『 3과목 :』
데이터 마트와 데이터 전처리

- Data Mart & Data Preprocessing
- Data Structures
- Data Gathering(Collect, Acquisition), Data Ingestion
- Data Invest & Exploratory Data Analysis, Data Visualization
- **Data Cleansing (정제)**
- Data Integration (통합)
- Data Reduction (축소)
- Data Transformation (변환)
- Feature Engineering & Data Encoding
- Cross Validation & Data Splitting
- Data Quality Assessment and Model Performance Evaluation
- 『3과목』 Self 점검

개요

학습목표

- 이 워크샵에서는 Data Cleansing 에 대해 알 수 있다.
- Matching and Formatting, Filtering & Sorting 에 대해 알 수 있다.
- 실제 데이터는 여러가지 노이즈와 문제가 있을 수 있으므로, 적절한 전처리가 필요한다.

눈높이 체크

- Data Cleansing 을 알고 계신가요
- Matching and Formatting, Filtering & Sorting 을 알고 계신가요?

1. Data Cleansing

Data Cleansing?

- 데이터 정제는 주어진 데이터에서 노이즈, 오류, 누락된 값 또는 불필요한 정보를 식별하고 수정하여 데이터의 정확성과 완전성을 향상시키는 과정을 의미한다.
- 데이터 정제는 데이터 분석 및 모델링 과정에서 정확한 결과를 얻기 위해 필수적인 단계로, 데이터의 품질을 향상시켜 유용한 인사이트를 도출하는 데 도움을 준다.
- 이 프로세스는 데이터의 이상치 제거, 중복 제거, 누락된 값 대체, 형식 통일 등의 작업을 포함할 수 있다.

  - 결측치 처리: 누락된 데이터를 확인하고, 대체하거나 삭제하는 방법을 사용하여 데이터의 완전성을 보완한다.
  - 이상치 탐지 및 처리: 이상치를 식별하고, 제거하거나 대체하여 데이터의 정확성을 향상시킨다.

1. Data Cleansing

데이터 조사와 정제Data Invest & Cleansing

- 데이터를 활용할 수 있도록 만드는 과정
- 데이터의 누락값, 불일치, 오류의 수정
- 컴퓨터가 읽을 수 없는 요소의 제거
- 숫자나 날짜 등의 형식에 대해 일관성 유지
- 적합한 파일 포맷으로 변환

출처 : https://www.dataentryoutsourced.com/blog/cxos-guide- to-marketing-and-sales-data-cleansing-and-enrichment/

1. Data Cleansing

데이터 조사와 정제Data Invest & Cleansing

- 속성의 데이터 타입과 도메인(속성 값의 범위)
- 속성 값의 분포 특성(대칭, 비대칭 등)
  - 대칭/비대칭 분포
  - 실제 값의 주요 분포 범위
  - 값의 표준편차
- 속성 간의 의존성
  - 속성 A의 값이 같은 데이터의 속성 B 값이 반드시 같다면, 속성 A와 속성 B 간의 함수적 종속성 존재 (A→B)

2. How does the data cleansing work?

데이터 정제 절차

- 일부 데이터는 올바르게 형식화되어 있어 바로 사용할 수 있지만, 대부분의 데이터는 형식 불일치 혹은 가독성 문제(예: 약어 또는 일치하지 않는 헤더 설명)가 있다. 둘 이상의 데이터 세트에서 데이터를 사용하는 경우 특히 심하다. 따라서 데이터를 정리하면 더 쉽게 저장, 검색 및 재사용을 할 수 있다.

- 일반적으로, 데이터 정제 절차는 다음과 같을 수 있다.

1. Matching 을 할 수 있다.
2. Formatting을 할 수 있다.
3. Filtering & Sorting을 할 수 있다.

2. How does the data cleansing work?

데이터 정제 절차

- Matching (일치):
  - 데이터의 일관성을 유지하기 위해 다른 소스에서 가져온 데이터 간에 일치하는지 확인한다. 예를 들어, 다른 시스템에서 가져온 데이터의 형식이나 구조를 확인하고 일치시키는 작업을 수행할 수 있다.

- Formatting (형식화):
  - 데이터를 일관된 형식으로 변환하거나 조정하여 일관성을 유지하고 분석에 용이하도록 만듭니다. 이는 데이터 타입 변환, 날짜 형식 표준화, 텍스트 형식 통일화 등을 포함할 수 있다.

2. How does the data cleansing work?

데이터 정제 절차

- Matching (일치) 예

```
import pandas as pd

# 두 데이터셋 예시 (data1과 data2)
data1 = pd.read_csv(`datasets/data1.csv')  # 첫 번째 데이터셋
data2 = pd.read_csv('datasets/data2.csv')  # 두 번째 데이터셋

# 일치시킬 기준이 되는 열을 기준으로 Merge (예시: 'key' 열을 기준으로 Merge)
merged_data = pd.merge(data1, data2, on='key_column', how='inner')
# 'key_column'은 두 데이터셋에서 일치시킬 기준이 되는 열 이름입니다.

# Merge된 데이터 확인
print(merged_data)
```

2. How does the data cleansing work?

데이터 정제 절차

- Filtering (필터링):
  - 데이터셋에서 특정 기준을 충족하는 행 또는 열을 선택하는 과정을 의미한다. 예를 들어, 조건에 따라 특정 날짜 범위의 데이터만 선택하거나, 특정 조건을 만족하는 행을 추출하는 것이 필터링에 해당한다.

- Sorting (정렬):
  - 데이터를 특정 기준에 따라 순서대로 배열하는 작업을 말한다. 이는 데이터를 오름차순 또는 내림차순으로 정렬하는 것을 의미하며, 보통 숫자나 날짜 등의 기준으로 데이터를 정렬한다.

3. Formatting

파이썬으로 글자를 출력하기

- 파이썬과 같은 프로그래밍 언어에서는 글자를 문자열(string)이라고 부른다. 파이썬에서 문자열을 만들 때는 따옴표를 사용한다. 따옴표에는 큰 따옴표와 작은 따옴표가 있으며 시작 따옴표와 종료 따옴표만 같으면 어느 것을 사용해도 상관없다.
  - 문자열을 출력하려면 print 명령을 사용한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	print("Hello!") print('한글도 쓸 수 있어요')
결과값1	Hello! 한글도 쓸 수 있어요.
비고	

3. Formatting

문자열 연산

- 문자열도 숫자처럼 덧셈과 곱셈 연산을 할 수 있다. 덧셈 연산은 두 문자열을 붙이고 곱셈 연산은 문자열을 반복한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	print("내 이름은 " + "홍길동" + "입니다.")
결과값1	내 이름은 홍길동입니다.

숫자를 문자열로 바꾸기

- 숫자를 문자열과 더하려면 str 명령을 써서 숫자를 문자열 자료형으로 바꾸어야 한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	print("*" * 10) n = 10 print("별표를 " + str(n) + "번 출력한다.") print("*" * n)
결과값1	********** 별표를 10번 출력한다. **********

3. Formatting

한 줄 띄우기

- print 명령은 한 번 호출할 때마다 한 줄씩 출력한다. 만약 print 명령을 한 번만 쓰면서 여러 줄에 걸쳐 출력을 하고 싶으면 문자열에 "다음 줄 넘기기(line feed) 기호"인 \n를 넣어야 한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	print("한 줄 쓰고\n그 다음 줄을 쓴다.")
결과값1	한 줄 쓰고 그 다음 줄을 쓴다.
비고	

줄을 바꾸지 않고 이어서 출력하기

- 반대로 print 명령을 여러번 쓰면서 줄은 바꾸지 않고 싶다면 다음과 같이 print 명령에 end=""이라는 인수를 추가한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	print("한 줄 쓰고 ", end="") print("이어서 쓴다.")
결과값1	한 줄 쓰고 이어서 쓴다.
비고	

3. Formatting

문자열 값을 가지는 변수

- 변수에는 숫자뿐만 아니라 문자열도 넣을 수 있다.

파일	소스코드
실습환경	py3_10_basic
소스코드	name = "홍길동" print("내 이름은 " + name + "입니다.") mark = "$" n = 20 print(mark + " 기호를 " + str(n) + "번 출력한다.") print(mark * n)
결과값1	내 이름은 홍길동입니다. $ 기호를 20번 출력한다. $$$$$$$$$$$$$$$$$$$$
비고	

3. Formatting

따옴표를 출력하기

- 파이썬에서 두 가지 종류의 다른 따옴표를 쓸 수 있는 이유는 문자열이 따옴표를 포함하는 경우가 있기 때문이다. 만약 따옴표로 둘러싸인 문자열에 따옴표가 포함되어 있다면 파이썬은 그 부분에서 문자열이 끝난다고 인식하여 오류가 발생한다.
- 이처럼 문자열 안에 큰따옴표가 있어야 할 때는 전체 문자열을 작은따옴표로 둘러싸면 된다.

파일	소스코드
실습환경	py3_10_basic

소스코드	print('둘리가 "호이!"하고 말했어요.')
	print("둘리가 '이제 어디로 가지?'하고 생각했어요.")
	# print("둘리가 "호이!"하고 말했어요")
	print("둘리가 \"호이!\"하고 말했어요")

결과값1	둘리가 "호이!"하고 말했어요. 둘리가 '이제 어디로 가지?'하고 생각했어요. 둘리가 "호이!"하고 말했어요
비고	

3. Formatting

따옴표를 출력하기

- \" 사용하기

```
print("둘리가 "호이!"하고 말했어요")
```

```
  Cell In[29], line 1
    print("둘리가 "호이!"하고 말했어요")
                  ^
SyntaxError: invalid syntax
```

```
print("둘리가 \"호이!\"하고 말했어요")
```

```
둘리가 "호이!"하고 말했어요
```

3. Formatting

탈출문자

- 줄 바꿈 문자인 '\n' 삽입

코드		설명
①	\n	개행 (줄바꿈)
②	\v	수직 탭
③	\t	수평 탭
④	\r	캐리지 리턴
⑤	\f	폼 피드
⑥	\a	벨 소리
⑦	\b	백 스페이스
⑧	\000	널문자
⑨	\\	문자 "\"
⑩	\'	단일 인용부호(')
⑪	\"	이중 인용부호(")

3. Formatting

탈출문자

- 줄 바꿈 문자인 '\n' 삽입

파일	소스코드
실습환경	py3_10_basic
소스코드	print("Life is short, You need Python.") print("Life is short,\n You need Python.") print("Pine \"Apple\"입니다.") # Pine"Apple"입니다. print("C:\\Users\\DaeKyeong\\Desktop\\PythonWorkspace>") # C:\Users\DaeKyeong\Desktop\PythonWorkspace> print("Red Apple\rPine") # PineApple print("Redd\bApple") # RedApple print("Red\tApple") # Red Apple
결과값1	Life is short, You need Python. Life is short, You need Python. Pine "Apple"입니다. C:\Users\DaeKyeong\Desktop\PythonWorkspace> PineApple RedApple Red Apple
비고	

3. Formatting

여러 줄의 문자열 출력하기

- 파이썬에서 여러 줄의 문자열을 출력하거나 변수에 할당하려면, " 문자" 나 '문자' 대신 """ 여러 줄의 문자열 """ 혹은 '''여러 줄의 문자열''' 을 사용하면 된다.

파일	소스코드
실습환경	py3_10_basic
소스코드	multi_line_string = """ 파이썬(영어: Python)은 1991년 프로그래머인 귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어이다. 파이썬이라는 이름은 귀도가 좋아하는 코미디 〈Monty Python's Flying Circus〉에서 따온 것이다.""" print(multi_line_string)
결과값1	파이썬(영어: Python)은 1991년 프로그래머인 귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어이다. 파이썬이라는 이름은 귀도가 좋아하는 코미디 〈Monty Python's Flying Circus〉에서 따온 것이다.
비고	

4. Matching, and Formatting

문자열 치환

- 문자열에서 특정 문자를 다른 문자로 바꾸려면 replace 메서드를 사용한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	print("2023.03.01".replace(".","-"))
결과값1	2023-03-01
비고	

- 문자열의 공백을 없애려면 " " 공백 문자열을 "" 빈 문자열로 바꾸면 된다.

파일	소스코드
실습환경	py3_10_basic
소스코드	print("word with space".replace(" ",""))
결과값1	wordwithspac
비고	

4. Matching, and Formatting

Replacing

- 데이터 프레임에 있는 값을 바꾸어야 하는 경우
  - 데이터 적재

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 라이브러리를 임포트한다.
	import pandas as pd
	
	# 데이터를 적재한다.
	dataframe = pd.read_csv("datasets/titanic.csv")
	
	# 두 개의 행을 확인한다.
	dataframe.head(2)

결과값1	
결과값2	

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

4. Matching, and Formatting

Replacing

- 데이터 프레임에 있는 값을 바꾸어야 하는 경우
  - replace 메서드 사용

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	# 값을 치환하고 두 개의 행을 출력한다. dataframe['Gender'].replace("female", "Woman").head(2) # "female"과 "male을 "Woman"과 "Man"으로 치환한다. dataframe['Gender'].replace(["female", "male"], ["Woman", "Man"]).head(5)
결과값1	
결과값2	0 male 1 Woman Name: Gender, dtype: object 0 Man 1 Woman 2 Woman 3 Woman 4 Man Name: Gender, dtype: object
비고	

4. Matching, and Formatting

Replacing

- 데이터 프레임에 있는 값을 바꾸어야 하는 경우
  - replace 메서드 사용

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 값을 치환하고 두 개의 행을 출력한다.
	dataframe.replace(1, "One").head(2)
	
	# 값을 치환하고 두 개의 행을 출력한다.
	dataframe.replace(r"1st", "First", regex=True).head(2)

결과값1	
결과값2	
비고	

```
# 값을 치환하고 두 개의 행을 출력한다.
dataframe.replace(1, "One").head(2)
```

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	One	0	3	Braund, Mr. Owen Harris	male	22.0	One	0	A/5 21171	7.2500	NaN	S
1	2	One	One	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	One	0	PC 17599	71.2833	C85	C

```
# 값을 치환하고 두 개의 행을 출력한다.
dataframe.replace(r"1st", "First", regex=True).head(2)
```

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

4. Matching, and Formatting

Replacing

- 데이터 프레임에 있는 값을 바꾸어야 하는 경우
  - replace 메서드 사용

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# female과 male을 person으로 바꿉니다.
	dataframe.replace(["female", "male"], "person").head(3)
	
	# female을 1로 바꾸고 male을 0으로 바꿉니다.
	dataframe.replace({"female": 1, "male": 0}).head(3)

결과값1	
결과값2	
비고	

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	person	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	person	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	person	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

4. Matching, and Formatting

Replacing

- Replacing headers
  - rename 메서드 사용

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 열 이름을 바꾸고 두 개의 행을 출력한다.
	dataframe.rename(columns={'PClass': 'Passenger Class'}).head(2)

결과값1	PassengerId Survived Pclass Name Gender Age SibSp Parch Ticket Fare Cabin Embarked Name Braund, Mr. Owen Harris 1 0 3 Braund, Mr. Owen Harris male 22.0 1 0 A/5 21171 7.2500 NaN S Cumings, Mrs. John Bradley (Florence Briggs Thayer) 2 1 1 Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0 1 0 PC 17599 71.2833 C85 C
결과값2	
비고	

4. Matching, and Formatting

Replacing

- Replacing headers
  - rename 메서드 사용

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 라이브러리를 임포트한다.
	import collections
	
	# 딕셔너리를 만듭니다.
	column_names = collections.defaultdict(str)
	
	# 키를 만듭니다.
	for name in dataframe.columns:
	  column_names[name]
	
	# 딕셔너리를 출력한다.
	column_names

결과값1	defaultdict(str, {'PassengerId': '', 'Survived': '', 'Pclass': '', 'Name': '', 'Gender': '', 'Age': '', 'SibSp': '', 'Parch': '', 'Ticket': '', 'Fare': '', 'Cabin': '', 'Embarked': ''})
비고	

4. Matching, and Formatting

Replacing

- Replacing headers
  - rename 메서드 사용

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 인덱스 0을 -1로 바꿉니다.
	dataframe.rename(index={0:-1}).head(2)
	
	# 열 이름을 소문자로 바꿉니다.
	dataframe.rename(str.lower, axis='columns').head(2)

결과값1	
결과값2	
비고	

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
-1	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

```
# 열 이름을 소문자로 바꿉니다.
dataframe.rename(str.lower, axis='columns').head(2)
```

	passengerid	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	cabin	embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

4. Matching, and Formatting

문자열 처리 함수 종류

함수이름	의미
str.upper()	문자열 str을 모두 대문자로 바꾸어 준다.
str.count(x)	문자열 str 중 x와 일치하는 것의 개수를 반환한다.
str.find(x)	문자열 str 중 문자 x가 처음으로 나온 위치를 반환한다. 없으면 -1을 반환한다.
str.index(x)	문자열 str 중 문자 x가 처음으로 나온 위치를 반환한다. 없으면 에러를 발생시킨다.
str.join(s)	s라는 문자열의 각각의 요소 문자 사이에 문자열 str를 삽입한다.
str.lower()	문자열 str을 모두 소문자로 바꾸어 준다.
str.lstrip()	문자열 str의 왼쪽 공백을 모두 지운다.
str.rstrip()	문자열 str의 오른쪽 공백을 모두 지운다.
str.strip()	문자열 str의 양쪽 공백을 모두 지운다.
str.replace(s,r)	문자열 str의 s라는 문자열을 r이라는 문자열로 치환한다.
str.split(s)	문자열 str을 s라는 문자열로 나누어 리스트 값을 돌려준다.
str.split()	s를 주지 않으면 공백으로 나누어 리스트 값을 돌려준다.
str.swapcase()	문자열 str의 대문자는 소문자로, 소문자는 대문자로 각각바꾸어 준다.

문자열 처리 함수 사용방법

파일	소스코드
실습환경	py3_10_basic

소스코드	python = "Life is short, You need Python."
	
	print(python.lower())
	print(python.upper())
	print(python[0].isupper()) # True : 0 번째 인덱스의 값이 대문자인지 확인
	print(len(python)) # 17 : 띄어쓰기를 포함한 문자열의 전체 길이 (length)
	print(python.replace("Python", "Java"))
	index = python.index("n") # 처음으로 발견된 n 의 인덱스
	print(index) # 5 : Python 의 n
	index = python.index("n", index + 1) # 6 번째 인덱스 이후에 처음으로 발견된 n 의 인덱스
	print(index)
	find = python.find("n") # 처음으로 발견된 n 의 인덱스
	print(find) # 5 : Python 의 n
	find = python.find("n", find + 1) # 6 번째 인덱스 이후에 처음으로 발견된 n 의 인덱스
	print(find)
	#print(python.index("Java")) # Java 가 없기 때문에 에러가 발생하며 프로그램 종료
	print(python.find("Java")) # Java 가 없으면 -1 을 반환(출력)하며 프로그램 계속 수행
	print(python.count("n")) # 2 : 문자열 내에서 n 이 나온 횟수

결과값1	life is short, you need python. LIFE IS SHORT, YOU NEED PYTHON. True 31 Life is short, You need Java. 19 29 19 29 -1 2
비고	

파이썬으로 문자열 위치 알기 (find, index)-값이 있을 때

파일	소스코드
실습환경	py3_10_basic
소스코드	company = '삼성전자,LG전자,현대자동차,CJ,SK텔레콤' print(company.find('전자')) print(company.index('전자'))
결과값1	2 2
비고	

4. Matching, and Formatting

파이썬으로 문자열 위치 알기 (find, index)-값이 없을 때

파일	소스코드
실습환경	py3_10_basic
소스코드	company= '삼성전자,LG전자,현대자동차,CJ,SK텔레콤' print(company.find('네이버')) print(company.index('네이버'))
결과값1	-1
결과값2	Traceback (most recent call last): File "C:\DEV\PycharmProjects\data_pre_processing/Test.py", line 101, in <module> print(company.index('네이버')) ValueError: substring not found
비고	

4. Matching, and Formatting

파이썬으로 문자열 위치 알기 (find, index)-예외처리

파일	소스코드
실습환경	py3_10_basic

소스코드	company= '삼성전자,LG전자,현대자동차,CJ,SK텔레콤'
	
	try:
	  print(company.index('전자'))
	except ValueError:
	  print("-1")

소스코드	company= '삼성전자,LG전자,현대자동차,CJ,SK텔레콤'
	
	try:
	  print(company.index('네이버'))
	except ValueError:
	  print("-1")

결과값1	2
결과값2	-1
비고	

4. Matching, and Formatting

파이썬으로 문자열 위치 알기 (find, index)-값을 계속 호출

파일	소스코드
실습환경	py3_10_basic

소스코드	company= '삼성전자,LG전자,현대자동차,CJ,SK텔레콤'
	
	index = 0
	while index > -1:
	  index = company.find('전자', index)
	  if index > -1:
	  print(index)
	  index += len('전자')

결과값1	2 7
비고	

4. Matching, and Formatting

파이썬으로 문자열 위치 알기 (find, index)-값을 계속 호출

파일	소스코드
실습환경	py3_10_basic

소스코드	company= '삼성전자,LG전자,현대자동차,CJ,SK텔레콤'
	
	index = 0
	while index > -1:
	  try:
	  index = company.index('전자', index)
	  print(index)
	  index += len('전자')
	  except ValueError:
	  break

결과값1	2 7
비고	

4. Matching, and Formatting

% 기호를 사용한 문자열 형식화

- 파이썬에서는 복잡한 문자열 출력을 위한 문자열 형식화(string formatting)를 지원한다.
  - 문자열을 형식화하는 방법에는 % 기호를 사용한 방식과 format 메서드를 사용한 방식, 그리고 f 문자열을 사용하는 방식이 있다.
  - 문자열 뒤에 % 기호를 붙이고 그 뒤에 다른 값을 붙이면 뒤에 붙은 값이 문자열 안으로 들어간다. "문자열" % 값
  - 이 때 문자열의 어느 위치에 값이 들어가는지를 표시하기 위해 문자열 안에 % 기호로 시작하는 형식지정 문자열(format specification string)을 붙인다.
    - 대표적인 형식지정 문자열은 다음과 같다.

형식지정 문자열	의미
%s	문자열 (String)
%c	문자 한개(character)
%d	정수 (Integer)
%f	부동소수 (floating-point)
%o	8진수
%x	16진수
%%	Literal % (문자 '%s' 자체)

4. Matching, and Formatting

고급 형식지정 문자열

- 형식지정 문자열은 여러가지 숫자 인수를 가질 수도 있다. % 기호 다음에 오는 정수는 값이 인쇄될 때 차지하는 공간의 길이를 뜻한다. 만약 공간의 길이가 인쇄될 값보다 크면 정수가 양수일 때는 값을 뒤로 보내고 공백을 앞에 채우거나 반대로 정수가 음수이면 값을 앞으로 보내고 공백을 뒤에 채운다. 만약 % 기호 다음에 소숫점이 있는 숫자가 오면 점 뒤의 숫자는 실수를 인쇄할 때 소숫점 아래로 그만큼의 숫자만 인쇄하라는 뜻이다.

고급 형식지정 문자열	의미
%20s	전체 20칸을 차지하는 문자열(공백을 앞에 붙인다.)
%-10d	전체 10칸을 차지하는 숫자(공백을 뒤에 붙인다.)
%.5f	부동소수점의 소수점 아래 5자리까지 표시

고급 형식지정 문자열

파일	소스코드
실습환경	py3_10_basic
소스코드	# 방법 1 print("나는 %d살입니다." % 20) # 나는 20살입니다 print("나는 %s을 좋아한다." % "파이썬") # 나는 파이썬을 좋아한다. print("Apple 은 %c로 시작해요." % "A") # Apple 은 A로 시작해요. print("나는 %s살입니다." % 20) # 나는 20살입니다 (%s 로도 정수값 표현 가능) print("나는 %s색과 %s색을 좋아해요." % ("파란", "빨간")) # 나는 파란색과 빨간색을 좋아해요.
결과값1	나는 20살입니다. 나는 파이썬을 좋아한다. Apple 은 A로 시작해요. 나는 20살입니다. 나는 파란색과 빨간색을 좋아해요.
비고	

4. Matching, and Formatting

format 메서드를 사용한 문자열 형식화

- 문자열 내에 중괄호 { } 를 집어 넣고 뒤에서 .format(값1, 값2, ...) 을 입력하면 이 값들이 문자열 내의 중괄호 부분에 들어가게 된다. 이 때 { } 만 넣으면 순서대로 값1, 값2, ... 가 들어가게 되며 만약 {0}, {1} 과 같이 인덱스 값을 의미하는 숫자를 넣게 되면 {0} 위치에는 값1, {1} 위치에는 값2, ... 이런 식으로 들어가게 된다.

파일	소스코드
실습환경	py3_10_basic
소스코드	# 방법 2 print("나는 {}살입니다.".format(20)) # 나는 20살입니다. print("나는 {}색과 {}색을 좋아해요.".format("파란", "빨간")) # 나는 파란색과 빨간색을 좋아해요 print("나는 {0}색과 {1}색을 좋아해요.".format("파란", "빨간")) # 나는 파란색과 빨간색을 좋아해요 print("나는 {1}색과 {0}색을 좋아해요.".format("파란", "빨간")) # 나는 빨간색과 파란색을 좋아해요
결과값1	나는 20살입니다. 나는 파란색과 빨간색을 좋아해요. 나는 파란색과 빨간색을 좋아해요. 나는 빨간색과 파란색을 좋아해요.
비고	

4. Matching, and Formatting

format 메서드를 사용한 문자열 형식화

- 문자열 내에 {이름} 과 같이 넣어두고, 마치 변수를 사용하는 것처럼 .format 내에서 이름과 값을 정의해두면, 그 이름에 해당하는 부분에 값을 집어넣게 된다.

파일	소스코드
실습환경	py3_10_basic
소스코드	# 방법 3 print("나는 {age}살이며, {color}색을 좋아해요.".format(age=20, color="빨간")) # 나는 20살이며, 빨간색을 좋아해요 print("나는 {age}살이며, {color}색을 좋아해요.".format(color="빨간", age=20)) # 나는 20살이며, 빨간색을 좋아해요 (.format 뒤에 순서를 변경해도 괜찮아요)
결과값1	나는 20살이며, 빨간색을 좋아해요. 나는 20살이며, 빨간색을 좋아해요.
비고	

4. Matching, and Formatting

f 문자열

- 파이썬 3.6부터는 f 문자열(f-string)이라는 것을 사용할 수 있다. f 문자열은 문자열의 앞에 f 글자를 붙인 문자열이다. f 문자열에서는 {} 안에 변수의 이름을 바로 사용할 수 있다.

파일	소스코드
실습환경	py3_10_basic
소스코드	# 방법 4 (파이썬 버전 3.6 부터 가능) age = 20 color = "빨간" print(f"나는 {age}살이며, {color}색을 좋아해요.") # 나는 20살이며, 빨간색을 좋아해요.
결과값1	나는 20살이며, 빨간색을 좋아해요.
비고	

5.Quiz

Question 1

다음과 같은 데이터를 정제하여

	A	B	C
0	1	1	1
1	0	0	1
2	1	1	0
3	0	0	0

아래와 같은 결과를 얻도록 하라.

0	A,B,C
1	C
2	A,B
3	

5.Quiz

Answer 1

```
import pandas as pd

# Original DataFrame
df = pd.DataFrame({
    'A': [1, 0, 1, 0],
    'B': [1, 0, 1, 0],
    'C': [1, 1, 0, 0]
})

df2 = df.copy()
for col in df.columns:
    df2[col] = df2[col].replace(to_replace = 1, value = col)
    df2[col] = df2[col].replace(to_replace = 0, value = "")

df2
>>
```

	A	B	C
0	A	B	C
1			C
2	A	B	
3			

5.Quiz

Answer 1

```
"a" + "," + "b"
## a,b

",".join(["a", "b"])
## a,b

",".join(df2.iloc[0])
## 'A,B,C'
>>
'A,B,C'

df2.apply(lambda x: ",".join(x), axis = 1)
## 0    A,B,C
## 1      ,,C
## 2     A,B,
## 3       ,,
## dtype: object
>>
0    A,B,C
1      ,,C
2     A,B,
3       ,,
dtype: object
```

5.Quiz

Answer 1

```
df.columns # 1
## Index(['A', 'B', 'C'], dtype='object')

df.columns[[True, False, True]] # 2
## Index(['A', 'C'], dtype='object')

df.columns[pd.Series([1, 0, 1]).astype("bool")] # 3
## Index(['A', 'C'], dtype='object')
>>
Index(['A', 'C'], dtype='object')

df.apply(lambda x: ",".join(df.columns[x.astype("bool")]), axis = 1)
## 0    A,B,C
## 1        C
## 2      A,B
## 3
## dtype: object
>>
0    A,B,C
1        C
2      A,B
3
dtype: object
```

5.Quiz

비밀번호 생성

사이트 별로 비밀번호를 만들어주는 프로그램을 작성하시오.

예) http://naver.com

1. 규칙1 : http:// 부분은 제외 → naver.com
2. 규칙2 : 처음 만나는 점(.) 이후 부분은 제외 → naver
3. 규칙3 : 남은 글자 중 처음 세 자리 + 글자 갯수 + 글자 내 'e'의 갯수 + '!'로 구성

(nav) (5) (1) (!)

예) 생성된 비밀번호 : nav51!

프로그램을 실행했을 때 나와야 하는 출력값은 다음과 같다.

1. http://naver.com 일 때
→ nav51!
2. http://google.com 일 때
→ goo61!

5.Quiz

해답 1

파일	소스코드
실습환경	py3_10_basic
소스코드	URL = "http://naver.com" # URL = "http://daum.net" # URL = http://google.com http_url = URL.replace("http://", "") # 규칙 1 http_url = http_url[:http_url.index(".")] # 규칙 2 # naver.com 일 때 http_url.index(".") 의 결과는 5 이므로 위 문장은 # http_url = mystr[0:5] 와 같음 http_pass = http_url[:3] + str(len(http_url)) + str(http_url.count("e")) + "!" # 규칙 3 print("{0} 의 비밀번호는 {1} 입니다.".format(URL, http_pass))
결과값1	http://naver.com 의 비밀번호는 nav51! 입니다.
비고	

6. Filtering & Sorting

페어 정렬 (시리즈 sort_values 메서드)

```
import pandas as pd

features = ['hour', 'attendance']
importances = [1, 0]

s = pd.Series(
    importances,
    index=features
)
print(s)
'''
hour        1
attendance    0
dtype: int64
'''


s = s.sort_values(ascending=False)
print(s)
'''
hour        1
attendance    0
dtype: int64
'''
```

6. Filtering & Sorting

페어 정렬 (시리즈 sort_values 메서드)

```
import pandas as pd

features = ['hour', 'attendance']
importances = [1, 0]

df = pd.DataFrame({
    'features': features,
    'importances': importances
})
print(df)
'''
     features  importances
0  attendance           0
1        hour           1
'''


df = df.sort_values('importances', ascending=False)
print(df)
'''
     features  importances
1        hour           1
0  attendance           0
'''
```

7. Finding Outliers and Bad Data

고유한 값

- 열에서 고유한 값 조회
  - unique 메서드 사용해 열에서 고유한 값 조회
  - value_counts 메서드 사용한 고유한 값과 등장 횟수 조회

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 라이브러리를 임포트한다.
	import pandas as pd
	
	# 데이터를 적재한다.
	dataframe = pd.read_csv("datasets/titanic.csv")
	
	# 두 개의 행을 확인한다.
	dataframe.head(2)
	
	# 고유한 값을 찾다.
	dataframe['Gender'].unique()

결과값1	PassengerId Survived Pclass Name Gender Age SibSp Parch Ticket Fare Cabin Embarked 0 1 0 3 Braund, Mr. Owen Harris male 22.0 1 0 A/5 21171 7.2500 NaN S 1 2 1 1 Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0 1 0 PC 17599 71.2833 C85 C
결과값2	array(['male', 'female'], dtype=object)
비고	

7. Finding Outliers and Bad Data

고유한 값

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	# 카운트를 출력한다. dataframe['Gender'].value_counts() # 카운트를 출력한다. dataframe['Pclass'].value_counts() # 고유한 값의 개수를 출력한다. dataframe['Pclass'].nunique() dataframe.nunique()
결과값1	male 577 female 314 Name: Gender, dtype: int64
결과값2	3 491 1 216 2 184 Name: Pclass, dtype: int64
결과값3	3
결과값4	PassengerId 891 Survived 2 Pclass 3 Name 891 Gender 2 Age 88 SibSp 7 Parch 7 Ticket 681 Fare 248 Cabin 147 Embarked 3 dtype: int64
비고	

7. Finding Outliers and Bad Data

고유한 값(결측값Missing values)

- 값이 존재하지 않고 비어있는 상태
- NA(Not Available)또는 NULL 값
  - NA: 결측값
  - NULL: 값이 없다
- 분석 대상의 속성 값이 상당 부분 비어있게되면, 분석 대상 데이터가 충분하지 않은 상태이므로 제대로 된 분석을 수행하기 어려움

7. Finding Outliers and Bad Data

결측값 구분

- MCAR(Missing Completely At Random)
  - 결측값이 관측된 데이터와 관측되지 않은 데이터와 독립적이며 완전 무작위로 발생
  - 데이터 분석 시 편향되지 않아서 결측값이 문제가 되지 않는 경우
  - 데이터가 MCAR인 경우는 거의 없음
- MAR(Missing At Random or MCARMissing Conditionally At Random)
  - 결측값이 조건이 다른 변수에 따라 조건부로 무작위 발생되는 경우
  - 변수의 조건에 따른 결측값이 설명할 수 있는 경우
  - 데이터 분석 시 편향이 발생할 수도 있음
- MNAR(Missing Not At Random)
  - MCAR 또는 MAR이 아닌 데이터
  - 무시할 수 없는 무응답 데이터 (누락된 이유가 존재)
  - 결측값이 무작위가 아니라서 주도면밀한 추가 조사가 필요한 경우

7. Finding Outliers and Bad Data

결측값 처리 방법

- 결측값 데이터 개체 또는 속성의 제거
  - 결측값이 발생한 데이터 개체를 분석 과정에서 제거하거나 해당 속성을 제거
  - 데이터가 충분히 많이 있다면 고려할만한 방법
  - 데이터 내에 결측치를 가진 데이터나 속성이 많은 경우 대부분의 정보가 제거될 수 있음
  - 실제로는 많이 사용하지 않는 방법
- 수동으로 결측값 입력
  - 결측값이 발생한 데이터를 다시 조사 및 수집하여 입력
  - 매우 고비용의 소모적인 방법
  - 결측값이 많은 경우 비현실적인 방법
- 전역상수global constant를 사용한 결측값 입력
  - 단순하고 명확한 방법
  - 예를 들어, 결측값을 0으로 입력
  - 전역상수 값이 분석 결과를 왜곡할 수 있음

7. Finding Outliers and Bad Data

결측값 처리 방법

- 결측값의 무시
  - 알고리즘이나 응용에 따라서는 결측치가 발생한 속성을 무시하고 분석을 수행할 수도 있음
  - 예를 들어, 개체들 사이의 유사성 계산에 있어 많은 수의 속성이 있는 경우 이 중 하나의 속성이 없다면 이를 제외하고 유사성을 계산할 수 있도록 알고리즘을 조정하는 것
  - 하나의 속성 값이 없더라도 유사성을 계산하는데 미치는 영향이 크지 않다면 이러한 방법도 적용 가능
  - 데이터 간 결측값을 가진 속성들이 산재해 있다면 너무 많은 데이터가 제외될 수 있음
  - 속성이 몇 개 없어 하나의 속성이라도 무시하기 힘든 경우라면 이러한 방법의 적용은 좋지 않음
- 결측값의 추정
  - 일반적으로 많이 사용되는 방법
  - 결측값이 발생한 데이터와 유사한 데이터를 사용하여 결측값을 추정하는 방법
  - 결측값을 추정하는 방법에 따라 다양한 형태가 존재

7. Finding Outliers and Bad Data

결측값 추정 방법

- 속성의 평균값을 사용하여 결측값 추정
  - 속성의 평균값을 결측값에 채워넣는 방법
  - 분석 결과를 왜곡시킬 위험성 존재
- 같은 클래스에 속하는 속성의 평균값 사용
  - 주어진 데이터와 같은 클래스(분류)에 속하는 튜플 들의 속성 평균값 사용
  - 동일 유형에 속하는 데이터의 평균값을 사용하므로 왜곡 가능성 줄임
- 가장 가능성이 높은 값으로 결측값 추정
  - 회귀분석, 베이지안Bayesian 기법, 의사결정트리 기법 등의 통계 또는 마이닝 기법을 활용하여 결측값 예측
  - 분석에 의해 가능성이 높은 값을 찾아내는 방법
  - 가장 효과적이고 높은 정확도의 결측값 예측 가능
  - 결측값을 채우기 위한 분석 가설을 세우는 등의 복잡성 존재

7. Finding Outliers and Bad Data

누락된 값 값 조회

파일	소스코드
	준비 py3_10_basic score.xlsx

실습환경

	A	B	C	D	E	F	G	H	I	J
	지원번호	이름	학교	키	국어	영어	수학	과학	사회	SW특기
1	1번	홍길동	강남고	197	90	85	100	95	85	Python
2	2번	박문수	강남고	184	40	35	50	55	25	Java
3	3번	이순신	강남고	168	80	75	70	80	75	Javascript
4	4번	임꺽정	강남고	187	40	60	70	75	80	
5	5번	강백호	강북고	188	15	20	10	35	10	
6	6번	황진희	강북고	202	80	100	95	85	80	C
7	7번	서화담	강북고	188	55	65	45	40	35	PYTHON
8	8번	정난정	강북고	190	100	85	90	95	95	C#

소스코드

결과값1

비고

7. Finding Outliers and Bad Data

누락된 값 값 조회

파일	소스코드
실습환경	준비 py3_10_basic score.xlsx
소스코드	## 누락된 값을 선택하고 두 개의 행을 출력한다. import pandas as pd df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
비고	

7. Finding Outliers and Bad Data

누락된 값 값 조회

- 결측값 조회를 위해 isnull 과 notnull을 사용

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	## 누락된 값을 선택하고 두 개의 행을 출력한다. df.isnull().head()
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 False False False False False False False False False 2번 False False False False False False False False False 3번 False False False False False False False False False 4번 False False False False False False False False True 5번 False False False False False False False False True
비고	

7. Finding Outliers and Bad Data

누락된 값 조회 모듈 작성해 보기

- 파이썬 코딩을 통해 누락된 값 조회 모듈을 작성해 본다.

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	def check_missing_col(dataframe):
	  missing_col = []
	  counted_missing_col = 0
	  for i, col in enumerate(dataframe.columns):
	  missing_values = sum(dataframe[col].isna())
	  is_missing = True if missing_values >= 1 else False
	  if is_missing:
	  counted_missing_col += 1
	  print(f'결측치가 있는 컬럼은: {col}입니다')
	  print(f'해당 컬럼에 총 {missing_values}개의 결측치가 존재한다.')
	  missing_col.append([col, dataframe[col].dtype])
	  if counted_missing_col == 0:
	  print('결측치가 존재하지 않다')
	  return missing_col
	
	missing_col = check_missing_col(df)

결과값1	결측치가 있는 컬럼은: SW특기입니다 해당 컬럼에 총 2개의 결측치가 존재한다.
비고	

7. Finding Outliers and Bad Data

전체 데이터 채우기 fillna

- fillna('없음')을 사용하면 결측치가 NaN 대신 '없음'으로 표기된다.

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.fillna('') # NaN 데이터를 빈 칸으로 채움 df.fillna('없음')
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 5번 강백호 강북고 188 15 20 10 35 10 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
결과값2	이름 학교 키 국어 영어 수학 이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 없음 5번 강백호 강북고 188 15 20 10 35 10 없음 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
비고	

7. Finding Outliers and Bad Data

전체 데이터 채우기 fillna

- 학교 데이터 전체를 NaN 으로 채움
- NaN을 사용하려면 넘파이 라이브러리를 임포트해야 함

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	import numpy as np df['학교'] = np.nan # 학교 데이터 전체를 NaN 으로 채움 df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 NaN 197 90 85 100 95 85 Python 2번 박문수 NaN 184 40 35 50 55 25 Java 3번 이순신 NaN 168 80 75 70 80 75 Javascript 4번 임꺽정 NaN 187 40 60 70 75 80 NaN 5번 강백호 NaN 188 15 20 10 35 10 NaN 6번 황진희 NaN 202 80 100 95 85 80 C 7번 서화담 NaN 188 55 65 45 40 35 PYTHON 8번 정난정 NaN 190 100 85 90 95 95 C#
비고	

7. Finding Outliers and Bad Data

전체 데이터 채우기 fillna

- inplace=True > df에는 반영 안 되어 있음

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.fillna('inplace Ex') df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 inplace Ex 197 90 85 100 95 85 Python 2번 박문수 inplace Ex 184 40 35 50 55 25 Java 3번 이순신 inplace Ex 168 80 75 70 80 75 Javascript 4번 임꺽정 inplace Ex 187 40 60 70 75 80 inplace Ex 5번 강백호 inplace Ex 188 15 20 10 35 10 inplace Ex 6번 황진희 inplace Ex 202 80 100 95 85 80 C 7번 서화담 inplace Ex 188 55 65 45 40 35 PYTHON 8번 정난정 inplace Ex 190 100 85 90 95 95 C#
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 NaN 197 90 85 100 95 85 Python 2번 박문수 NaN 184 40 35 50 55 25 Java 3번 이순신 NaN 168 80 75 70 80 75 Javascript 4번 임꺽정 NaN 187 40 60 70 75 80 NaN 5번 강백호 NaN 188 15 20 10 35 10 NaN 6번 황진희 NaN 202 80 100 95 85 80 C 7번 서화담 NaN 188 55 65 45 40 35 PYTHON 8번 정난정 NaN 190 100 85 90 95 95 C#
비고	

7. Finding Outliers and Bad Data

전체 데이터 채우기 fillna

- inplace=True > df에는 반영 되어 있음

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.fillna('inplace Ex', inplace=True) df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 inplace Ex 197 90 85 100 95 85 Python 2번 박문수 inplace Ex 184 40 35 50 55 25 Java 3번 이순신 inplace Ex 168 80 75 70 80 75 Javascript 4번 임꺽정 inplace Ex 187 40 60 70 75 80 inplace Ex 5번 강백호 inplace Ex 188 15 20 10 35 10 inplace Ex 6번 황진희 inplace Ex 202 80 100 95 85 80 C 7번 서화담 inplace Ex 188 55 65 45 40 35 PYTHON 8번 정난정 inplace Ex 190 100 85 90 95 95 C#
비고	

7. Finding Outliers and Bad Data

일부 데이터 채우기 fillna

- SW특기 데이터 중에서 NaN 에 대해서 채움

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	import pandas as pd df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df df['SW특기'].fillna('확인 중', inplace=True) # SW특기 데이터 중에서 NaN 에 대해서 채움 df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 확인 중 5번 강백호 강북고 188 15 20 10 35 10 확인 중 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
비고	

7. Finding Outliers and Bad Data

데이터 제외하기 dropna

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df df.dropna(inplace=True) # 전체 데이터 중에서 NaN 을 포함하는 데이터 삭제 df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
비고	

7. Finding Outliers and Bad Data

데이터 제외하기 dropna

- axis : index or columns
- how : any or all

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
결과값2	
비고	

7. Finding Outliers and Bad Data

데이터 제외하기 dropna

- axis : index or columns
- how : any or all

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	import numpy as np df['학교'] = np.nan df df.loc['4번'] = [np.nan, np.nan, np.nan, np.nan, np.nan, np.nan, np.nan, np.nan, np.nan] df
결과값1	
결과값2	
비고	

7. Finding Outliers and Bad Data

데이터 제외하기 dropna

- axis : index or columns
- how : any or all

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.dropna(axis='index', how='all') df.dropna(axis='index', how='any') # NaN 이 하나라도 있는 row 삭제
결과값1	
결과값2	
비고	

7. Finding Outliers and Bad Data

데이터 제외하기 dropna

- axis : index or columns
- how : any or all

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df df.dropna(axis='columns') # NaN 이 하나라도 있는 column 삭제
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
결과값2	이름 학교 키 국어 영어 수학 과학 사회 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 2번 박문수 강남고 184 40 35 50 55 25 3번 이순신 강남고 168 80 75 70 80 75 4번 임꺽정 강남고 187 40 60 70 75 80 5번 강백호 강북고 188 15 20 10 35 10 6번 황진희 강북고 202 80 100 95 85 80 7번 서화담 강북고 188 55 65 45 40 35 8번 정난정 강북고 190 100 85 90 95 95
비고	

7. Finding Outliers and Bad Data

데이터 제외하기 dropna

- 데이터 전체가 NaN 인 경우에만 Column 삭제

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['학교'] = np.nan df df.dropna(axis='columns', how='all') # 데이터 전체가 NaN 인 경우에만 Column 삭제
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 NaN 197 90 85 100 95 85 Python 2번 박문수 NaN 184 40 35 50 55 25 Java 3번 이순신 NaN 168 80 75 70 80 75 Javascript 4번 임꺽정 NaN 187 40 60 70 75 80 NaN 5번 강백호 NaN 188 15 20 10 35 10 NaN 6번 황진희 NaN 202 80 100 95 85 80 C 7번 서화담 NaN 188 55 65 45 40 35 PYTHON 8번 정난정 NaN 190 100 85 90 95 95 C#
결과값2	이름 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 197 90 85 100 95 85 Python 2번 박문수 184 40 35 50 55 25 Java 3번 이순신 168 80 75 70 80 75 Javascript 4번 임꺽정 187 40 60 70 75 80 NaN 5번 강백호 188 15 20 10 35 10 NaN 6번 황진희 202 80 100 95 85 80 C 7번 서화담 188 55 65 45 40 35 PYTHON 8번 정난정 190 100 85 90 95 95 C#
비고	

7. Finding Outliers and Bad Data

이상치 탐지Anomaly/Outlier Detection

- 이상치anomalies/outliers 란무엇인가?
  - **데이터의 나머지 부분과 상당히 다른 데이터 요소 집합**
- 자연적 함의Natural implication가 이상한 것은 상대적으로 드문현상
  - **수 많은 데이터가 있는 경우, 수천 개 중에 하나가 자주 발생**
  - **상황이 중요, 예: 7월에 기온이 몹시 추움**
- 중요하거나 방해가 될 수 있음
  - **10 피트(3.048 미터) 키, 2살**
  - **비정상적으로 높은 혈압**

7. Finding Outliers and Bad Data

이상치 탐지Anomaly/Outlier Detection

- 이상치 탐지Anomaly/Outlier Detection
- 이상치의 원인
- 다른 클래스의 데이터
  - **오렌지의 무게를 측정하지만 자몽이 몇 개 섞여 있음**

- 자연 변형Natural variation
  - **비정상적으로 키가 큰 사람들**

- 데이터 오류Data errors
  - **200 파운드 (약 90kg), 2살**

7. Finding Outliers and Bad Data

이상치 탐지Anomaly/Outlier Detection

- 이상치 탐지Anomaly/Outlier Detection
- 노이즈Noise와 이상치Anomalies의 구분
- 노이즈는 잘못되었거나, 임의적이거나, 값이 있거나 오염된 객체
  - **무게가 잘못 기록됨**
  - **오렌지와 섞인 자몽**
- 노이즈가 반드시 비정상적인 값이나 객체를 생성하지는 않음
- 노이즈는 흥미롭지 않음
- 이상치가 노이즈의 결과가 아닌 경우는 흥미로울 수 있음
- 노이즈와 이상치는 관련이 있지만 별개의 개념

7. Finding Outliers and Bad Data

이상치 탐지Anomaly/Outlier Detection

- 이상치 탐지Anomaly/Outlier Detection
- 일반적인 이슈: 속성의 수
- 많은 이상치가 하나의 속성으로 정의
  - **신장Height**
  - **모양Shape**
  - **색깔Color**
- 모든 속성을 사용하여 이상치를 찾기가 어려울 수 있음
  - **노이즈 또는 관련 없는 속성**
  - **객체는 일부 속성과 관련해서만 이상치를 가짐**
- 그러나 어떤 속성에서는 객체가 이상치가 아닐 수도 있음

7. Finding Outliers and Bad Data

이상치 탐지Anomaly/Outlier Detection

- 이상치 탐지Anomaly/Outlier Detection
- 일반적인 이슈: 이상치 점수
- 많은 이상치 탐지 기술은 단지 이진 분류만을 제공
  - **객체가 이상치이거나 그렇지 않음**
  - **특히 분류 기반 접근법에 해당**
- 다른 접근법은 모든 포인트에 점수를 할당
  - **점수는 객체가 비정상인 정도를 측정**
  - **객체의 순위를 매길 수 있음**
- 결국 이진 결정이 필요할 수 있음
  - **이 신용 카드 거래가 신고되어야 하나?**
  - **여전히 점수를 얻는 데 유용**

7. Finding Outliers and Bad Data

이상치 탐지Anomaly/Outlier Detection

- 이상치 탐지Anomaly/Outlier Detection
- 모델 기반 이상치 탐지
- 데이터에 대한 모델을 생성하고 확인
- 비지도Unsupervised
  - **이상치는 잘 맞지 않는 포인트**
  - **이상치는 모델을 왜곡시키는 포인트**
  - **예제:**
    - 통계분포Statistical distribution
    - 클러스터Clusters
    - 회귀분석Regression
    - 기하학Geometric
    - 그래프Graph
- 지도Supervised
  - **이상치는 희귀한 등급으로 간주**
  - **학습 데이터가 필요**

7. Finding Outliers and Bad Data

이상치 탐지Anomaly/Outlier Detection

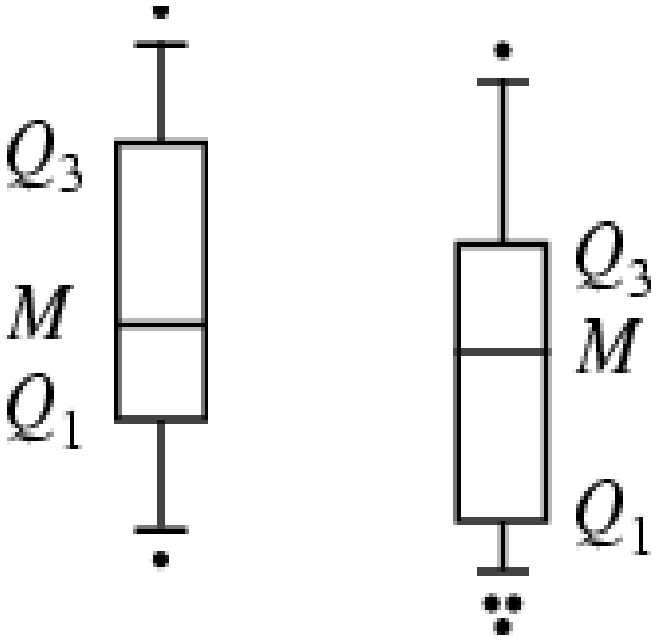
- 이상치 탐지Anomaly/Outlier Detection
- 추가적인 이상치 탐지 기술
- 근접 기반Proximity-based
  - **이상치는 다른 포인트와 멀리 떨어진 지점**
  - **일부 경우 그래픽로도 감지 가능**
- 밀도 기반Density-based
  - **저밀도 포인트는 이상치**
- 패턴 매칭Pattern matching
  - **이례적이지만 중요한 이벤트 또는 객체의 프로필이나 템플릿 생성**
  - **이러한 패턴을 탐지하는 알고리즘은 일반적으로 간단하고 효율적**

7. Finding Outliers and Bad Data

이상치 탐지Anomaly/Outlier Detection

- 이상치 탐지Anomaly/Outlier Detection
- 시각적 접근방법

◦ **박스 플롯Boxplots 또는 분산형 플롯(산포도)scatter plots**



7. Finding Outliers and Bad Data

이상치 탐지Anomaly/Outlier Detection

- 이상치 탐지Anomaly/Outlier Detection
- 이외 통계적 접근 방식
  - **근접성 기반 이상치 탐지**
  - **밀도 기반 이상치 탐지**
  - **군집 기반 이상 치 탐지 방식이 있다**

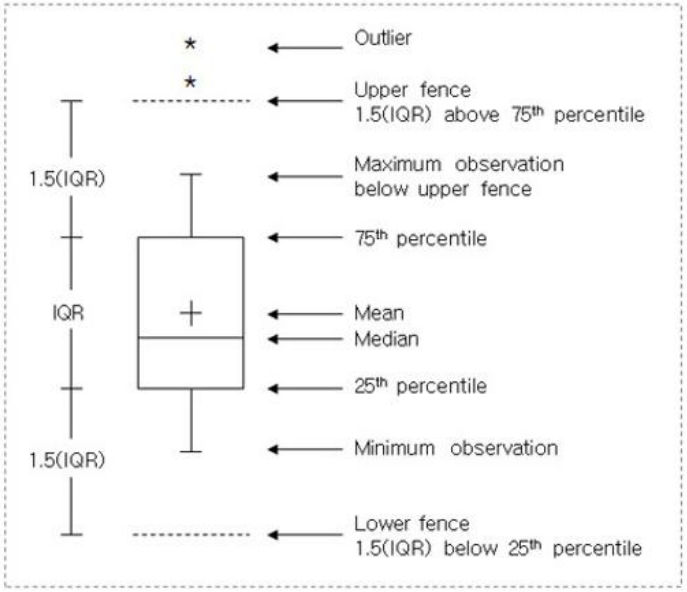
7. Finding Outliers and Bad Data

극단값 절단(trimming)

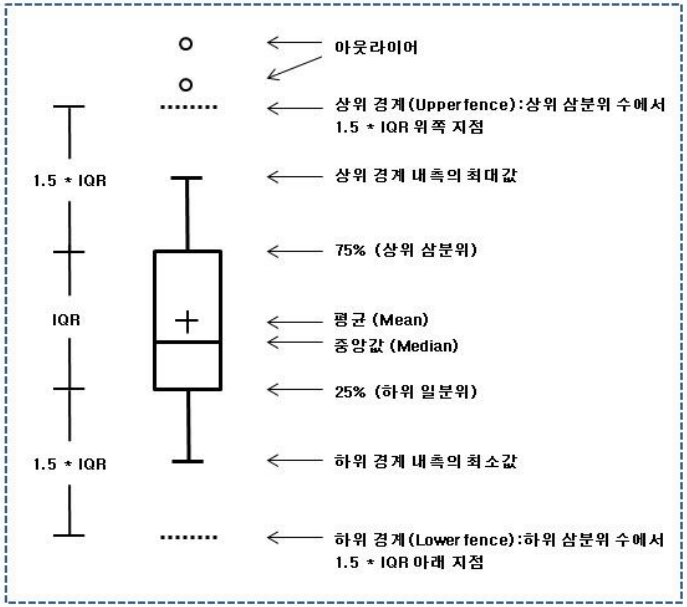
- 기하평균을 이용한 제거
- 상, 하위 5%에 해당되는 데이터 제거

극단값 조정(winsorizing)

- 상한값과 하한값을 벗어나는 값들을 상한 , 하한값 으로 바꾸어 활용하는 방법



※ IQR : Inter Quantile Range



IQR : 사분위 범위

7. Finding Outliers and Bad Data

이상치 다루기

- 개요

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 라이브러리를 임포트한다.
	import numpy as np
	from sklearn.covariance import EllipticEnvelope
	from sklearn.datasets import make_blobs
	
	# 모의 데이터를 만듭니다.
	features, _ = make_blobs(n_samples = 10,
	n_features = 2,
	centers = 1,
	random_state = 1)
	
	# 첫 번째 샘플을 극단적인 값으로 바꿉니다.
	features[0,0] = 10000
	features[0,1] = 10000
	
	# 이상치 감지 객체를 만듭니다.
	outlier_detector = EllipticEnvelope(contamination=.1)
	
	# 감지 객체를 훈련한다.
	outlier_detector.fit(features)
	
	# 이상치를 예측한다.
	outlier_detector.predict(features)

결과값1	array([-1, 1, 1, 1, 1, 1, 1, 1, 1, 1])

7. Finding Outliers and Bad Data

이상치 다루기

- 정규분포를 따른다 가정.
- 이런 가정을 기반으로 데이터를 둘러싼 타원 작성. 타원 안 샘플을 정상치(레이블 1)로 분류, 타원 밖 샘플을 이상치(레이블 -1)로 분류

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 하나의 특성을 만듭니다.
	feature = features[:,0]
	
	# 이상치의 인덱스를 반환하는 함수를 만듭니다.
	def indicies_of_outliers(x):
	  q1, q3 = np.percentile(x, [25, 75])
	  iqr = q3 - q1
	  lower_bound = q1 - (iqr * 1.5)
	  upper_bound = q3 + (iqr * 1.5)
	  return np.where((x > upper_bound) | (x < lower_bound))
	
	# 함수를 실행한다.
	indicies_of_outliers(feature)

결과값1	((array([0], dtype=int64),)
비고	

7. Finding Outliers and Bad Data

이상치 삭제

- 데이터에서 'Bathrooms' 열의 값 중 하나가 116로 매우 이상한 값으로 보입니다. 현실적으로 집에 116개의 화장실이 있다는 것은 말이 되지 않다. 이런 이유로 데이터를 더 자세히 살펴보고 처리해야 할 필요가 있다.
- 가장 가능성 있는 경우는 데이터 입력 오류입니다. 이 오류를 수정하려면 'Bathrooms' 열의 값이 116인 행을 확인하고, 이를 수정하거나 해당 행을 제거하여 데이터를 정리해야 한다.

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 라이브러리를 임포트한다.
	import pandas as pd
	
	# 데이터프레임을 만듭니다.
	houses = pd.DataFrame()
	houses['Price'] = [534433, 392333, 293222, 4322032]
	houses['Bathrooms'] = [2, 3.5, 2, 116]
	houses['Square_Feet'] = [1500, 2500, 1500, 48000]
	
	# 샘플을 필터링한다.
	houses[houses['Bathrooms'] < 20]

결과값1	Price Bathrooms Square_Feet 0 534433 2.0 1500 1 392333 3.5 2500 2 293222 2.0 1500
비고	

7. Finding Outliers and Bad Data

이상치 표시 후 특성으로 포함

- NumPy의 np.where() 함수를 사용하여 조건을 만족하는 경우와 그렇지 않은 경우에 값을 할당하는 데 사용되었다. np.where() 함수는 조건에 따라 배열 또는 시리즈의 값을 대체한다. 여기서는 'Bathrooms' 열의 값이 20 미만인 경우 'Outlier' 열에 0을 할당하고, 그렇지 않은 경우에는 1을 할당하였다. 이를 통해 'Outlier' 열에 이상치(값이 20 이상인 경우)를 감지하기 위한 지표를 추가했다. houses 데이터프레임에는 이제 'Outlier' 열이 추가되었으며, 해당 열의 값은 'Bathrooms' 열의 값에 따라 조건에 따라 할당된 것을 확인할 수 있다. 'Outlier' 열의 값이 0인 경우 'Bathrooms' 값이 20 미만이며, 값이 1인 경우 20 이상인 것으로 해석될 수 있다.

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	# 라이브러리를 임포트한다. import numpy as np # 불리언 조건을 기반으로 특성을 만듭니다. houses["Outlier"] = np.where(houses["Bathrooms"] < 20, 0, 1) # 데이터를 확인한다. houses
결과값1	Price Bathrooms Square_Feet 0 534433 2.0 1500 1 392333 3.5 2500 2 293222 2.0 1500
비고	

7. Finding Outliers and Bad Data

이상치 특성 변환으로 영향력 줄임

- 이 코드는 'Square_Feet' 열의 값들을 하나씩 가져와 각 값에 대한 자연 로그를 계산하여 'Log_Of_Square_Feet' 열에 추가한다. 이를 통해 'Square_Feet' 열의 값에 로그 변환된 값이 들어간 새로운 열이 생성됩니다. 이 변환을 통해 데이터가 비선형 관계를 가질 때 모델의 성능을 향상시킬 수 있다. 로그를 취함으로써 데이터의 분포가 더 정규 분포에 가까워지거나 스케일링이 잘 이루어질 수 있다.

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	# 로그 특성 houses["Log_Of_Square_Feet"] = [np.log(x) for x in houses["Square_Feet"]] # 데이터를 확인한다. houses
결과값1	
비고	

	Price	Bathrooms	Square_Feet	Outlier	Log_Of_Square_Feet
0	534433	2.0	1500	0	7.313220
1	392333	3.5	2500	0	7.824046
2	293222	2.0	1500	0	7.313220
3	4322032	116.0	48000	1	10.778956

7. Finding Outliers and Bad Data

수치 특성 2개의 구간 나누기

- age라는 특성을 생성하고, 이를 Binarizer를 사용하여 threshold를 18로 설정하여 변환하였다. 결과적으로 18 미만의 값은 0으로, 18 이상의 값은 1로 변환되었다. 6과 12는 18보다 작기 때문에 0으로 변환되었고, 20, 36, 65는 18보다 크거나 같기 때문에 1로 변환되었다.

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 라이브러리를 임포트한다.
	import numpy as np
	from sklearn.preprocessing import Binarizer
	
	# 특성을 만듭니다.
	age = np.array([[6],
	[12],
	[20],
	[36],
	[65]])
	
	# Binarizer 객체를 만듭니다.
	binarizer = Binarizer(threshold=18)
	
	# 특성을 변환한다.
	binarizer.fit_transform(age)

결과값1	
비고	

	Price	Bathrooms	Square_Feet	Outlier	Log_Of_Square_Feet
0	534433	2.0	1500	0	7.313220
1	392333	3.5	2500	0	7.824046
2	293222	2.0	1500	0	7.313220
3	4322032	116.0	48000	1	10.778956

7. Finding Outliers and Bad Data

수치 특성 2개의 구간 나누기

- np.digitize() 함수는 숫자형 데이터를 구간(bins)에 따라 나누는 데 사용됩니다. 이 함수는 주어진 배열의 각 요소를 해당하는 구간의 인덱스로 변환한다.
- 주어진 코드에서는 age라는 특성을 np.digitize() 함수를 사용하여 구간을 설정하고, 각 값이 속하는 구간의 인덱스를 반환하도록 하였다. bins=[20, 30, 64]로 설정되어 있으므로 20보다 작은 값은 0, 20 이상 30 미만의 값은 1, 30 이상 64 미만의 값은 2, 64 이상의 값은 3으로 변환됩니다.
- 6과 12는 20보다 작기 때문에 0에 해당하는 구간으로 변환되었다.
- 20은 20 이상 30 미만의 구간에 속하므로 1에 해당하는 값으로 변환되었다.
- 36은 30 이상 64 미만의 구간에 속하므로 2에 해당하는 값으로 변환되었다.
- 65는 64 이상의 구간에 속하므로 3에 해당하는 값으로 변환되었다.

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 특성을 나눕니다.
	np.digitize(age, bins=[20,30,64])

결과값1	array([[0], [0], [1], [2], [3]], dtype=int64)
비고	

7. Finding Outliers and Bad Data

수치 특성 2개의 구간 나누기

- age 특성을 np.digitize() 함수를 사용하여 구간을 설정하고, 각 값이 속하는 구간의 인덱스를 반환한다. 이때 bins=[20,30,64]로 설정되어 있으며, right=True로 설정되어 각 구간의 오른쪽 경계값을 포함한다.
- 6과 12는 20보다 작기 때문에 0에 해당하는 구간으로 변환됩니다.
- 20은 20 이상 30을 포함하는 구간에 속하므로 0에 해당하는 값으로 변환됩니다.
- 36은 30 이상 64를 포함하는 구간에 속하므로 3에 해당하는 값으로 변환됩니다.
- 65는 64보다 크거나 같기 때문에 3에 해당하는 값으로 변환됩니다.
- 따라서, right=True로 설정할 경우, 각 값이 오른쪽 경계값을 포함하는 구간의 인덱스를 반환한다.

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 특성을 나눕니다.
	np.digitize(age, bins=[20,30,64], right=True)

결과값1	array([[0], [0], [0], [2], [3]], dtype=int64)
비고	

7. Finding Outliers and Bad Data

수치 특성 2개의 구간 나누기

- np.digitize() 함수를 사용하여 age라는 특성을 구간을 나눌 때, bins=[18]와 같이 구간을 설정하면 구간은 [18)과 [18, +∞)로 나뉘게 됩니다.
- 여기서 bins=[18]는 하나의 경계값을 가진 리스트로, 18을 기준으로 두 개의 구간으로 나뉩니다.
- 6과 12는 18 미만의 구간에 속하므로 0에 해당하는 값으로 변환됩니다.
- 20, 36, 65는 18 이상의 구간에 속하므로 1에 해당하는 값으로 변환됩니다.

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	# 특성을 나눕니다.
	np.digitize(age, bins=[18])

결과값1	array([[0], [0], [1], [1], [1]], dtype=int64)
비고	

8. 그룹화

원복

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	import pandas as pd df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df df.groupby('학교')
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
결과값2	<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001DA9CC97160>
비고	

8. 그룹화

계산 가능한 데이터들의 평균값

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.groupby('학교').get_group('강남고')
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN8번 정난정 강북고 190 100 85 90 95 95 C#
비고	

8. 그룹화

그룹화를 한 뒤

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.groupby('학교').size() # 각 그룹의 크기 df.groupby('학교').size()['강남고'] # 학교로 그룹화를 한 뒤에 능남고에 해당하는 데이터의 수 df.groupby('학교')['키'].mean() # 학교로 그룹화를 한 뒤에 키의 평균 데이터 df.groupby('학교')[['국어', '영어', '수학']].mean() # 학교로 그룹화를 한 뒤에 국어, 영어, 수학 평균 데이터
결과값1	학교 강남고 4 강북고 4 dtype: int64
결과값2	4
결과값3	학교 강남고 184.0 강북고 192.0
결과값4	국어 영어 수학 학교 강남고 62.5 63.75 72.5 강북고 62.5 67.50 60.0
비고	

8. 그룹화

학교별, 학년별 평균 데이터

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['학년'] = [3, 3, 2, 1, 1, 3, 2, 2] # 학년 Column 추가 df
	df.groupby(['학교', '학년']).sum()
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 학년 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 3 2번 박문수 강남고 184 40 35 50 55 25 Java 3 3번 이순신 강남고 168 80 75 70 80 75 Javascript 2 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 1 5번 강백호 강북고 188 15 20 10 35 10 NaN 1 6번 황진희 강북고 202 80 100 95 85 80 C 3 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 2 8번 정난정 강북고 190 100 85 90 95 95 C# 2
결과값3	키 국어 영어 수학 과학 사회 학교 학년 강남고 1 187 40 60 70 75 80 2 168 80 75 70 80 75 3 381 130 120 150 150 110 강북고 1 188 15 20 10 35 10 2 378 155 150 135 135 130 3 202 80 100 95 85 80
비고	

8. 그룹화

학교별, 학년별 평균 데이터

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df = df.drop(["이름", "학교", "SW특기"], axis=1) df df.groupby('학년').mean().sort_values('키') df.groupby('학년').mean().sort_values('키', ascending=False)
결과값1	키 국어 영어 수학 과학 사회 학년 2 182.000000 78.333333 75.000000 68.333333 71.666667 68.333333 1 187.500000 27.500000 40.000000 40.000000 55.000000 45.000000 3 194.333333 70.000000 73.333333 81.666667 78.333333 63.333333
결과값2	키 국어 영어 수학 과학 사회 학년 3 194.333333 70.000000 73.333333 81.666667 78.333333 63.333333 1 187.500000 27.500000 40.000000 40.000000 55.000000 45.000000 2 182.000000 78.333333 75.000000 68.333333 71.666667 68.333333
비고	

9. Quiz

###01_문제

- 다음은 대한민국 영화 중에서 관객 수가 가장 많은 상위 8개의 데이터입니다.
  - 주어진 코드를 이용하여 퀴즈를 풀어보시오.

'영화' : ['명량', '극한직업', '신과함께-죄와 벌', '국제시장', '괴물', '도둑들', '7번방의 선물', '암살'],
'개봉 연도' : [2014, 2019, 2017, 2014, 2006, 2012, 2013, 2015],
'관객 수' : [1761, 1626, 1441, 1426, 1301, 1298, 1281, 1270], # (단위 : 만 명)
'평점' : [8.88, 9.20, 8.73, 9.16, 8.62, 7.64, 8.83, 9.10

9. Quiz

###01_문제

1. 전체 데이터 중에서 '영화' 정보만 출력하시오

2. 전체 데이터 중에서 '영화', '평점' 정보를 출력하시오.

3. 2015년 이후에 개봉한 영화 데이터 중에서 '영화', '개봉 연도' 정보를 출력하시오.

4. 주어진 계산식을 참고하여 '추천 점수' Column 을 추가하시오.

5. 전체 데이터를 '개봉 연도' 기준 내림차순으로 출력하시오.

9. Quiz

답

- 데이터 로드

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	import pandas as pd
	data = {
	'영화' : ['명량', '극한직업', '신과함께-죄와 벌', '국제시장', '괴물', '도둑들', '7번방의 선물', '암살'],
	'개봉 연도' : [2014, 2019, 2017, 2014, 2006, 2012, 2013, 2015],
	'관객 수' : [1761, 1626, 1441, 1426, 1301, 1298, 1281, 1270], # (단위 : 만 명)
	'평점' : [8.88, 9.20, 8.73, 9.16, 8.62, 7.64, 8.83, 9.10]
	}
	df = pd.DataFrame(data)
	df

결과값1	영화 개봉 연도 관객 수 평점 0 명량 2014 1761 8.88 1 극한직업 2019 1626 9.20 2 신과함께-죄와 벌 2017 1441 8.73 3 국제시장 2014 1426 9.16 4 괴물 2006 1301 8.62 5 도둑들 2012 1298 7.64 6 7번방의 선물 2013 1281 8.83 7 암살 2015 1270 9.10
비고	

9. Quiz

답

- 전체 데이터 중에서 '영화' 정보만 출력하시오.

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['영화']
결과값1	0 명량 1 극한직업 2 신과함께-죄와 벌 3 국제시장 4 괴물 5 도둑들 6 7번방의 선물 7 암살 Name: 영화, dtype: object
비고	

9. Quiz

답

- 전체 데이터 중에서 '영화', '평점' 정보를 출력하시오.

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df[['영화', '평점']]
결과값1	영화 평점 0 명량 8.88 1 극한직업 9.20 2 신과함께-죄와 벌 8.73 3 국제시장 9.16 4 괴물 8.62 5 도둑들 7.64 6 7번방의 선물 8.83 7 암살 9.10
비고	

9. Quiz

답

- 2015년 이후에 개봉한 영화 데이터 중에서 '영화', '개봉 연도' 정보를 출력하시오.

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.loc[df['개봉 연도'] >= 2015, ['영화', '개봉 연도']]
결과값1	영화 개봉 연도 1 극한직업 2019 2 신과함께-죄와 벌 2017 7 암살 2015
비고	

9. Quiz

답

- 주어진 계산식을 참고하여 '추천 점수' Column 을 추가하시오.
  - > 추천 점수 = (관객수 * 평점) // 100
  - 예) 첫 번째 영화인 '명량'의 경우, 추천 점수 = (관객수 1761 * 평점 8.88) // 100 = 156

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['추천 점수'] = (df['관객 수'] * df['평점']) // 100 df
결과값1	영화 개봉 연도 관객 수 평점 추천 점수 0 명량 2014 1761 8.88 156.0 1 극한직업 2019 1626 9.20 149.0 2 신과함께-죄와 벌 2017 1441 8.73 125.0 3 국제시장 2014 1426 9.16 130.0 4 괴물 2006 1301 8.62 112.0 5 도둑들 2012 1298 7.64 99.0 6 7번방의 선물 2013 1281 8.83 113.0 7 암살 2015 1270 9.10 115.0
비고	

9. Quiz

답

- 전체 데이터를 '개봉 연도' 기준 내림차순으로 출력하시오.

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.sort_values('개봉 연도', ascending=False)
결과값1	영화 개봉 연도 관객 수 평점 추천 점수 1 극한직업 2019 1626 9.20 149.0 2 신과함께-죄와 벌 2017 1441 8.73 125.0 7 암살 2015 1270 9.10 115.0 0 명량 2014 1761 8.88 156.0 3 국제시장 2014 1426 9.16 130.0 6 7번방의 선물 2013 1281 8.83 113.0 5 도둑들 2012 1298 7.64 99.0 4 괴물 2006 1301 8.62 112.0
비고	

9. Quiz

02_ADS_sample_1.csv

Attention

뉴욕 airBnB : https://www.kaggle.com/ptoscano230382/air-bnb-ny-2019

DataUrl = 'datasets/AB_NYC_2019.csv'

9. Quiz

Question 1

데이터를 로드하고 상위 5개 컬럼을 출력하라

Answer 1

```
df= pd.read_csv('datasets/AB_NYC_2019.csv')

df.head(5)
>>
```

id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	
	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month		
	calculated_host_listings_count	availability_365						
0	2539	Clean & quiet apt home by the park		2787	John	Brooklyn	Kensington	
	40.64749	-73.97237	Private room	149	1	9	2018-10-19	0.21	6
	365							
1	2595	Skylit Midtown Castle		2845	Jennifer	Manhattan	Midtown	40.75362	-
73.98377	Entire home/apt	225	1	45	2019-05-21	0.38	2	355	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !		4632	Elisabeth	Manhattan	Harlem		
	40.80902	-73.94190	Private room	150	3	0	NaN	NaN	1
	365							
3	3831	Cozy Entire Floor of Brownstone		4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-
73.95976	Entire home/apt	89	1	270	2019-07-05	4.64	1	194	
4	5022	Entire Apt: Spacious Studio/Loft by central park		7192	Laura	Manhattan	East Harlem		
	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19	0.10	1
	0							

9. Quiz

Question 2

데이터의 각 host_name의 빈도수를 구하고 host_name으로 정렬하여 상위 5개를 출력하라

Answer 2

```
Ans = df.groupby('host_name').size()

Ans

>>
```

```
host_name
'Cil                        1
(Ari) HENRY LEE             1
(Email hidden by Airbnb)    6
(Mary) Haiy                 1
-TheQueensCornerLot         1
                           ..
단비                         1
빈나                         1
소정                         2
진                          1
현선                         1
Length: 11452, dtype: int64
```

9. Quiz

Question 3

데이터의 각 host_name의 빈도수를 구하고 빈도수 기준 내림차순 정렬한 데이터 프레임을 만들어라. 빈도수 컬럼은 counts로 명명하라

Answer 3

```
Ans = df.groupby('host_name').size().\
            to_frame().rename(columns={0:'counts'}).\
            sort_values('counts',ascending=False)

Ans.head(5)

>>
```

	counts
host_name	
Michael	417
David	403
Sonder (NYC)	327
John	294
Alex	279

9. Quiz

Question 4

neighbourhood_group의 값에 따른 neighbourhood컬럼 값의 갯수를 구하여라

Answer 4

```
Ans = df.groupby(['neighbourhood_group','neighbourhood'], as_index=False).size()
Ans
>>
```

	neighbourhood_group	neighbourhood	size
0	Bronx	Allerton	42
1	Bronx	Baychester	7
2	Bronx	Belmont	24
3	Bronx	Bronxdale	19
4	Bronx	Castle Hill	9
...	...	...	...
216	Staten Island	Tottenville	7
217	Staten Island	West Brighton	18
218	Staten Island	Westerleigh	2
219	Staten Island	Willowbrook	1
220	Staten Island	Woodrow	1

221 rows × 3 columns

9. Quiz

Question 5

neighbourhood_group의 값에 따른 neighbourhood컬럼 값 중 neighbourhood_group그룹의 최댓값들을 출력하라

Answer 5

```
Ans= df.groupby(['neighbourhood_group','neighbourhood'], as_index=False).size()\
                .groupby(['neighbourhood_group'], as_index=False).max()
Ans
```

>>

	neighbourhood_group	neighbourhood	size
0	Bronx	Woodlawn	70
1	Brooklyn	Windsor Terrace	3920
2	Manhattan	West Village	2658
3	Queens	Woodside	900
4	Staten Island	Woodrow	48

9. Quiz

Question 6

n eighbourhood_group 값에 따른 price값의 평균, 분산, 최대, 최소 값을 구하여라

Answer 6

```
Ans = df[['neighbourhood_group','price']].groupby('neighbourhood_group').agg(['mean','var','max','min'])
Ans

>>
```

	price			
	mean	var	max	min
neighbourhood_group				
Bronx	87.496792	11386.885081	2500	0
Brooklyn	124.383207	34921.719135	10000	0
Manhattan	196.875814	84904.159185	10000	0
Queens	99.517649	27923.130227	10000	10
Staten Island	114.812332	77073.088342	5000	13

9. Quiz

Question 7

neighbourhood_group 값에 따른 reviews_per_month 평균, 분산, 최대, 최소 값을 구하여라

Answer 7

```
Ans =
df[['neighbourhood_group','reviews_per_month']].groupby('neighbourhood_group').agg(['mean','var','max','min'])
Ans
>>
```

	reviews_per_month			
	mean	var	max	min
neighbourhood_group				
Bronx	1.837831	2.799878	10.34	0.02
Brooklyn	1.283212	2.299040	14.00	0.01
Manhattan	1.272131	2.651206	58.50	0.01
Queens	1.941200	4.897848	20.94	0.01
Staten Island	1.872580	2.840895	10.12	0.02

9. Quiz

Question 8

neighbourhood 값과 neighbourhood_group 값에 따른 price 의 평균을 구하라

Answer 8

```
Ans = df.groupby(['neighbourhood','neighbourhood_group']).price.mean()
Ans
```

>>

```
neighbourhood     neighbourhood_group
Allerton          Bronx                87.595238
Arden Heights     Staten Island        67.250000
Arrochar          Staten Island       115.000000
Arverne           Queens              171.779221
Astoria           Queens              117.187778
                                  ...
Windsor Terrace   Brooklyn            138.993631
Woodhaven         Queens               67.170455
Woodlawn          Bronx                60.090909
Woodrow           Staten Island       700.000000
Woodside          Queens               85.097872
Name: price, Length: 221, dtype: float64
```

9. Quiz

Question 9

neighbourhood 값과 neighbourhood_group 값에 따른 price 의 평균을 계층적 indexing 없이 구하라

Answer 9

```
Ans = df.groupby(['neighbourhood','neighbourhood_group']).price.mean().unstack()
Ans

>>
```

neighbourhood_group		Bronx	Brooklyn	Manhattan	Queens	Staten Island
neighbourhood						
Allerton	87.595238	NaN	NaN	NaN	NaN	
Arden Heights	NaN	NaN	NaN	NaN	67.25	
Arrochar	NaN	NaN	NaN	NaN	115.00	
Arverne	NaN	NaN	NaN	171.779221	NaN	
Astoria	NaN	NaN	NaN	117.187778	NaN	
...	...	...	...	...	...	
Windsor Terrace		NaN	138.993631	NaN	NaN	NaN
Woodhaven	NaN	NaN	NaN	67.170455	NaN	
Woodlawn	60.090909	NaN	NaN	NaN	NaN	
Woodrow	NaN	NaN	NaN	NaN	700.00	
Woodside	NaN	NaN	NaN	85.097872	NaN	

221 rows × 5 columns

9. Quiz

unstack

특정 레벨의 인덱스를 열로 이동시켜 분석이나 시각화 등의 작업을 수월하게 할 수 있다.

```
import pandas as pd

# MultiIndex를 가진 예제 데이터프레임
index = pd.MultiIndex.from_tuples([('A', 1), ('A', 2), ('B', 1), ('B', 2)], names=['letter', 'number'])
Df_11 = pd.DataFrame({'value': [10, 20, 30, 40]}, ind
Df_11
>>
```

		value
letter	number	
A	1	10
	2	20
B	1	30
	2	40

```
df_unstacked = df.unstack(level='letter')
df_unstacked
```

```
Df_11 _unstacked = Df_11.unstack(level='letter')
Df_11 _unstacked
>>
```

	value	
letter	A	B
number		
1	10	30
2	20	40

9. Quiz

Question 10

neighbourhood 값과 neighbourhood_group 값에 따른 price 의 평균을 계층적 indexing 없이 구하고 nan 값은 -999값으로 채워라

Answer 10

```
Ans = df.groupby(['neighbourhood','neighbourhood_group']).price.mean().unstack().fillna(-999)
Ans
```

>>

neighbourhood_group	Bronx	Brooklyn	Manhattan	Queens	Staten Island
neighbourhood					
Allerton	87.595238	-999.000000	-999.0	-999.000000	-999.00
Arden Heights	-999.000000	-999.000000	-999.0	-999.000000	67.25
Arrochar	-999.000000	-999.000000	-999.0	-999.000000	115.00
Arverne	-999.000000	-999.000000	-999.0	171.779221	-999.00
Astoria	-999.000000	-999.000000	-999.0	117.187778	-999.00
...	...	...	...	...	...
Windsor Terrace	-999.000000	138.993631	-999.0	-999.000000	-999.00
Woodhaven	-999.000000	-999.000000	-999.0	67.170455	-999.00
Woodlawn	60.090909	-999.000000	-999.0	-999.000000	-999.00
Woodrow	-999.000000	-999.000000	-999.0	-999.000000	700.00
Woodside	-999.000000	-999.000000	-999.0	85.097872	-999.00

221 rows × 5 columns

9. Quiz

Question 11

데이터중 neighbourhood_group 값이 Queens값을 가지는 데이터들 중 neighbourhood 그룹별로 price값의 평균, 분산, 최대, 최소값을 구하라

Answer 11

```
Ans =
df[df.neighbourhood_group=='Queens'].groupby(['neighbourhood']).price.agg(['mean','var','max','min'])
Ans

>>
```

neighbourhood	mean	var	max	min
Arverne	171.779221	37383.411141	1500	35
Astoria	117.187778	122428.811196	10000	25
Bay Terrace	142.000000	6816.400000	258	32
Bayside	157.948718	166106.470985	2600	30
Bayswater	87.470588	2330.889706	230	45
Belle Harbor	171.500000	8226.571429	350	85
Bellerose	99.357143	3093.016484	240	42

9. Quiz

Question 12

데이터중 neighbourhood_group 값에 따른 room_type 컬럼의 숫자를 구하고 neighbourhood_group 값을 기준으로 각 값의 비율을 구하여라

Answer 12

```
Ans =
df[['neighbourhood_group','room_type']].groupby(['neighbourhood_group','room_type']).size().
unstack()
Ans.loc[:,:] = (Ans.values /Ans.sum(axis=1).values.reshape(-1,1))
Ans
>>
```

room_type	Entire home/apt	Private room	Shared room
neighbourhood_group			
Bronx	0.347388	0.597617	0.054995
Brooklyn	0.475478	0.503979	0.020543
Manhattan	0.609344	0.368496	0.022160
Queens	0.369926	0.595129	0.034945
Staten Island	0.471850	0.504021	0.024129

9. Quiz

03_ADS_sample_1.csv

Attention

카드이용데이터 : https://www.kaggle.com/sakshigoyal7/credit-card-customers

DataUrl = 'datasets/BankChurnersUp.csv'

9. Quiz

Question 13

데이터를 로드하고 데이터 행과 열의 갯수를 출력하라

Answer 13

```
df =pd.read_csv('datasets/BankChurnersUp.csv',index_col=0)
df.head()
```

>>

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	
	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon			Contacts_Count_12_mon		Credit_Limit		
	Total_Revolving_Bal		Avg_Open_To_Buy		Total_Amt_Chng_Q4_Q1		Total_Trans_Amt			
0	768805383	Existing Customer	45	M	3	High School	Married	$60K-$80K		
	Blue	39	5	1	3	12691.0	777	11914.0	1.335	
	1144									
1	818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K		
	Blue	44	6	1	2	8256.0	864	7392.0	1.541	
	1291									
2	713982108	Existing Customer	51	M	3	Graduate	Married	$80K-$120K		
	Blue	36	4	1	0	3418.0	0	3418.0	2.594	
	1887									
3	769911858	Existing Customer	40	F	4	High School	Unknown	Less than $40K		
	Blue	34	3	4	1	3313.0	2517	796.0	1.405	
	1171									
4	709106358	Existing Customer	40	M	3	Uneducated	Married	$60K-$80K		
	Blue	21	5	1	0	4716.0	0	4716.0	2.175	
	816									

9. Quiz

Question 13

데이터를 로드하고 데이터 행과 열의 갯수를 출력하라

Answer 13

```
# DataFrame의 행과 열 개수 출력
print(f"행의 수: {df.shape[0]}")
print(f"열의 수: {df.shape[1]}")
```

>>

```
행의 수: 10127
열의 수: 18
```

```
print(df.shape)  # (행의 수, 열의 수)를 튜플로 출력
```

>>

```
(10127, 18)
```

9. Quiz

Question 14

Income_Category의 카테고리를 map 함수를 이용하여 다음과 같이 변경하여 newIncome 컬럼에 매핑하라 Unknown : N
Less than $40K : a
$40K - $60K : b
$60K - $80K : c
$80K - $120K : d
$120K +’ : e

9. Quiz

Answer 14

```
dic = {
    'Unknown'        : 'N',
    'Less than $40K' : 'a',
    '$40K - $60K'    : 'b',
    '$60K - $80K'    : 'c',
    '$80K - $120K'   : 'd',
    '$120K +'        : 'e'
}

df['newIncome']  =df.Income_Category.map(lambda x: dic[x])

Ans = df['newIncome']

Ans
>>
```

```
0      c
1      a
2      d
3      a
4      c
      ..
10122   b
10123   b
10124   a
10125   b
10126   a
Name: newIncome, Length: 10127, dtype: object
```

9. Quiz

Question 15

Income_Category의 카테고리를 apply 함수를 이용하여 다음과 같이 변경하여 newIncome 컬럼에 매핑하라 Unknown : N
Less than $40K : a
$40K - $60K : b
$60K - $80K : c
$80K - $120K : d
$120K +’ : e

9. Quiz

Answer 15

```
def changeCategory(x):
    if x =='Unknown':
        return 'N'
    elif x =='Less than $40K':
        return 'a'
    elif x =='$40K - $60K':
        return 'b'
    elif x =='$60K - $80K':
        return 'c'
    elif x =='$80K - $120K':
        return 'd'
    elif x =='$120K +' :
        return 'e'

df['newIncome']  =df.Income_Category.apply(changeCategory)

Ans = df['newIncome']

Ans
>>

0      c
1      a
2      d
3      a
4      c
      ..
10122   b
10123   b
10124   a
10125   b
10126   a
Name: newIncome, Length: 10127, dtype: object
```

9. Quiz

Question 16

Customer_Age의 값을 이용하여 나이 구간을 AgeState 컬럼으로 정의하라. (0~9 : 0 , 10~19 :10 , 20~29 :20 … 각 구간의 빈도수를 출력하라

Answer 16

```
df['AgeState']  = df.Customer_Age.map(lambda x: x//10 *10)

Ans = df['AgeState'].value_counts().sort_index()

Ans
>>
```

```
AgeState
20     195
30    1841
40    4561
50    2998
60     530
70       2
Name: count, dtype: int64
```

9. Quiz

Question 17

Education_Level의 값중 Graduate단어가 포함되는 값은 1 그렇지 않은 경우에는 0으로 변경하여 newEduLevel 컬럼을 정의하고 빈도수를 출력하라

Answer 17

```
df['newEduLevel'] = df.Education_Level.map(lambda x : 1 if 'Graduate' in x else 0)
Ans = df['newEduLevel'].value_counts()

Ans
>>
```

```
newEduLevel
0    6483
1    3644
Name: count, dtype: int64
```

9. Quiz

Question 18

Credit_Limit 컬럼값이 4500 이상인 경우 1 그외의 경우에는 모두 0으로 하는 newLimit 정의하라. newLimit 각 값들의 빈도수를 출력하라

Answer 18

```
df['newLimit'] = df.Credit_Limit.map(lambda x : 1 if x>=4500 else 0)
Ans = df['newLimit'].value_counts()

Ans

>>
```

```
newLimit
1    5096
0    5031
Name: count, dtype: int64
```

9. Quiz

Question 19

Marital_Status 컬럼값이 Married 이고 Card_Category 컬럼의 값이 Platinum인 경우 1 그외의 경우에는 모두 0으로 하는 newState 컬럼을 정의하라. newState의 각 값들의 빈도수를 출력하라

9. Quiz

Answer 19

```
def check(x):
    if x.Marital_Status =='Married' and x.Card_Category =='Platinum':
        return 1
    else:
        return 0


df['newState'] = df.apply(check,axis=1)

Ans  = df['newState'].value_counts()

Ans

>>
```

```
newState
0    10120
1        7
Name: count, dtype: int64
```

9. Quiz

Question 20

Gender 컬럼값 M인 경우 male F인 경우 female로 값을 변경하여 Gender 컬럼에 새롭게 정의하라. 각 value의 빈도를 출력하라

9. Quiz

Answer 20

```
def changeGender(x):
    if x =='M':
        return 'male'
    else:
        return 'female'
df['Gender'] = df.Gender.apply(changeGender)
Ans = df['Gender'].value_counts()

Ans
>>
```

```
Gender
female    5358
male      4769
Name: count, dtype: int64
```

THANK YOU.