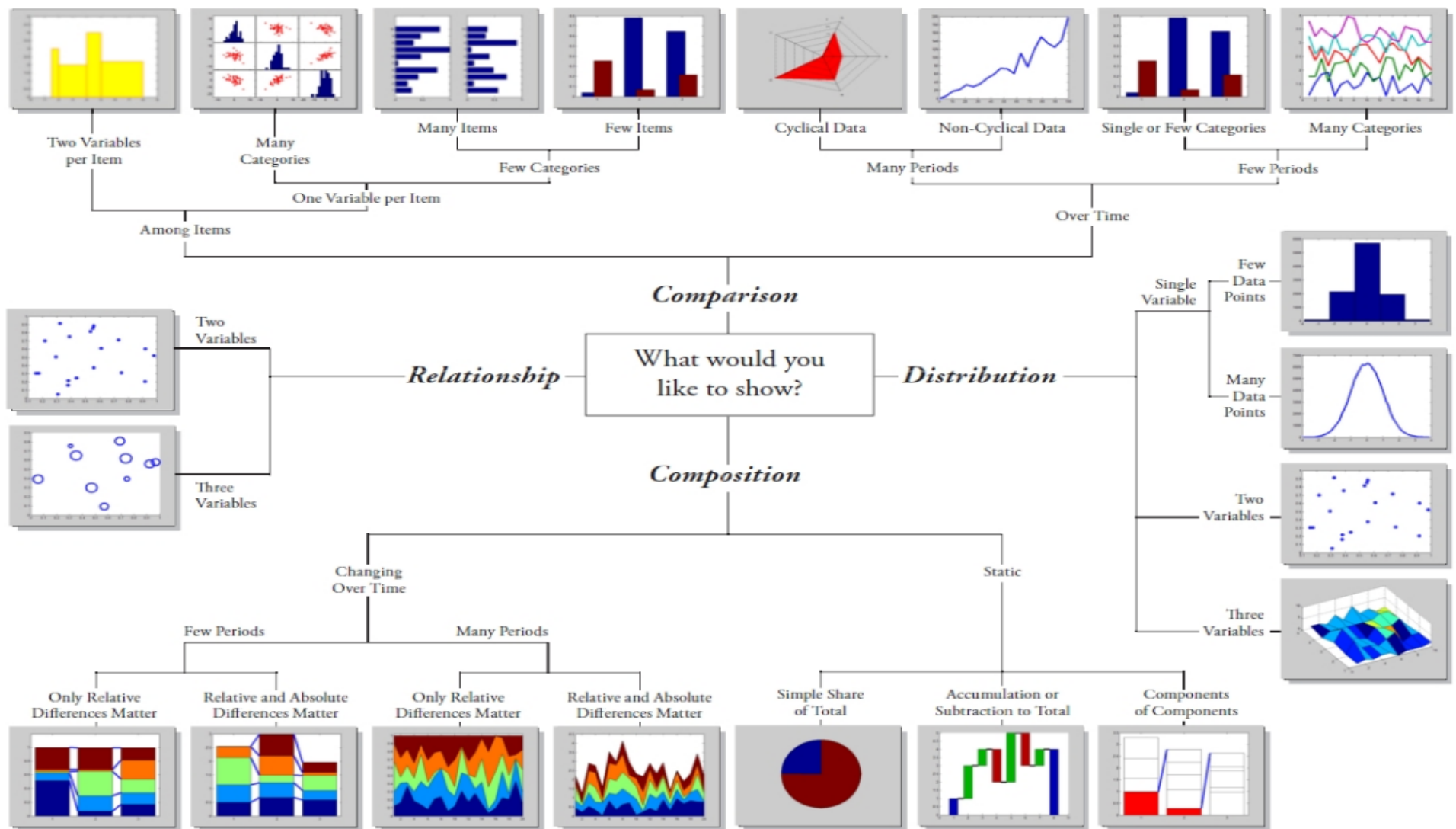
- ❖ EDA: Exploratory Data Analysis
- ❖ 데이터 특징과 데이터에 내재된 관계 파악
- ❖ 이를 위해 시각화와 통계적 분석 방법 활용
- ❖ 파악하고자 하는 내용
  - 자료의 유형과 범위
  - 데이터 수준 파악
  - 단독 변수 혹은 복합 변수의 분포 및 의미 파악
- ❖ 핵심 주제
  - 저항성 강조: 이상치 등 부분적 변동에 대한 민감성 확인
  - 잔차 계산: 관찰 값들이 주 경향에서 벗어난 정도 파악
  - 자료 변수의 재표현: 변수를 적당한 척도로 바꾸는 것
  - 시각화를 통한 현시성: 분석 결과를 이해하기 쉽게 시각화하는 것

# EDA 특징을 위한 시각화 항목

---

- ❖ 막대그래프
- ❖ 히스토그램
- ❖ 박스플롯
- ❖ 산점도
- ❖ 선 그래프 (수평/수직, 함수, 회귀, 꺾은선)
- ❖ 상관관계 시각화(산점도 행렬, 행렬그래프(히트맵))
- ❖ Pandas Profiling(overview, variables, interactions, correlation 등의 정보)

# EDA 특징을 위한 시각화 항목



# EDA 특징을 위한 시각화 항목

---

## ❖ Relationship:

- a connection or correlation between two or more variables through the data presented
- market cap of a given stock over time versus overall market trend.

## ❖ Comparison

- one set of variables apart from another, and display how those two variables interact
- the number of visitors to five competing web sites in a single month.

## ❖ Composition

- collect different types of information that make up a whole and display them together
- the search terms that those visitors used to land on your site, or how many of them came from links, search engines, or direct traffic.

## ❖ Distribution

- lay out a collection of related or unrelated information simple to see how it correlates, if at all, and to understand if there's any interaction between the variables
- the number of bugs reported during each month of a beta



---



**B**

## 데이터 전처리

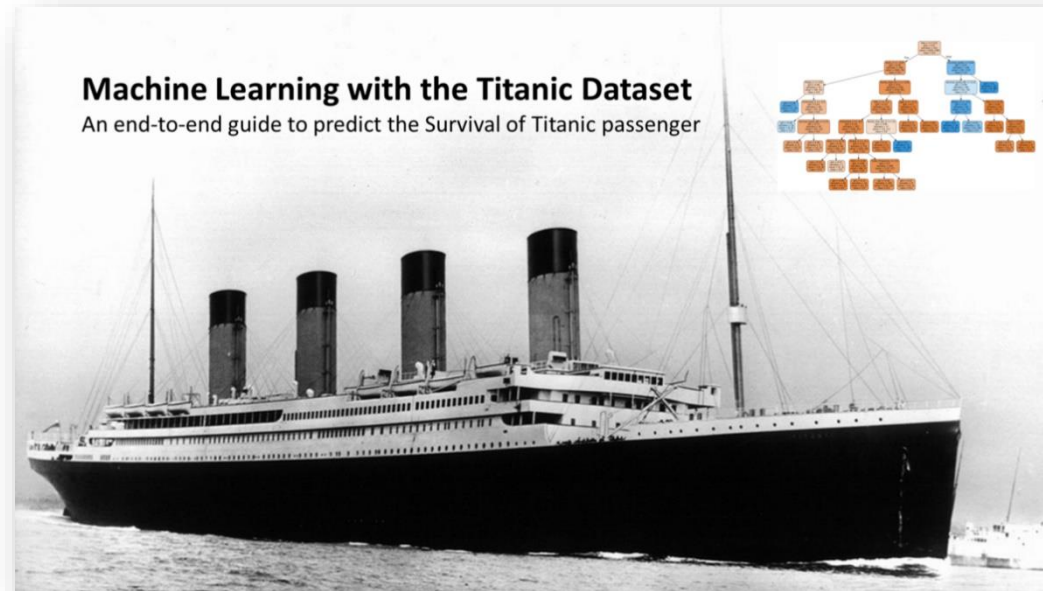
# 데이터 전처리

## ❖ 데이터 전처리 무엇인가요?

- 수집한 데이터를 탐색해보면 데이터를 손볼 곳이 많을 것이다
  - 중간에 데이터가 빠졌거나, 틀린 값이 들어 있거나, 데이터의 단위가 틀릴 수가 있다
- 데이터 분석의 품질에 큰 영향을 미치는 데이터 전처리는 매우 중요한 작업이다
- 실제로 데이터 분석 알고리즘 자체를 수행하는데 걸리는 시간보다 분석에 필요한 데이터 전처리하는 과정에 더 많은 시간이 걸린다
  - 80%-90% 시간이 data preprocessing에 소요됨
- 분석하기 좋게 데이터를 고치는 모든 작업을 데이터 전처리(data preprocessing)라고 한다

# 데이터 전처리

## ❖ Dataset



- titanic.csv 파일을 로컬 저장소에 저장

```
import pandas as pd
```

```
titanic_df = pd.read_csv("D:/titanic.csv")
```

# 데이터 전처리

## ❖ info()

- 데이터셋에 대한 필수 세부정보를 제공
  - titanic\_df.info()

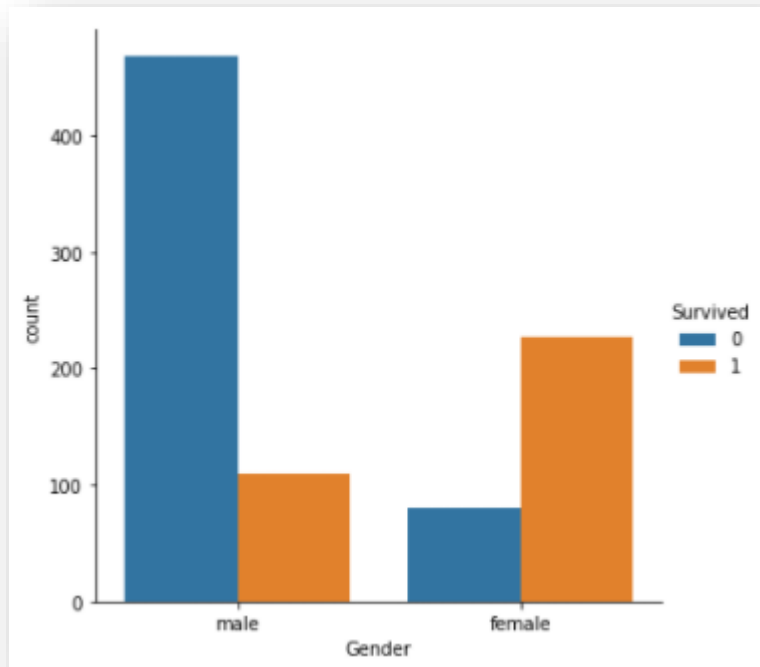
```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 895 entries, 0 to 894  
Data columns (total 12 columns):  
PassengerId    895 non-null int64  
Survived       895 non-null int64  
Pclass         895 non-null int64  
Name           895 non-null object  
Gender         895 non-null object  
Age            718 non-null float64  
SibSp          895 non-null int64  
Parch          895 non-null int64  
Ticket         895 non-null object  
Fare           884 non-null float64  
Cabin          204 non-null object  
Embarked       893 non-null object  
dtypes: float64(2), int64(5), object(5)  
memory usage: 66.5+ KB
```

# 데이터 전처리

## ❖ Target Variable

```
import seaborn as sns
```

```
sns.catplot(x="Gender", hue="Survived",  
            kind="count", data=titanic_df)
```



---



C

## 데이터 정제

# 데이터 정제

## ❖ drop()

- Used to delete columns and rows
  - `titanic_df.drop(columns='Name', inplace=True)`
  - `titanic_df.head(5)`

	PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	male	35.0	0	0	373450	8.0500	NaN	S

- Task
  - 'Cabin', 'Embarked' 및 'Ticket' 열을 동시에 제거

# 데이터 정제

## ❖ drop()

- Used to delete columns and rows
  - `titanic_df.drop(columns=['Cabin', 'Ticket', 'Embarked'], inplace=True)`
  - `titanic_df.head(5)`

	PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Fare
0	1	0	3	male	22.0	1	0	7.2500
1	2	1	1	female	38.0	1	0	71.2833
2	3	1	3	female	26.0	0	0	7.9250
3	4	1	1	female	35.0	1	0	53.1000
4	5	0	3	male	35.0	0	0	8.0500



# 데이터 정제

---

## ❖ 중복 데이터 처리

- Data comes from different sources
  - When collecting and consolidating data from various sources, it's possible that data duplicates exist
- Benefits of removing duplicate data
  - Efficient storage allocation
  - Cost savings
  - Faster data analysis
  - Avoid misleading statistics and maintain high accuracy of analysis

# 데이터 정제

## ❖ 중복 데이터 처리

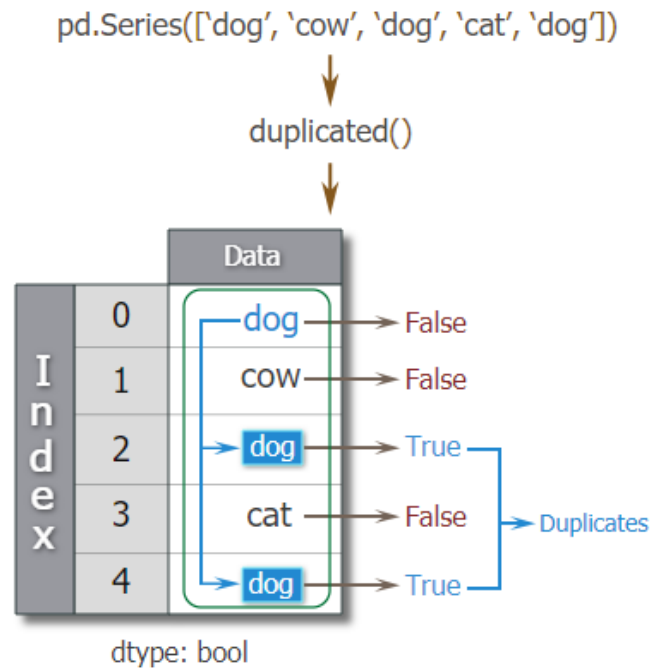
- `titanic_df.tail(10)`

	PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Fare
885	886	0	3	female	39.0	0	5	29.125
886	887	0	2	male	27.0	0	0	13.000
887	888	1	1	female	19.0	0	0	30.000
888	889	0	3	female	NaN	1	2	23.450
889	890	1	1	male	26.0	0	0	30.000
890	891	0	3	male	32.0	0	0	7.750
891	891	0	3	male	32.0	0	0	7.750
892	891	0	3	male	32.0	0	0	7.750
893	891	0	3	male	32.0	0	0	7.750
894	891	0	3	male	32.0	0	0	7.750

# 데이터 정제

## ❖ duplicated()

- Used to indicate duplicate values



# 데이터 정제

## ❖ duplicated()

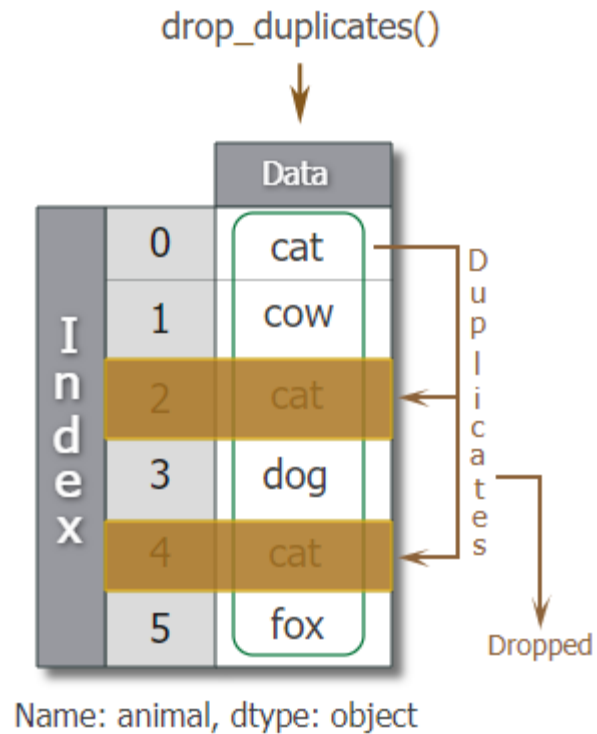
- `titanic_df[titanic_df.duplicated()]`

	PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Fare
891	891	0	3	male	32.0	0	0	7.75
892	891	0	3	male	32.0	0	0	7.75
893	891	0	3	male	32.0	0	0	7.75
894	891	0	3	male	32.0	0	0	7.75

# 데이터 정제

## ❖ drop\_duplicates()

- Used to remove duplicate rows



# 데이터 정제

## ❖ drop\_duplicates()

- Used to remove duplicate rows
  - `titanic_df.drop_duplicates(inplace = True)`
  - `titanic_df[titanic_df.duplicated()]`
  - `titanic_df.tail(10)`

	PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Fare
881	882	0	3	male	33.0	0	0	7.8958
882	883	0	3	female	22.0	0	0	10.5167
883	884	0	2	male	28.0	0	0	10.5000
884	885	0	3	male	25.0	0	0	7.0500
885	886	0	3	female	39.0	0	5	29.1250
886	887	0	2	male	27.0	0	0	13.0000
887	888	1	1	female	19.0	0	0	30.0000
888	889	0	3	female	NaN	1	2	23.4500
889	890	1	1	male	26.0	0	0	30.0000
890	891	0	3	male	32.0	0	0	7.7500

# 데이터 정제

---

## ❖ 누락된 값 다루기

- Can occur due to many reasons such as data entry errors or data collection problems
- Absence of data
  - Reduces the power of data analysis
    - Most data analysis techniques ignore cases with missing data
  - Causes bias or incorrect conclusion in the data analysis
    - Invalidate the entire analysis process
    - May cause substantial risk in clinical studies
  - Reduces representativeness of the data

# 데이터 정제

## ❖ 누락된 값 다루기

- 누락되거나 null 값
  - 데이터셋의 품질을 가리킴
- `isnull()` 함수를 사용해 누락된 값이 있는 위치를 확인
  - `titanic_df.isnull().head(10)`

	PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Fare
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
5	False	False	False	False	True	False	False	False
6	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False

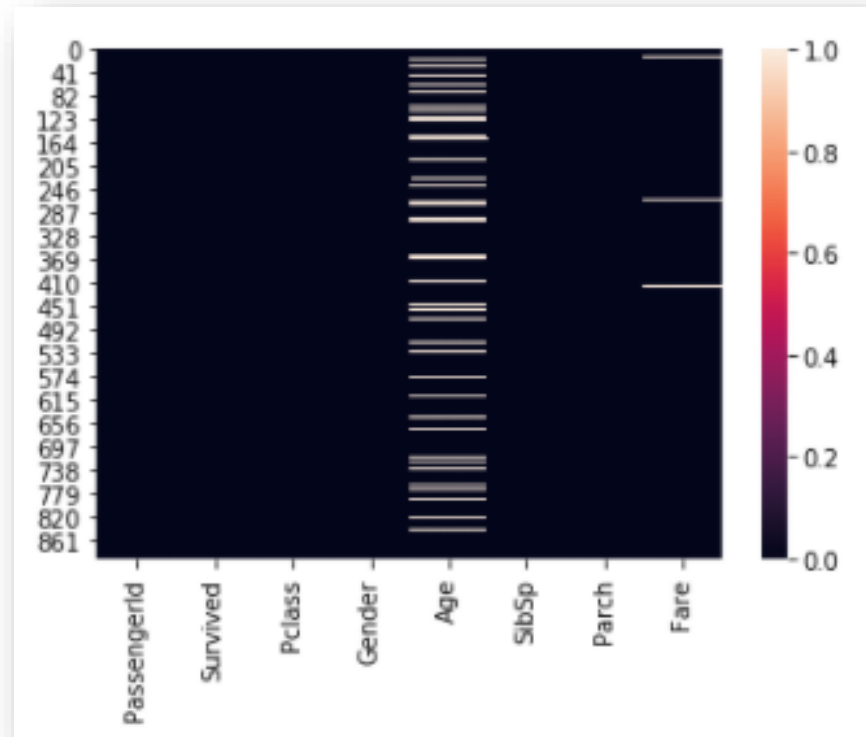


# 데이터 정제

## ❖ 누락된 값 다루기

```
import seaborn as sns
```

```
sns.heatmap(titanic_df.isnull(), cbar=False)
```



# 데이터 정제

## ❖ 누락된 값 다루기

- 누락된 값의 요약을 확인
  - `titanic_df.isnull().sum()`

```
PassengerId    0
Survived        0
Pclass         0
Gender         0
Age          177
SibSp          0
Parch          0
Fare           11
dtype: int64
```

# 데이터 정제

## ❖ 데이터 대체

- Null을 채우기 위한 `fillna()` 함수
  - `titanic_df['Age'].fillna(titanic_df['Age'].mean(), inplace=True)`
  - `titanic_df.isnull().sum()`

```
PassengerId 0
Survived     0
Pclass       0
Gender       0
Age          0
SibSp        0
Parch        0
Fare        11
dtype: int64
```

# 데이터 정제

## ❖ Null 값 제거

- dropna() 함수를 사용하여 null 값이 있는 행 제거
  - titanic\_df.dropna(inplace=True)
  - titanic\_df.isnull().sum()

```
PassengerId 0
Survived     0
Pclass       0
Gender       0
Age          0
SibSp        0
Parch        0
Fare         0
dtype: int64
```

# 데이터 정제

---

## ❖ Null 값 제거

- `dropna()`를 사용해 열을 제거 할 수도 있음
  - `Titanic_df.dropna(axis=1)`
    - 0, or 'index' : 누락된 값이 포함된 행 제거
    - 1, or 'columns' : 누락된 값이 포함된 열 제거
- 힌트
  - 누락된 데이터가 적은 경우에만 null 데이터를 제거하는 것이 좋음

---



D

## 데이터 변화

# 데이터 변화

---

## ❖ columns

- 데이터셋의 열 이름을 출력
  - `titanic_df.columns`

```
Index(['PassengerId', 'Survived', 'Pclass', 'Gender', 'Age', 'SibSp',  
      'Parch', 'Fare'],  
      dtype='object')
```

# 데이터 변화

## ❖ Renaming column names

- There are cases when you want to change the names of columns for better readability

1	Passenger	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, M	male	22	1	0	A/5 21171			S
3	2	1	1	Cumings, f	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen, f	female	26	0	0	STON/O2.	7.925		S
5	4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, M	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy, m	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Palsson, M	male	2	3	1	349909	21.075		S
10	9	1	3	Johnson, f	female	27	0	2	347742	11.1333		S
11	10	1	2	Nasser, M	female	14	1	0	237736	30.0708		C
12	11	1	3	Sandstrom, f	female	4	1	1	PP 9549	16.7	G6	S
13	12	1	1	Bonnell, M	female	58	0	0	113783	26.55	C103	S
14	13	0	3	Saunders, m	male	20	0	0	A/5. 2151	8.05		S
15	14	0	3	Andersson, m	male	39	1	5	347082	31.275		S
16	15	0	3	Vestrom, f	female	14	0	0	350406	7.8542		S
17	16	1	2	Hewlett, M	female	55	0	0	248706	16		S
18	17	0	3	Rice, Master	male	2	4	1	382652	29.125		Q
19	18	1	2	Williams, m	male		0	0	244373	13		S

Number of Siblings and Spouses Aboard



# 데이터 변화

## ❖ rename()

- dict를 통해 특정 또는 모든 열의 이름 바꾸기

```
titanic_df.rename(columns={  
    'SibSp': 'Sibling.Spouse',  
}, inplace=True)
```

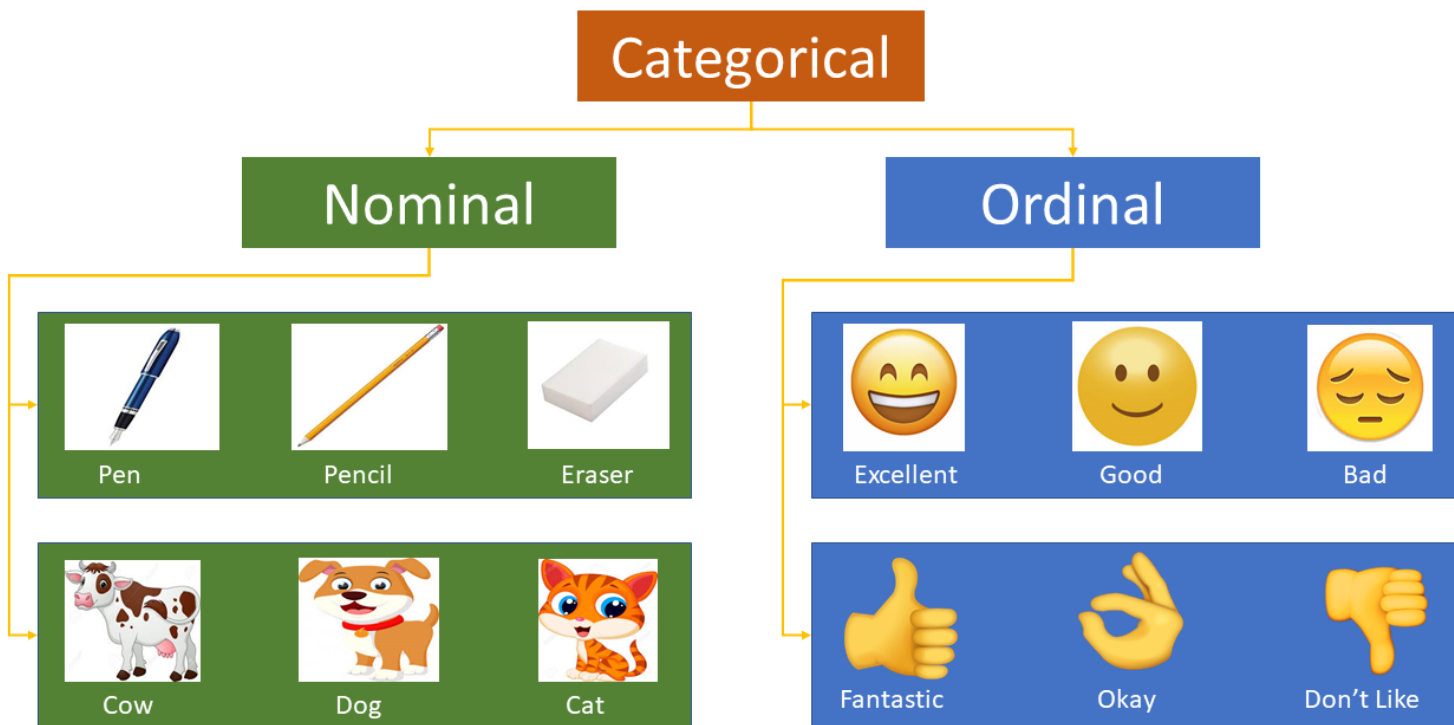
```
titanic_df.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Gender', 'Age',  
      'Sibling.Spouse', 'Parch', 'Fare'],  
      dtype='object')
```

# 데이터 변화

## ❖ Data encoding











- What is categorical data?



# 데이터 변화

## ❖ Data encoding

- Certain types of machine learning algorithms require all input and output variables to be numeric

Gender		Is_Male	Is_Female	Tree		Type
	→	0	1		→	1
	→	0	1		→	2
	→	1	0		→	1
	→	0	1		→	2
	→	1	0		→	3

# 데이터 변화

## ❖ Data encoding

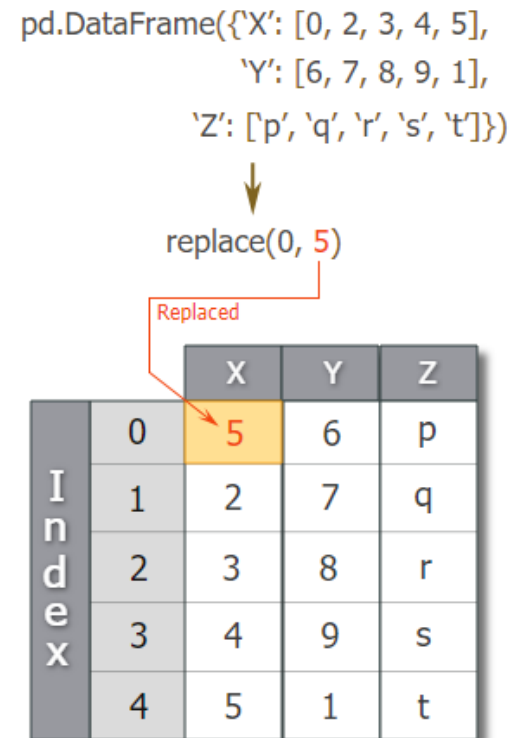
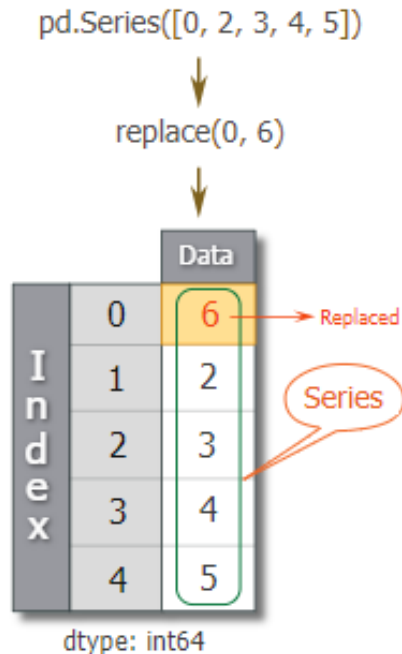
- `titanic_df.head(10)`

	PassengerId	Survived	Pclass	Gender	Age	Subling.Spouse	Parch	Fare
0	1	0	3	male	22.000000	1	0	7.2500
1	2	1	1	female	38.000000	1	0	71.2833
2	3	1	3	female	26.000000	0	0	7.9250
3	4	1	1	female	35.000000	1	0	53.1000
4	5	0	3	male	35.000000	0	0	8.0500
5	6	0	3	male	29.699118	0	0	8.4583
6	7	0	1	male	54.000000	0	0	51.8625
7	8	0	3	male	2.000000	3	1	21.0750
8	9	1	3	female	27.000000	0	2	11.1333
9	10	1	2	female	14.000000	1	0	30.0708

# 데이터 변화

## ❖ replace()

- Used to replace row values



# 데이터 변화

## ❖ replace()

- `titanic_df.head(10)`
- `titanic_df.replace({'Gender': {'male': 0}}, inplace = True)`
- `titanic_df.replace({'Gender': {'female': 1}}, inplace = True)`

	PassengerId	Survived	Pclass	Gender	Age	Subling.Spouse	Parch	Fare
0	1	0	3	0	22.000000	1	0	7.2500
1	2	1	1	1	38.000000	1	0	71.2833
2	3	1	3	1	26.000000	0	0	7.9250
3	4	1	1	1	35.000000	1	0	53.1000
4	5	0	3	0	35.000000	0	0	8.0500
5	6	0	3	0	29.699118	0	0	8.4583
6	7	0	1	0	54.000000	0	0	51.8625
7	8	0	3	0	2.000000	3	1	21.0750
8	9	1	3	1	27.000000	0	2	11.1333
9	10	1	2	1	14.000000	1	0	30.0708

---



D

## 데이터 정규화

# 데이터 정규화

---

## ❖ 이상 값 처리

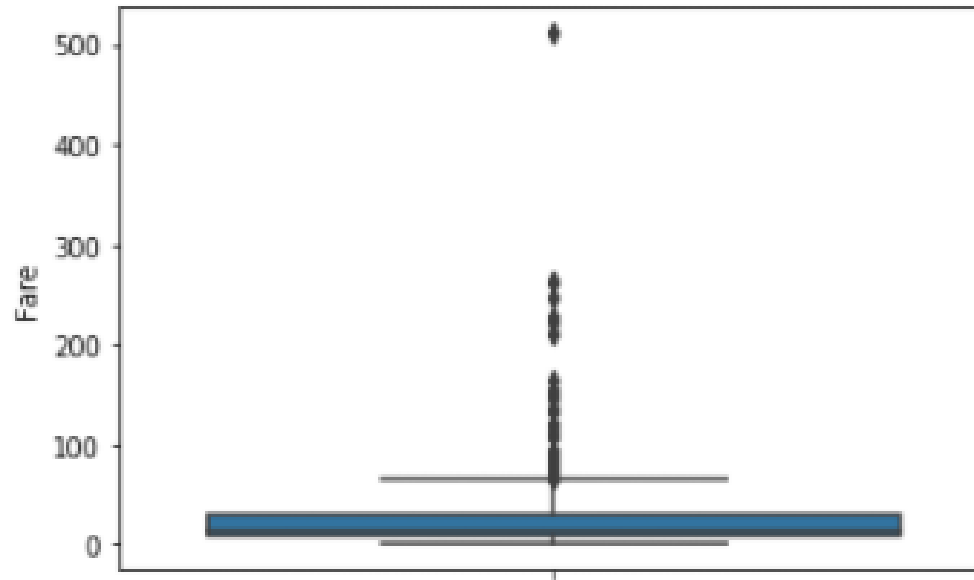
- Outlier is a data point that differs significantly from other data points
- Causes of outliers
  - Data entry errors (i.e., human errors)
  - Measurement errors (i.e., instrument or device errors)
  - Intentional (i.e., malicious insertions)
  - Data preprocessing errors (i.e., unintended data manipulations or mutations)



# 데이터 정규화

## ❖ 이상 값 처리

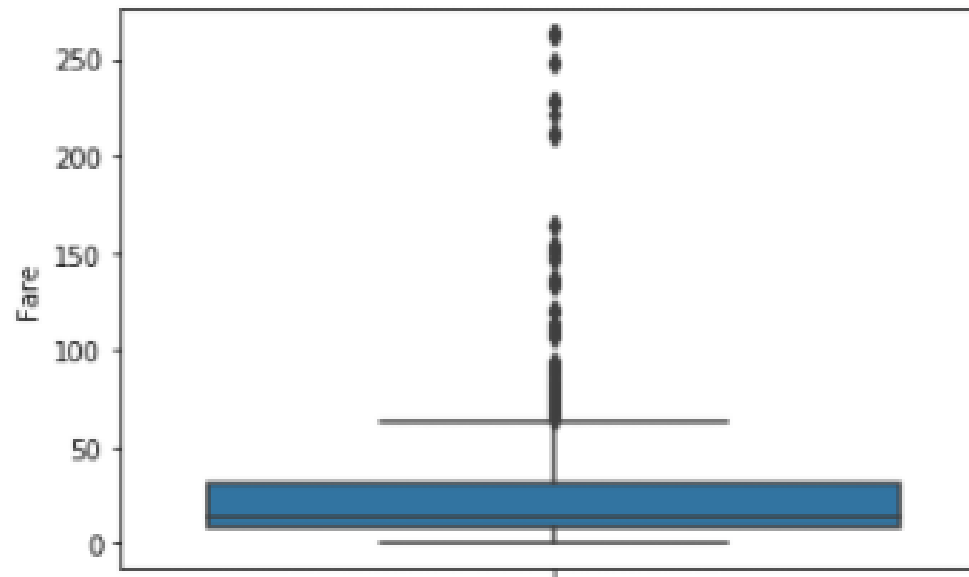
- `sns.boxplot(y = titanic_df['Fare'] )`;



# 데이터 정규화

## ❖ 이상 값 처리

- `titanic_df.drop(titanic_df[titanic_df.Fare >= 500].index, inplace = True)`
- `sns.boxplot(y = titanic_df['Fare'] );`



# 데이터 정규화

## ❖ 정규화는 왜 필요한가요?

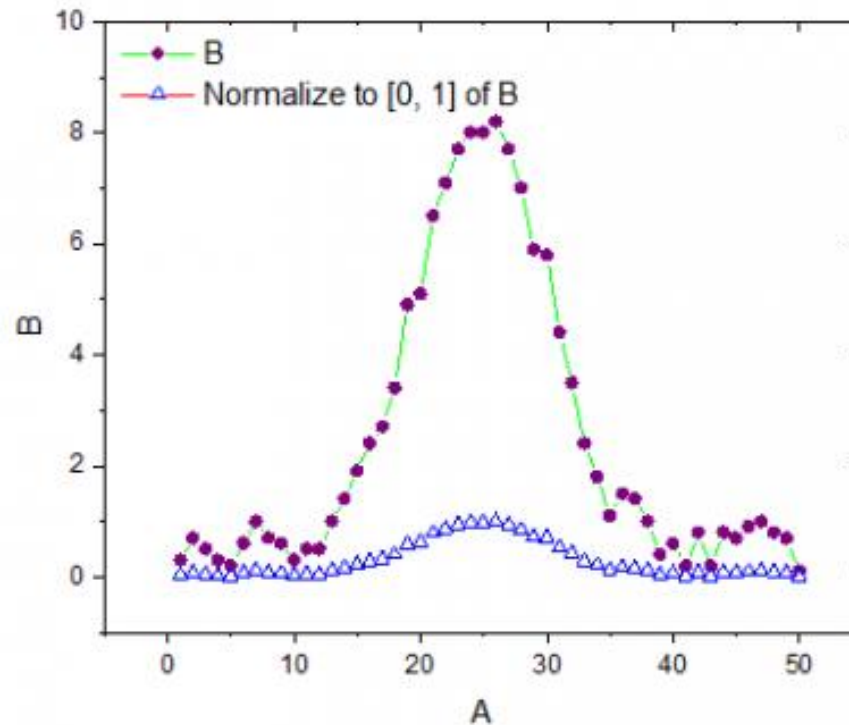
person_name	Salary	Year_of_experience	Expected Position Level
Aman	100000	10	2
Abhinav	78000	7	4
Ashutosh	32000	5	8
Dishi	55000	6	7
Abhishek	92000	8	3
Avantika	120000	15	1
Ayushi	65750	7	5

The attributes salary and year\_of\_experience are on different scale and hence attribute salary can take high priority over attribute year\_of\_experience in the model.

# 데이터 정규화

## ❖ 정규화는 왜 필요한가요?

- Normalize data and scale down distortions



# 데이터 정규화

## ❖ 정규화는 왜 필요한가요?

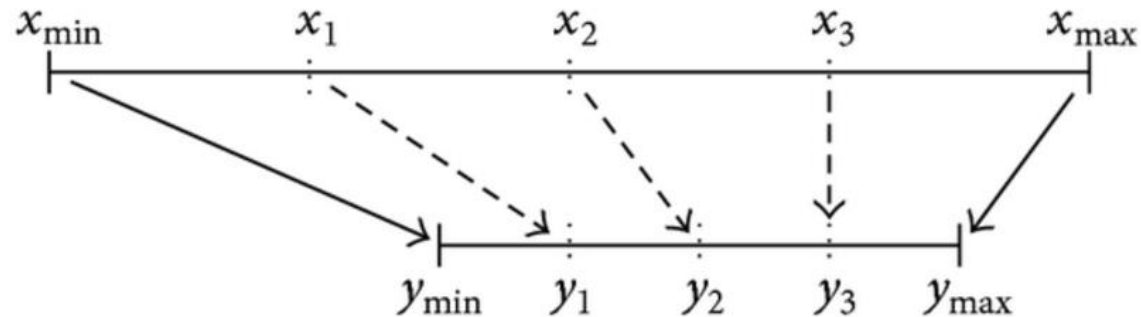
- `titanic_df.describe()`

	PassengerId	Survived	Pclass	Age	Subling.Spouse	Parch	Fare
count	877.000000	877.000000	877.000000	877.000000	877.000000	877.000000	877.000000
mean	447.272520	0.379704	2.314709	29.774818	0.523375	0.379704	30.470386
std	258.040911	0.485590	0.833886	13.033179	1.103528	0.808459	41.212739
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.000000	0.000000	2.000000	22.000000	0.000000	0.000000	7.895800
50%	449.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	670.000000	1.000000	3.000000	35.000000	1.000000	0.000000	30.695800
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	263.000000

# 데이터 정규화

## ❖ Min-max 정규화

- 0과 1 사이의 데이터 정규화



- 다음 공식에 따라 계산

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

# 데이터 정규화

## ❖ Min-max 정규화

- sklearn의 MinMaxScaler() 함수 사용

```
from sklearn.preprocessing import MinMaxScaler  
  
scaler = MinMaxScaler()  
  
titanic_df[['Fare']] = scaler.fit_transform(titanic_df[['Fare']])  
  
titanic_df.describe()
```

# 데이터 정규화

## ❖ Min-max 정규화

- sklearn의 MinMaxScaler() 함수 사용

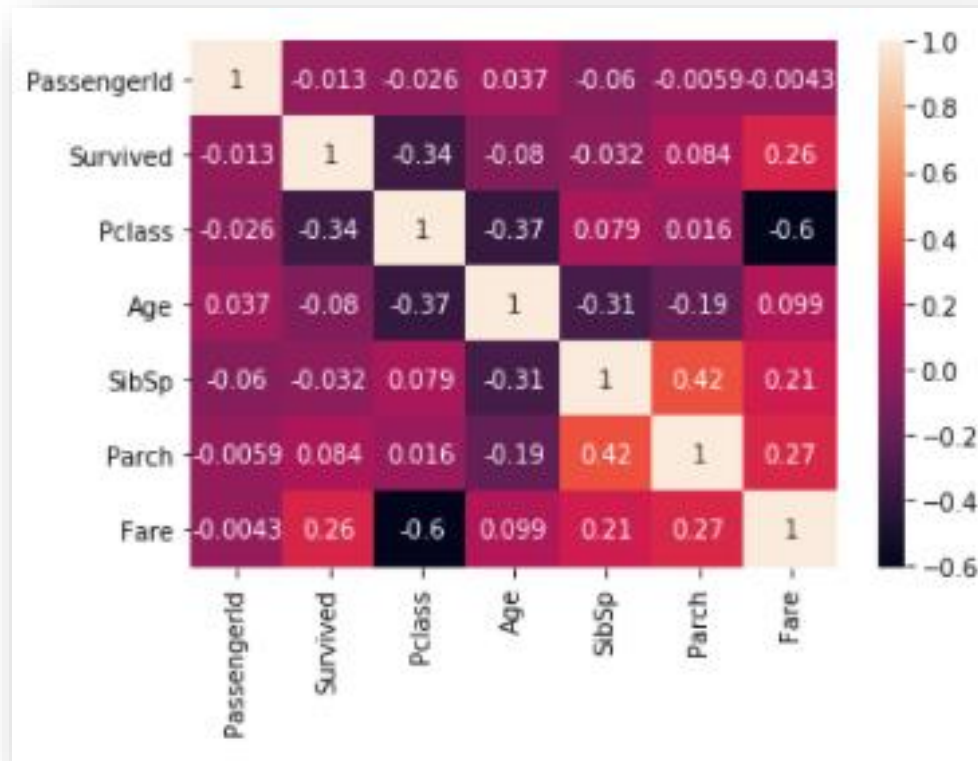
	PassengerId	Survived	Pclass	Age	Subling.Spouse	Parch	Fare
count	877.000000	877.000000	877.000000	877.000000	877.000000	877.000000	877.000000
mean	447.272520	0.379704	2.314709	29.774818	0.523375	0.379704	0.115857
std	258.040911	0.485590	0.833886	13.033179	1.103528	0.808459	0.156702
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.000000	0.000000	2.000000	22.000000	0.000000	0.000000	0.030022
50%	449.000000	0.000000	3.000000	29.699118	0.000000	0.000000	0.054959
75%	670.000000	1.000000	3.000000	35.000000	1.000000	0.000000	0.116714
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	1.000000



# 데이터 정규화

## ❖ Correlation

- `sns.heatmap(titanic_df.corr(), annot=True)`



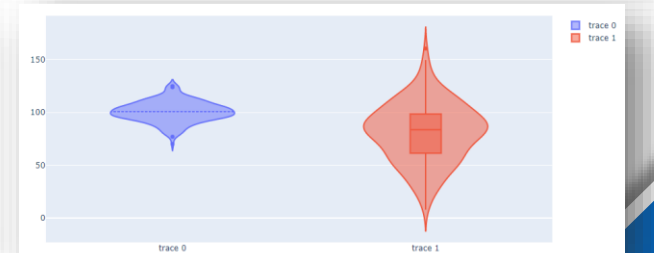
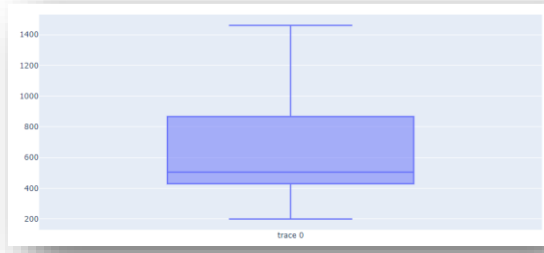
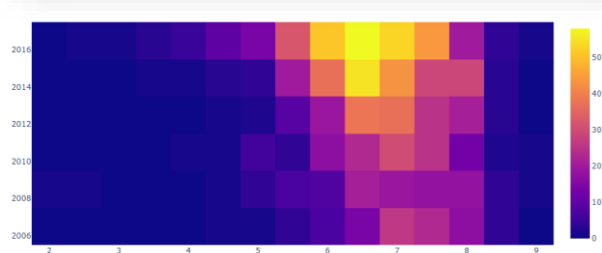
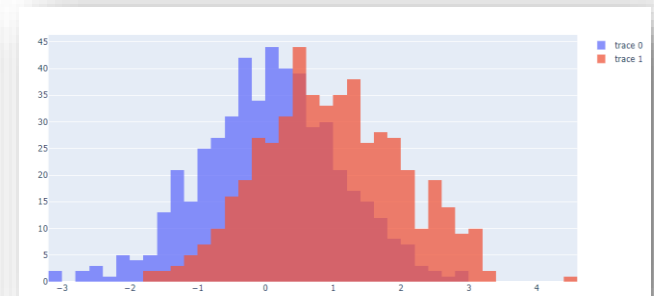
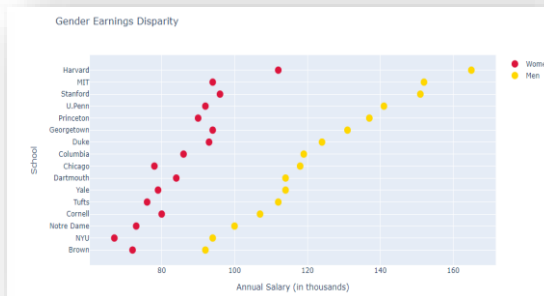
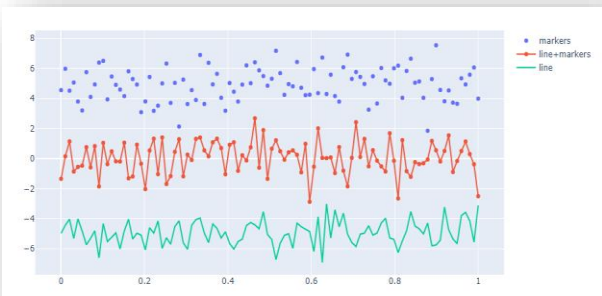
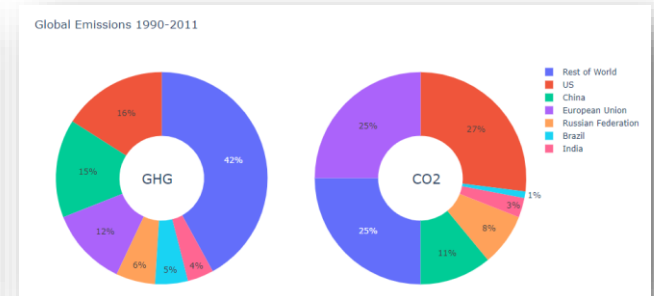
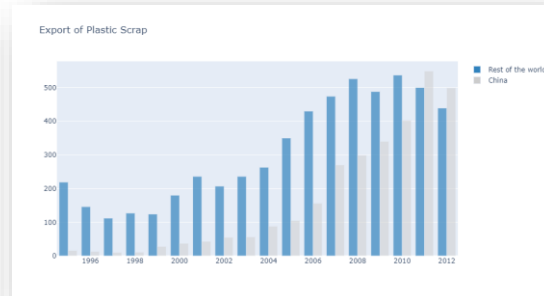
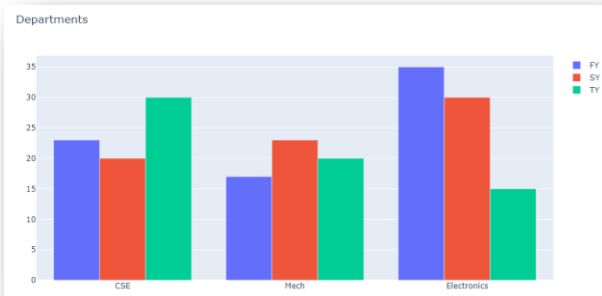
# Final Task (insurance.csv)

---

1. 불필요한 열 제거하기
  - Number 열
2. 중복 데이터 찾기 및 제거하기
  - duplicated() & drop\_duplicate()
3. Null 값 제거
  - dropna()
4. rename() 이용하여 열 이름 바꾸기
  - Gender -> gender, Insurance Fee -> charges
5. replace() 이용하여 행 값 바꾸기
  - female -> 1, male -> 0
6. 이상 값 처리
7. Min-max 방법을 이용하여 데이터 정규화
8. 'charges' 컬럼과 상관 관계가없는 컬럼 삭제

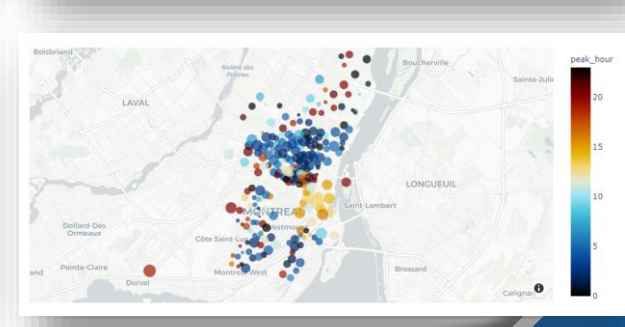
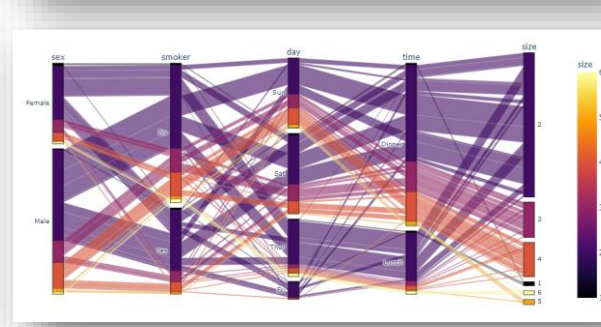
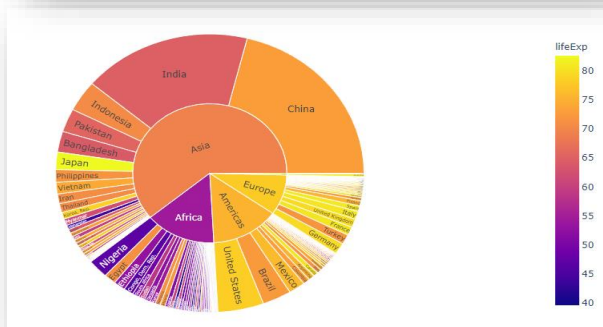
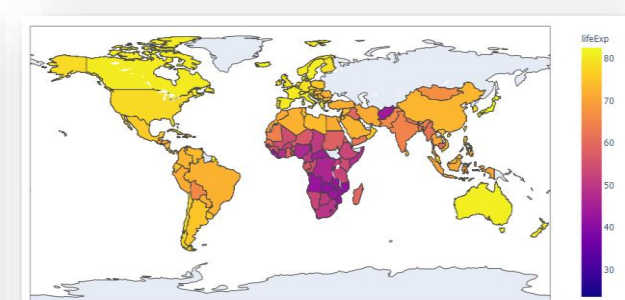
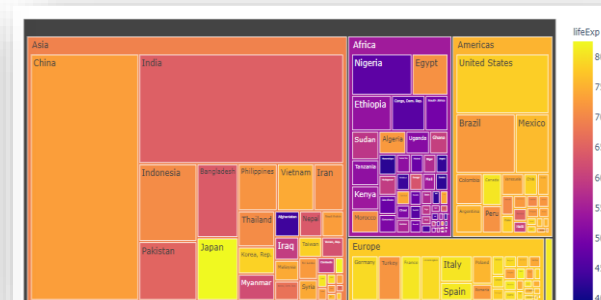
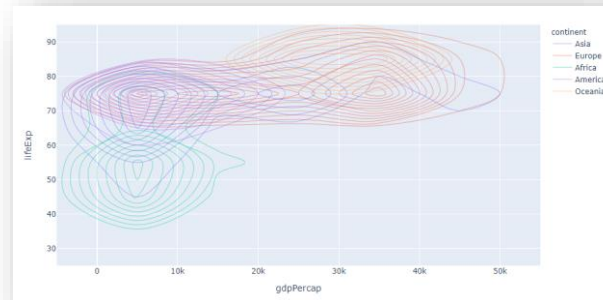
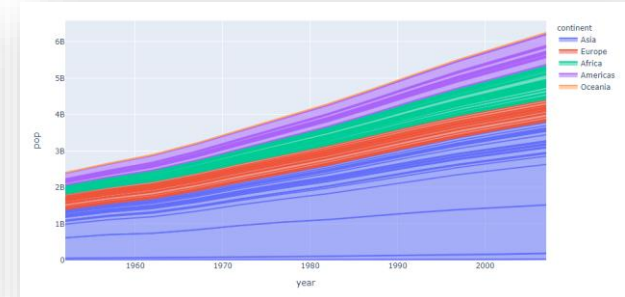
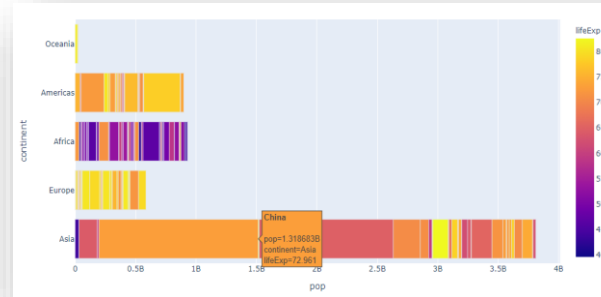
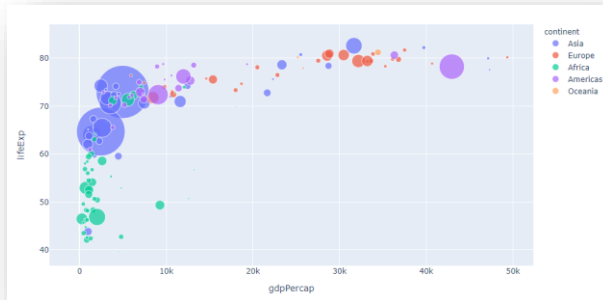
# Plotly

## ❖ Studied plots



# Plotly Express

## ❖ Studied plots





감사합니다!