The Python -
전처리 프로젝트

The Python - 전처리 프로젝트

제목 차례

1장. 전처리 프로젝트

1절. 프로젝트 개요

1.1. 대회에 사용하는 파일

- 데이터 다운로드 주소 : http://bit.ly/30PGMky
- CUST_DATA.csv : 고객 데이터
- CALIM_DATA.csv : 보험 청구 데이터

1.2. 요구사항

다음의 요구사항대로 전처리를 진행해야 합니다. 전처리를 수행한 후의 일부 요약 정보는 주어진 엑셀파일에 입력해야 합니다. 원문자(예: ①)는 요약정보 입력양식의 번호입니다.

1) 고객의 나이(AGE) 전처리

- 고객의 나이(AGE)를 연령대로 변환하세요. 예를 들면 47세는 4가 됩니다.
- 변환 후 AGE의 ①평균과 ②표준편차(표본표준편차)를 소수점 이하 6자리까지 엑셀파일에 입력하세요.

2) 지역(CTPR)을 숫자로 인코딩

- 지역이 없는 결측치 행은 삭제하세요. 그리고 ③삭제 후 행의 수를 엑셀파일에 입력하세요.
- 텍스트 데이터를 단순 레이블 인코딩 하세요. 그리고 레이블 인코딩한 값들의 ④평균과 ⑤표준편차를 엑셀파일에 입력하세요.(소수점 6자리까지)
- 레이블 인코딩 한 데이터를 이용해서 원 핫 인코딩 하세요. 그 결과에서 서울 지역의 ⑥평균과 ⑦표준편차를 엑셀파일에 입력하세요.(소수점 6자리까지)
- 원 핫 인코딩한 열의 이름은 지역의 이름이 되도록 하세요.
- 고객 데이터와 지역을 원-핫 인코딩 한 데이터를 하나의 데이터 프레임으로 합치세요. 합치기 전에 결측치 행이 삭제되어 인덱스가 바뀌었다는 것에 주의 하세요.
- 지역(CTPR) 열을 삭제하세요.

3) 직업에서 직업 코드만 빼내기

- 직업(OCCP_GRP_1) 열의 값에서 가장 첫 문자(직업코드)만 빼내서 저장하세요.
- 숫자로 바꾸는 것이 아닙니다. 문자 그대로 두세요.(직업코드가 없는 데이터는 'n'으로 표시되도록 하세요.)

- ⑧가장 많은 직업의 코드를 엑셀파일에 입력하세요.
- ⑨가장 많은 직업 코드의 개수를 엑셀파일에 입력하세요.

4) 고객별 평균 입원일

- 보험 청구 데이터(CLAIM_DATA.csv)에서 각 고객별 평균 입원 일수를 저장한 열을 고객 테이블에 추가하세요.
- 입원일을 저장한 열의 이름은 VLID_HOSP_OTDA입니다.
- 새로 만들어져야 할 열 이름은 HOSP_DAY입니다.
- HOSP_DAY 열의 ⑩평균과 ⑪표준편차를 입력하세요.
- ⑫고객들 중에서 가장 많이 입원한 고객의 일수는 며칠인지 엑셀파일에 입력하세요.

5) 데이터 구조 변경

- 사고원인(ACCI_DVSN)과 청구코드(DMND_RESN_CODE)를 이용한 파생변수를 만드세요.
- 사고원인 구분 코드(재해(1), 교통재해(2), 질병(3))와 지급청구의 원인이 되는 사유 코드(사망(01), 입원(02), 통원(03), 장해(04), 수술(05), 진단(06), 치료(07), 해지/무효(09))를 조합하여 고객별로 사고구분_청구사유 횟수를 계산해야 합니다.
- 새로 만들어진 열의 이름은 1_1, 1_2 ... 형식여야 합니다.(2_7, 2_9는 없음)
- 다음 그림은 이해를 돕기 위한 그림입니다.

	CUST_ID	1_1	1_2	1_3	1_4	1_5	1_6	1_7	1_9	2_1	2_2	2_3	2_4	2_5	2_6	3_1	3_2	3_3	3_4	3_5	3_6	3_7	3_9
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
2	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

- 이렇게 만들어진 사고유형_청구코드의 조합 테이블은 고객테이블과 병합되어야 합니다.
- 교통재해(2)와 입원(2)의 횟수에 대한 ⑬평균과 ⑭표준편차를 엑셀파일에 소수점 6째자리까지 입력하세요.

6) 열 삭제

- 다음 열(column)들을 삭제하세요.
- 'FP_CAREER', 'DIVIDED_SET', 'RESI_TYPE_CODE', 'OCCP_GRP_2', 'MATE_OCCP_GRP_1', 'MATE_OCCP_GRP_2'

7) 결과 파일 저장

- 결과 파일을 *홍길동*_result.csv 파일로 저장하세요.(홍길동 대신 본인의 이름을 사용)
- 인덱스는 파일에 저장되지 않아야 합니다.

- CUST_ID열은 정수형 이어야 합니다.
- 저장하는 파일의 인코딩은 utf-8입니다.

1.3. 제출 파일

제출할 3개입니다. 파일에 본인의 이름이 입력되어 있어야 합니다. 아래의 예에서 '홍길동'은 본인의 이름으로 대체되어야 합니다.

- 최종 전처리 된 csv 파일 : 예) *홍길동*_result.csv
- 요약정보가 입력된 엑셀파일 : 예) *홍길동*_요약정보.xlsx
- 주피터노트북 파일 : 예) *홍길동*_전처리.ipynv

1.4. 최종 결과 파일 형식(예상)

다음은 예상되는 최종 결과 파일의 형식입니다.

	CUST_ID	SIU_CUST_YN	SEX	AGE	RESI_COST	CUST_RGST	OCCP_GRP_1	TOTALPREM	MINCRDT	MAXCRDT	WEDD_YN	CHLD_CNT	LTBN_CHLD
0	1	N	2.0	4.0	21111.0	199910.0	3	146980441.0	NaN	NaN	Y	2.0	
1	2	N	1.0	5.0	40000.0	199910.0	3	94600109.0	1.0	6.0	Y	2.0	
2	3	N	1.0	6.0	0.0	199910.0	5	18501269.0	NaN	NaN	N	0.0	
3	4	N	2.0	6.0	12861.0	199910.0	2	317223657.0	2.0	99.0	N	0.0	
4	5	N	2.0	5.0	0.0	199910.0	2	10506072.0	8.0	8.0	Y	3.0	

MAX_PAYM_YM	MAX_PRM	CUST_INCM	RCBASE_HSHD_INCM	JPBASE_HSHD_INCM	강원	경기	경남	경북	광주	대구	대전	부산	서울	세종	울산	인천	전남	전북	제주	충남	충북
200811.0	319718.0	4879.0	10094.0	11337.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
200012.0	341341.0	6509.0	9143.0	6509.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
200305.0	131300.0	4180.0	0.0	4180.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
201009.0	1493184.0	NaN	4270.0	5914.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
200807.0	166760.0	3894.0	0.0	8885.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

HOSP_DAY	1_1	1_2	1_3	1_4	1_5	1_6	1_7	1_9	2_1	2_2	2_3	2_4	2_5	2_6	3_1	3_2	3_3	3_4	3_5	3_6	3_7	3_9
1.250000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
2.666667	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
16.000000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0.000000	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25.000000	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

1.5. 요약 정보 입력 양식

다음 표에 전처리 과정에서 계산한 요약정보를 입력해야 합니다. 표준편차는 모표준편차가 아닌 표본표준편차를 입력해야 합니다.

번호	요약 정보	값
1	나이를 연령대로 변화 후 평균	
2	나이를 연령대로 변환 후 표준편차	
3	지역 결측치 제거 후 행의 수	
4	단순 레이블 인코딩 후 평균	
5	단순 레이블 인코딩 후 표준편차	
6	원 핫 인코딩 후 서울 지역의 평균	
7	원 핫 인코딩 후 서울 지역의 표준편차	
8	가장 많은 직업의 코드	
9	가장 많은 직업의 수	
10	HOSP_DAY 열의 평균	
11	HOSP_DAY 열의 표준편차	
12	HOSP_DAY 열의 MAX	
13	교통재해(2)와 입원(2)의 횟수에 대한 평균	
14	교통재해(2)와 입원(2)의 횟수에 대한 표준편차	

2절. 데이터 불러오기

2.1. 라이브러리 로드 및 기본 설정

필요한 라이브러리를 로드 합니다.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

브라우저의 화면 전체를 사용할 수 있도록 스타일을 지정합니다.

```
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
```

데이터프레임을 화면에 출력할 때 보여야 하는 최대 열의 개수를 지정합니다.

```
pd.options.display.max_columns = 999
```

2.2. 데이터 불러오기

데이터파일을 불러옵니다.

```
cust_df = pd.read_csv("CUST_DATA.csv", encoding="utf-16")
cust_df.head()
```

	CUST_ID	DIVIDED_SET	SIU_CUST_YN	SEX	AGE	RESI_COST	RESI_TYPE_CODE	FP_CAREER	CUST_RGST	CTPR	OCCP_GRP_1	OCCP_GRP_2	TOTALPREM	MINCRDT	MAXCRDT	WEDD_YN	MATE_OCCP_GRP_1	MATE_OCCP_GRP_2	CHLD_CNT	LTBN_CHLD_AGE	MAX
0	1	1	N	2	47	21111	20.0	N	199910.0	충북	3.사무직		146980441.0	NaN	NaN	Y	3.사무직	2자산업 종사자	2.0	13.0	
1	2	1	N	1	53	40000	20.0	N	199910.0	서울	3.사무직		94600109.0	1.0	6.0	Y	1.주부	주부	2.0	17.0	
2	3	1	N	1	60	0	NaN	N	199910.0	서울	5.서비스	2자산업 종사자	18501269.0	NaN	NaN	N	NaN	NaN	0.0	0.0	
3	4	1	N	2	64	12861	40.0	Y	199910.0	경기	2.자영업	3자산업 종사자	317223957.0	2.0	99.0	N	NaN	NaN	0.0	0.0	
4	5	1	N	2	54	0	NaN	Y	199910.0	광주	2.자영업	3자산업 종사자	10596072.0	8.0	8.0	Y	3.사무직	고위 공무원	3.0	19.0	

데이터의 요약 정보를 출력합니다.

```
cust_df.describe(include='all')
```

	CUST_ID	DIVIDED_SET	SIU_CUST_YN	SEX	AGE	RESI_COST	RESI_TYPE_CODE	FP_CAREER	CUST_RGST	CTPR	OCCP_GRP_1	OCCP_GRP_2	TOTALPREM	MINCRDT	MAXCRDT	WEDD_YN	MATE_OCCP_GRP_1	MATE_OCCP_GRP_2
count	22400.000000	22400.000000	20937	22400.000000	22400.000000	22400.000000	21146.000000	22400	21944.000000	21779	21805	21805	1.660900e+04	12924.000000	12924.000000	21927	10573	10573
unique	NaN	NaN	2	NaN	NaN	NaN	NaN	2	NaN	17	8	25	NaN	NaN	NaN	2	8	24
top	NaN	NaN	N	NaN	NaN	NaN	NaN	N	NaN	경기	8.기타	주부	NaN	NaN	NaN	Y	1.주부	주부
freq	NaN	NaN	18801	NaN	NaN	NaN	NaN	21182	NaN	5200	4879	4837	NaN	NaN	NaN	12010	2394	2402
mean	11200.500000	1.000045	NaN	1.565134	44.734866	15914.413973	25.765251	NaN	198924.250638	NaN	NaN	NaN	2.998266e+07	5.838904	19.480656	NaN	NaN	NaN
std	6466.467351	0.271368	NaN	0.495750	15.445707	14963.317519	20.691869	NaN	19989.329112	NaN	NaN	NaN	4.897438e+07	3.515198	32.363151	NaN	NaN	NaN
min	1.000000	1.000000	NaN	1.000000	2.000000	0.000000	11.000000	NaN	101.000000	NaN	NaN	NaN	5.000000e+02	0.000000	0.000000	NaN	NaN	NaN
25%	5600.750000	1.000000	NaN	1.000000	34.000000	6732.750000	20.000000	NaN	200306.000000	NaN	NaN	NaN	6.665459e+06	6.000000	6.000000	NaN	NaN	NaN
50%	11200.500000	1.000000	NaN	2.000000	46.000000	12222.000000	20.000000	NaN	200306.000000	NaN	NaN	NaN	1.622463e+07	6.000000	6.000000	NaN	NaN	NaN
75%	16800.250000	1.000000	NaN	2.000000	56.000000	20868.000000	30.000000	NaN	200402.000000	NaN	NaN	NaN	3.479439e+07	6.000000	7.000000	NaN	NaN	NaN
max	22400.000000	2.000000	NaN	2.000000	89.000000	305555.000000	99.000000	NaN	201602.000000	NaN	NaN	NaN	1.840052e+09	99.000000	99.000000	NaN	NaN	NaN

```
cust_df.shape
```

```
(22400, 25)
```

3절. 데이터 전처리

3.1. 나이를 연령대로 변환

람다식을 이용해서 나이를 연령대로 변환합니다. map() 함수는 시리즈의 각 요소에 함수를 적용시킵니다.

```
cust_df.AGE = cust_df.AGE.map(lambda x: int(x//10))
cust_df.head()
```

	CUST_ID	DIVIDED_SET	SIU_CUST_YN	SEX	AGE	RESI_COST
0	1	1	N	2	4	21111
1	2	1	N	1	5	40000
2	3	1	N	1	6	0
3	4	1	N	2	6	12861
4	5	1	N	2	5	0

나이를 연령대로 변환한 데이터를 이용해서 요약정보를 출력합니다.

```
cust_df.describe()
```

	CUST_ID	DIVIDED_SET	SEX	AGE	RESI_COST
count	22400.000000	22400.000000	22400.000000	22400.000000	22400.000000
mean	11200.500000	1.080045	1.565134	4.014509	15914.413973
std	6466.467351	0.271368	0.495750	1.577022	14963.317519
min	1.000000	1.000000	1.000000	0.000000	0.000000
25%	5600.750000	1.000000	1.000000	3.000000	6732.750000
50%	11200.500000	1.000000	2.000000	4.000000	12222.000000
75%	16800.250000	1.000000	2.000000	5.000000	20988.000000
max	22400.000000	2.000000	2.000000	8.000000	305555.000000

3.2. 지역 텍스트(CTPR)를 숫자로 인코딩

1) 결측치 행 삭제

결측치 행의 인덱스를 출력해 봅니다.

```
drop_index = cust_df.loc[cust_df.CTPR.isnull()].index
drop_index
```

```
Int64Index([  82,  125,  164,  169,  170,  172,  174,  189,  211,  218,
            ...
            22129, 22167, 22186, 22191, 22223, 22235, 22269, 22310, 22341, 22358],
           dtype='int64', length=621)
```

결측치 행이 아닌 행들만 빼내서 저장합니다.

```
cust_df = cust_df.loc[~cust_df.CTPR.isnull()]
cust_df.shape
```

```
(21779, 25)
```

전처리한 데이터를 임시 저장합니다.

```
cust_df.to_csv("CUST_DATA_1-1.csv", index=False, encoding="utf-8-sig")
```

2) 레이블 인코딩

지역 텍스트를 숫자로 인코딩 합니다. 그러기 위해서 LabelEncoder 클래스를 이용합니다.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le.fit(cust_df.CTPR)
```

```
LabelEncoder()
```

```
le.classes_
```

```
array(['강원', '경기', '경남', '경북', '광주', '대구', '대전', '부산',
'서울', '세종', '울산',
       '인천', '전남', '전북', '제주', '충남', '충북'], dtype=object)
```

```
ctpr = le.transform(cust_df.CTPR)
ctpr
```

```
array([16,  8,  8, ...,  1,  1,  1])
```

```
ctpr.mean(), ctpr.std() # 넘파이의 표준편차는 모표준편차임
```

```
(6.2645208687267555, 4.676745153961355)
```

```
ctpr_df = pd.DataFrame(data=np.c_[ctpr])
print(ctpr_df.mean())
print(ctpr_df.std(ddof=0))    # ddof(델타자유도)가 0이면 모표준편차
print(ctpr_df.std())          # ddof(델타자유도)가 1이면 표본표준편차
```

```
0    6.264521
dtype: float64
0    4.676745
dtype: float64
0    4.676853
dtype: float64
```

3) 원-핫 인코딩

레이블 인코딩한 데이터를 이용해서 원-핫 인코딩하기 위해 OneHotEncoder 클래스를 사용합니다.

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
enc.fit(ctpr.reshape(-1,1))
```

```
OneHotEncoder(categorical_features=None, categories=None, drop=None,
              dtype=<class 'numpy.float64'>, handle_unknown='error',
              n_values=None, sparse=True)
```

```
ctpr_onehot = enc.transform(ctpr.reshape(-1,1))
ctpr_onehot
```

```
<21779x17 sparse matrix of type '<class 'numpy.float64'>'
    with 21779 stored elements in Compressed Sparse Row format>
```

원-핫 인코딩한 데이터를 데이터프래임으로 만들고 요약정보을 출력합니다

```
ctpr_df = pd.DataFrame(ctpr_onehot.toarray(), columns=le.classes_)
ctpr_df.describe()
```

	강원	경기	경남	경북	광주	대구	대전	부산	서울	세종	울산	인천	전남	전북	제주	충남	충북
count	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000
mean	0.026218	0.238762	0.073236	0.045043	0.039625	0.045824	0.023417	0.069746	0.168373	0.001883	0.032508	0.067267	0.051380	0.044171	0.008724	0.035217	0.028606
std	0.159786	0.426337	0.260529	0.207404	0.195082	0.209108	0.151227	0.254724	0.374206	0.043348	0.177350	0.250489	0.220776	0.205480	0.092996	0.184333	0.166699
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

이렇게 만들어진 ctpr_df 데이터프레임의 인덱스는 0부터 21779 까지 갖지만...

```
ctpr_df.index
```

```
RangeIndex(start=0, stop=21779, step=1)
```

cust_df 데이터프레임은 중간의 행들이 삭제되어서 인덱스가 순서대로 되어 있지 않습니다.

```
cust_df.index
```

```
Int64Index([    0,     1,     2,     3,     4,     5,     6,     7,     8,     9,
            ...
            22390, 22391, 22392, 22393, 22394, 22395, 22396, 22397, 22398, 22399],
           dtype='int64', length=21779)
```

그래서 고객데이터와 원-핫 인코딩한 cust_df 데이터와 ctpr_df 데이터를 결합(Concatenation)하기 전에 cust_df 데이터프레임의 인덱스를 초기화해야 합니다.

```
cust_df.reset_index(drop=True, inplace=True)
cust_df = pd.concat([cust_df, ctpr_df], axis=1)
cust_df.head()
```

	CUST_ID	DIVIDED_SET	SIU_CUST_YN	SEX	AGE	RESI_COST	RESI_TYPE_CODE	FP_CAREER	CUST_RGST	CTPR	OCCP_GRP_1	OCCP_GRP_2	TOTALPREM	MINCRDT	MAXCRDT	WEDD_YN	MATE_OCCP_GRP_1	MATE_OCCP_GRP_2	CHLD_CNT	LTBN_CHLD_AGE	MAX
0	1.0	1.0	N	2.0	4.0	21111.0	20.0	N	199910.0	충북	3.사무직	사무직	146980441.0	NaN	NaN	Y	3.사무직	2차산업 종사자	2.0	13.0	
1	2.0	1.0	N	1.0	5.0	40000.0	20.0	N	199910.0	서울	3.사무직	사무직	94600109.0	1.0	6.0	Y	1.주부	주부	2.0	17.0	
2	3.0	1.0	N	1.0	6.0	0.0	NaN	N	199910.0	서울	5.서비스	2차산업 종사자	18501269.0	NaN	NaN	N	NaN	NaN	0.0	0.0	
3	4.0	1.0	N	2.0	6.0	12861.0	40.0	Y	199910.0	경기	2.자영업	3차산업 종사자	317223657.0	2.0	99.0	N	NaN	NaN	0.0	0.0	
4	5.0	1.0	N	2.0	5.0	0.0	NaN	Y	199910.0	광주	2.자영업	3차산업 종사자	10506972.0	8.0	8.0	Y	3.사무직	고위 공무원	3.0	19.0	

원본 데이터 열(CTPR)은 삭제합니다.

```
cust_df.drop("CTPR", axis=1, inplace=True)
cust_df.head()
```

	CUST_ID	DIVIDED_SET	SIU_CUST_YN	SEX	AGE	RESI_COST	RESI_TYPE_CODE	FP_CAREER	CUST_RGST	OCCP_GRP_1	OCCP_GRP_2	TOTALPREM	MINCRDT	MAXCRDT	WEDD_YN	MATE_OCCP_GRP_1	MATE_OCCP_GRP_2	CHLD_CNT	LTBN_CHLD_AGE	MAX_PAYM
0	1.0	1.0	N	2.0	4.0	21111.0	20.0	N	199910.0	3.사무직	사무직	146980441.0	NaN	NaN	Y	3.사무직	2차산업 종사자	2.0	13.0	2008
1	2.0	1.0	N	1.0	5.0	40000.0	20.0	N	199910.0	3.사무직	사무직	94600109.0	1.0	6.0	Y	1.주부	주부	2.0	17.0	2000
2	3.0	1.0	N	1.0	6.0	0.0	NaN	N	199910.0	5.서비스	2차산업 종사자	18501269.0	NaN	NaN	N	NaN	NaN	0.0	0.0	2003
3	4.0	1.0	N	2.0	6.0	12861.0	40.0	Y	199910.0	2.자영업	3차산업 종사자	317223657.0	2.0	99.0	N	NaN	NaN	0.0	0.0	2010
4	5.0	1.0	N	2.0	5.0	0.0	NaN	Y	199910.0	2.자영업	3차산업 종사자	10506972.0	8.0	8.0	Y	3.사무직	고위 공무원	3.0	19.0	2009

전처리한 데이터를 임시 저장합니다.

```
cust_df.to_csv("CUST_DATA_1-2.csv", index=False, encoding="utf-8-sig")
```

3.3. 직업 코드 빼내기

직업 코드는 '3.사무직' 형식으로 되어 있습니다. 이 데이터에서 첫 번째 문자(직업코드)만 빼내야 합니다. 빼낸 직업코드를 숫자로 바꾸는 것이 아닙니다. 문자 그대로 두세요. 직업코드가 없는 데이터는 'n'으로 표시되도록 하세요.

map() 함수를 이용해서 직업코드 문자열에서 첫 문자만 빼냅니다.

```
cust_df.OCCP_GRP_1 = cust_df.OCCP_GRP_1.map(lambda x: str(x)[0])
```

요약 통계량을 통해서 가장 많은 직업의 코드와 해당 직업코드의 수를 확인합니다.

```
cust_df.describe(include='all')
```

	CUST_ID	DIVIDED_SET	SIU_CUST_YN	SEX	AGE	RESI_COST	RESI_TYPE_CODE	FP_CAREER	CUST_RGST	OCCP_GRP_1	OCCP_GRP_2	TOTALPREM	MINCRDT	MAXCRDT	WEDD_YN	MATE_OCCP_GRP_1	MATE_OCCP_GRP_2	CHI
count	21779.000000	21779.000000	20032	21779.000000	21779.000000	21779.000000	20717.000000	21779	21779.000000	22372	21710	1.626700e+04	12792.000000	12792.000000	21762	10544	10544	21762
unique	NaN	NaN	2	NaN	NaN	NaN	NaN	2	NaN	9	25	NaN	NaN	NaN	2	8	24	
top	NaN	NaN	N	NaN	NaN	NaN	NaN	N	NaN	8	주부	NaN	NaN	NaN	Y	1.주부	주부	
freq	NaN	NaN	18254	NaN	NaN	NaN	NaN	20577	NaN	4868	4834	NaN	NaN	NaN	11978	2387	2395	
mean	11194.977910	1.680215	NaN	1.569723	3.987832	16967.469527	25.726553	NaN	198910.253969	NaN	NaN	3.925541e+07	5.638024	19.581223	NaN	NaN	NaN	0
std	6460.805265	0.271532	NaN	0.495126	1.587412	14854.162250	20.629856	NaN	17052.758201	NaN	NaN	4.927764e+07	3.529302	32.464282	NaN	NaN	NaN	0
min	1.000000	1.000000	NaN	1.000000	0.000000	0.000000	11.000000	NaN	101.000000	NaN	NaN	5.000000e+32	0.000000	0.000000	NaN	NaN	NaN	0
25%	5598.500000	1.000000	NaN	1.000000	3.000000	6865.000000	20.000000	NaN	200306.000000	NaN	NaN	6.782680e+06	6.000000	6.000000	NaN	NaN	NaN	0
50%	11178.000000	1.000000	NaN	2.000000	4.000000	12261.000000	20.000000	NaN	200306.000000	NaN	NaN	1.641480e+07	6.000000	6.000000	NaN	NaN	NaN	0
75%	16791.500000	1.000000	NaN	2.000000	5.000000	21111.000000	20.000000	NaN	200401.000000	NaN	NaN	3.514479e+07	6.000000	7.000000	NaN	NaN	NaN	2
max	22400.000000	2.000000	NaN	2.000000	8.000000	305555.000000	99.000000	NaN	201692.000000	NaN	NaN	1.840052e+09	99.000000	99.000000	NaN	NaN	NaN	6

전처리한 데이터를 임시 저장합니다.

```
cust_df.to_csv("CUST_DATA_1-3.csv", index=False, encoding="utf-8-sig")
```

4절. 파생변수 추가

4.1. 고객별 평균 입원일 집계

보험 청구 데이터에서 입원 일을 저장한 열은 VLID_HOSP_OTDA입니다. 각 고객별로 청구 데이터를 분석해서 고객별 평균 입원일을 집계하고 고객 데이터프레임에 추가해야 합니다. 새로 만들어져야 할 열 이름은 HOSP_DAY입니다

사용할 데이터는 청구 데이터(CLAIM_DATA) 입니다.

```
claim_df = pd.read_csv("CLAIM_DATA.csv", encoding="utf-16")
claim_df.head()
```

	CUST_ID	POLY_NO	ACCI_OCCP_GRP1	ACCI_OCCP_GRP2	CHANG_FP_YN	CNTT_RECP_SQNO	RECP_DATE	ORIG_RESN_DATE	RESN_DATE	CRNT_PROG_DVSN	ACCI_DVSN	CAUS_CODE	CAUS_CODE_DTAL	DSAS_NAME	DMND_RESN_CODE	DMND_RSCD_SQNO	HOSP_
0	5936	1365	8.기타	학생	Y	2006011200001	20060112	20060109	20060109	11	1	W21	#	하악부표재성 손상	3	2	
1	5936	6151	8.기타	학생	Y	2006011200002	20060112	20060109	20060109	11	1	W21	#	하악부표재성 손상	3	2	
2	5936	10364	8.기타	학생	Y	2006011200003	20060112	20060109	20060109	11	1	W21	#	하악부표재성 손상	3	2	
3	1043	1247	1.주부	주부	N	2006011200004	20060112	20060105	20060105	23	3	A09	#	기타세균성장자감염|손목및손의2도화상|피부염	2	1	
4	8545	11236	1.주부	주부	Y	2006011200005	20060112	20060110	20060110	11	3	I83	#	좌측하지정맥류혈관발거술|레이저소작술	5	1	

고객의 아이디로 그룹핑 합니다. 이렇게 만들어진 데이터는 DataFrameGroupBy 객체입니다.

```
claim_df_g = claim_df.groupby(claim_df.CUST_ID)
claim_df_g
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at
0x000002A20579B198>
```

그룹핑한 데이터를 평균을 계산하고 VLID_HOSP_OTDA열 만 빼냅니다.

```
claim_df_g_mean = claim_df_g.mean()
claim_df_hospday = claim_df_g_mean.loc[:,["VLID_HOSP_OTDA"]]
claim_df_hospday.columns
```

```
Index(['VLID_HOSP_OTDA'], dtype='object')
```

행 인덱스를 초기화하고 행 인덱스는 행의 이름이 되도록 합니다.

```
claim_df_hospday.reset_index(level=0, inplace=True)
```

이렇게 해서 만들어진 데이터는 고객별 평균입원일을 저장한 데이터프레임입니다.

```
claim_df_hospday.columns = ['CUST_ID', 'HOSP_DAY']
claim_df_hospday.head()
```

	CUST_ID	HOSP_DAY
0	1	1.250000
1	2	2.666667
2	3	16.000000
3	4	0.000000
4	5	25.000000

고객 데이터와 고객별 평균입원일을 저장한 데이터 프레임을 병합합니다.

```
cust_df = pd.merge(cust_df, claim_df_hospday, how="inner")
cust_df.head()
```

	CUST_ID	DIVIDED_SET	SIU_CUST_YN	SEX	AGE	RESI_COST	RESI_TYPE_CODE	FP_CAREER	CUST_RGST	OCCP_GRP_1	OCCP_GRP_2	TOTALPREM	MINCRDT	MAXCRDT	WEDD_YN	MATE_OCCP_GRP_1	MATE_OCCP_GRP_2	CHLD_CNT	LTBN_CHLD_AGE	MAX_PAYM_
0	1.0	1.0	N	2.0	4.0	21111.0	20.0	N	199910.0	3	사무직	1469980441.0	NaN	NaN	Y	3.사무직	2차산업 종사자	2.0	13.0	2008
1	2.0	1.0	N	1.0	5.0	40000.0	20.0	N	199910.0	3	사무직	94600100.0	1.0	6.0	Y	1.주부	주부	2.0	17.0	2000
2	3.0	1.0	N	1.0	6.0	0.0	NaN	N	199910.0	5	2차산업 종사자	18501269.0	NaN	NaN	N	NaN	NaN	0.0	0.0	2009
3	4.0	1.0	N	2.0	6.0	12861.0	40.0	Y	199910.0	2	3차산업 종사자	3172239357.0	2.0	99.0	N	NaN	NaN	0.0	0.0	2010
4	5.0	1.0	N	2.0	5.0	0.0	NaN	Y	199910.0	2	3차산업 종사자	10506072.0	8.0	8.0	Y	3.사무직	그외 공무원	3.0	19.0	2008

병합된 데이터의 열 정보를 출력합니다.

```
cust_df.columns
```

```
Index(['CUST_ID', 'DIVIDED_SET', 'SIU_CUST_YN', 'SEX', 'AGE', 'RESI_COST',
       'RESI_TYPE_CODE', 'FP_CAREER', 'CUST_RGST', 'OCCP_GRP_1', 'OCCP_GRP_2',
       'TOTALPREM', 'MINCRDT', 'MAXCRDT', 'WEDD_YN', 'MATE_OCCP_GRP_1',
       'MATE_OCCP_GRP_2', 'CHLD_CNT', 'LTBN_CHLD_AGE', 'MAX_PAYM_YM',
       'MAX_PRM', 'CUST_INCM', 'RCBASE_HSHD_INCM', 'JPBASE_HSHD_INCM', '강원',
       '경기', '경남', '경북', '광주', '대구', '대전', '부산', '서울', '세종',
'울산', '인천', '전남',
       '전북', '제주', '충남', '충북', 'HOSP_DAY'],
      dtype='object')
```

요약정보를 출력합니다. HOSP_DAY 열의 평균과 표준편차 그리고 MAX값을 확인합니다.

```
cust_df.describe()
```

	CUST_ID	DIVIDED_SET	SEX	AGE	RESI_COST	RESI_TYPE_CODE	CUST_RGST	TOTALPREM	MINCRDT	MAXCRDT	CHLD_CNT	LTBN_CHLD_AGE	MAX_PAYM_YM	MAX_PRM	CUST_INCM	RCBASE_HSHD_INCM	JPBASE_HSHD_INC
count	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	20717.000000	21779.000000	1.626700e+04	12792.000000	12792.000000	21762.000000	21762.000000	15583.000000	1.558300e+04	16999.000000	21779.000000	21556.0000
mean	11194.877910	1.080215	1.569723	3.987832	16067.469627	25.726553	199910.253960	3.025541e+07	5.838024	19.581223	0.713905	8.656833	200887.957914	4.836190e+05	2694.432849	4822.017540	5208.0839
std	6460.805265	0.271632	0.495126	1.567412	14954.162250	20.629956	17052.758201	4.927764e+07	3.529302	32.464292	0.955101	11.601555	522.195038	1.338032e+06	2191.392914	3054.927893	2713.4916
min	1.000000	1.000000	1.000000	0.000000	0.000000	11.000000	101.000000	5.000000e+02	0.000000	0.000000	0.000000	0.000000	199801.000000	5.000000e+00	0.000000	0.000000	0.0000
25%	5598.500000	1.000000	1.000000	3.000000	6865.000000	20.000000	200306.000000	8.782680e+06	6.000000	6.000000	0.000000	0.000000	200405.000000	1.138185e+05	0.000000	3081.000000	3561.0000
50%	11178.000000	1.000000	2.000000	4.000000	12261.000000	20.000000	200306.000000	1.641480e+07	6.000000	6.000000	0.000000	0.000000	200902.000000	2.282620e+05	3425.000000	4848.000000	4689.5000
75%	16791.500000	1.000000	2.000000	5.000000	21111.000000	20.000000	200401.000000	3.514479e+07	6.000000	7.000000	2.000000	19.000000	201403.000000	4.886120e+05	4393.000000	6638.500000	6854.0000
max	22400.000000	2.000000	2.000000	8.000000	305555.000000	99.000000	201602.000000	1.840052e+09	99.000000	99.000000	6.000000	35.000000	207110.000000	5.689262e+07	12687.000000	19829.000000	25872.0000

4.2. 피벗 테이블 생성

사고원인(ACCI_DVSN)과 청구코드(DMND_RESN_CODE)를 이용해서 파생변수를 추가해야 합니다.

다음 코드는 사용자 아이디와, 사고원인, 청그코드 데이터만 빼냅니다.

```
claim_df_acci = claim_df.loc[:, ["CUST_ID", "ACCI_DVSN",
"DMND_RESN_CODE"]]
claim_df_acci["value"] = 1
claim_df_acci.head()
```

	CUST_ID	ACCI_DVSN	DMND_RESN_CODE	value
0	5936	1	3	1
1	5936	1	3	1
2	5936	1	3	1
3	1043	3	2	1
4	8545	3	5	1

피벗테이블을 만듭니다. columns 열의 값이 새로운 피벗테이블의 열 인덱스가 되며

```
cust_claim_df = claim_df_acci.pivot_table(index=["CUST_ID"],
                                          columns=["ACCI_DVSN",
"DMND_RESN_CODE"],
                                          values=["value"],
                                          aggfunc='sum', fill_value=0)

cust_claim_df.head()
```

	value																				
ACCI_DVSN	1									2						3					
DMND_RESN_CODE	1	2	3	4	5	6	7	9		1	2	3	4	5	6	1	2	3	4	5	6	7	9
CUST_ID																							
1	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	4	0	0	0	0	0
2	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	3	0	0	0	0	0
3	0	0	0	0	0	0	0	0		1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	9	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0		1	0	0	0	0	0	0	0	0	0	0	0	0	0

col_level에 따라 reset_index 결과가 어떻게 달라지는지 확인해 봅내다.

```
cust_claim_df.reset_index(level=["CUST_ID"], col_level=1).head()
```

		value																						
ACCI_DVSN	CUST_ID	1									2							3						
DMND_RESN_CODE		1	2	3	4	5	6	7	9	1	2	3	4	5	6	1	2	3	4	5	6	7	9
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
2	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

```
cust_claim_df.reset_index(level=["CUST_ID"], col_level=0).head()
```

	CUST_ID	value																						
ACCI_DVSN		1									2							3						
DMND_RESN_CODE		1	2	3	4	5	6	7	9	1	2	3	4	5	6	1	2	3	4	5	6	7	9
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
2	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

```
cust_claim_df = cust_claim_df.reset_index(level=["CUST_ID"], col_level=1)
cust_claim_df.head()
```

		value																						
ACCI_DVSN	CUST_ID	1									2							3						
DMND_RESN_CODE		1	2	3	4	5	6	7	9	1	2	3	4	5	6	1	2	3	4	5	6	7	9
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
2	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

```
cust_claim_df.columns = cust_claim_df.columns.droplevel(level=0)
cust_claim_df.head()
```

ACCI_DVSN	CUST_ID	1									2							3						
DMND_RESN_CODE		1	2	3	4	5	6	7	9	1	2	3	4	5	6	1	2	3	4	5	6	7	9
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
2	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

다중 인덱스의 열 이름들을 출력합니다.

```
cust_claim_df.columns
```

```
MultiIndex(levels=[[1, 2, 3, 'CUST_ID'], [1, 2, 3, 4, 5, 6, 7, 9, '']],
           labels=[[3, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2,
2, 2, 2, 2], [8, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5,
6, 7]],
           names=['ACCI_DVSN', 'DMND_RESN_CODE'])
```

다중 인덱스의 열 이름들이 삭제 레벨에 따라 어떻게 만들어지는지 확인해 봅니다.

```
cust_claim_df.columns.droplevel(level=1)
```

```
Index(['CUST_ID', 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3,
3, 3, 3],
      dtype='object', name='ACCI_DVSN')
```

```
cust_claim_df.columns.droplevel(level=0)
```

```
Index(['', 1, 2, 3, 4, 5, 6, 7, 9, 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6, 7, 9],
dtype='object', name='DMND_RESN_CODE')
```

다중 인덱스의 열 이름을 조합해서 열의 이름으로 지정합니다.

```
cust_claim_df.columns = ['_'.join([str(col) for col in cols]) for cols in
cust_claim_df.columns]
cust_claim_df.columns
```

```
Index(['CUST_ID_', '1_1', '1_2', '1_3', '1_4', '1_5', '1_6', '1_7', '1_9',
       '2_1', '2_2', '2_3', '2_4', '2_5', '2_6', '3_1', '3_2', '3_3',
'3_4',
       '3_5', '3_6', '3_7', '3_9'],
      dtype='object')
```

```
cust_claim_df.head()
```

	CUST_ID_	1_1	1_2	1_3	1_4	1_5	1_6	1_7	1_9	2_1	2_2	2_3	2_4	2_5	2_6	3_1	3_2	3_3	3_4	3_5	3_6	3_7	3_9
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
2	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

```
cust_claim_df.rename(columns={"CUST_ID_": "CUST_ID"}, inplace=True)
```

	CUST_ID	1_1	1_2	1_3	1_4	1_5	1_6	1_7	1_9	2_1	2_2	2_3	2_4	2_5	2_6	3_1	3_2	3_3	3_4	3_5	3_6	3_7	3_9
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
2	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

고객정보 데이터프레임과 사고원인/청구코드의 조합 데이터프레임을 병합합니다.

```
result = pd.merge(cust_df, cust_claim_df, how="inner")
result.head()
```

	CUST_ID	DIVIDED_SET	SIU_CUST_YN	SEX	AGE	RESI_COST	RESI_TYPE_CODE	FP_CAREER	CUST_RGST	OCCP_GRP_1	OCCP_GRP_2	TOTALPREM	MINCRDT	MAXCRDT	WEDD_YN	MATE_OCCP_GRP_1	MATE_OCCP_GRP_2	CHLD_CNT	LTBN_CHLD_AGE	MAX_PAYM
0	1.0	1.0	N	2.0	4.0	21111.0	20.0	N	199910.0	3	사무직	146980441.0	NaN	NaN	Y	3.사무직	2차산업 종사자	2.0	13.0	2003
1	2.0	1.0	N	1.0	5.0	40000.0	20.0	N	199910.0	3	사무직	94600109.0	1.0	6.0	Y	1.주부	주부	2.0	17.0	2000
2	3.0	1.0	N	1.0	6.0	0.0	NaN	N	199910.0	5	2차산업 종사자	18501269.0	NaN	NaN	N	NaN	NaN	0.0	0.0	2003
3	4.0	1.0	N	2.0	6.0	12861.0	40.0	Y	199910.0	2	3차산업 종사자	317223657.0	2.0	99.0	N	NaN	NaN	0.0	0.0	2010
4	5.0	1.0	N	2.0	5.0	0.0	NaN	Y	199910.0	2	3차산업 종사자	10506072.0	8.0	8.0	Y	3.사무직	고위 공무원	3.0	19.0	2008

```
result.shape
```

```
(21779, 64)
```

```
result.describe()
```

	CUST_ID	DIVIDED_SET	SEX	AGE	RESI_COST	RESI_TYPE_CODE	CUST_RGST	TOTALPREM	MINCRDT	MAXCRDT	CHLD_CNT	LTBN_CHLD_AGE	MAX_PAYM_YM	MAX_PRM	CUST_INCM	RCBASE_HSHD_INCM
count	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	20717.000000	21779.000000	1.626700e+04	12792.000000	12792.000000	21762.000000	21762.000000	15583.000000	1.558300e+04	16999.000000	21779.000000
mean	11194.877910	1.080215	1.569723	3.987832	16667.469627	25.726553	199910.253960	3.025541e+07	5.838024	19.581223	0.713905	8.656833	200887.967914	4.836190e+05	2694.432849	4822.017540
std	6460.805265	0.271632	0.495126	1.567412	14954.162259	29.529956	17052.756201	4.827764e+07	3.529392	32.464282	0.955101	11.601555	522.195038	1.336032e+06	2191.392814	3054.927893
min	1.000000	1.000000	1.000000	0.000000	0.000000	11.000000	101.000000	5.000000e-02	0.000000	0.000000	0.000000	0.000000	199801.000000	0.000000e+00	0.000000	0.000000
25%	5598.500000	1.000000	1.000000	3.000000	6865.000000	20.000000	200306.000000	6.782680e+06	6.000000	6.000000	0.000000	0.000000	200405.000000	1.138185e+05	0.000000	3081.000000
50%	11178.000000	1.000000	2.000000	4.000000	12261.000000	20.000000	200306.000000	1.641480e+07	6.000000	6.000000	0.000000	0.000000	200902.000000	2.282620e+05	3425.000000	4848.000000
75%	16791.500000	1.000000	2.000000	5.000000	21111.000000	20.000000	200401.000000	3.514479e+07	6.000000	7.000000	2.000000	19.000000	201403.000000	4.886120e+05	4393.000000	6638.500000
max	22400.000000	2.000000	2.000000	8.000000	305555.000000	99.000000	201602.000000	1.840052e+09	99.000000	99.000000	6.000000	35.000000	207110.000000	5.686262e+07	12687.000000	19829.000000

JPIBASE_HSHD_INCM	강원	경기	경남	경북	광주	대구	대전	부산	서울	세종	울산	인천	전남	전북	제주	충남	충북
21556.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000	21186.000000
5208.083967	0.026291	0.239167	0.073209	0.045502	0.039130	0.045455	0.023317	0.069433	0.166838	0.001841	0.032380	0.067545	0.051213	0.044275	0.009538	0.035401	0.028368
2713.491611	0.160003	0.426585	0.260485	0.208407	0.193908	0.208304	0.150913	0.254194	0.374618	0.042866	0.177011	0.250969	0.220437	0.205709	0.092540	0.184795	0.166025
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3561.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4689.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
6854.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25072.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

HOSP_DAY	1_1	1_2	1_3	1_4	1_5	1_6	1_7	1_9	2_1	2_2	2_3	2_4	2_5	2_6	3_1	3_2	3_3
21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000	21779.000000
6.717302	0.000045	0.502599	0.352541	0.021626	0.170026	0.370935	0.002296	0.000184	0.000046	0.350200	0.028238	0.010790	0.021305	0.030809	0.000459	1.796226	0.716883
7.738851	0.006776	1.529176	1.488209	0.219452	0.530864	0.833424	0.051555	0.027104	0.006776	1.092847	0.304528	0.179854	0.214939	0.199449	0.023469	4.264381	2.057895
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4.333333	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
9.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.000000	0.000000
289.000000	1.000000	32.000000	65.000000	7.000000	9.000000	16.000000	2.000000	4.000000	1.000000	28.000000	13.000000	11.000000	9.000000	4.000000	2.000000	90.000000	45.000000

3_4	3_5	3_6	3_7	3_9
21779.000000	21779.000000	21779.000000	21779.000000	21779.000000
0.007209	0.731163	0.062767	0.004913	0.000046
0.110074	1.325328	0.286949	0.113079	0.006776
0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	1.000000	0.000000	0.000000	0.000000
4.000000	28.000000	5.000000	7.000000	1.000000

5절. 마무리

5.1. CUST_ID 열 형 변환

CUST_ID 열은 고객의 아이디를 저장한 열입니다. 전처리 과정에서 float 유형으로 바뀐 것을 다시 int 유형으로 바꿔줍니다.

```
result = result.astype({"CUST_ID": int})
result.head()
```

	CUST_ID	DIVIDED_SET	SIU_CUST_YN	SEX	AGE	RESI_COST	RESI_TYPE_CODE	FP_CAREER	CUST_RGST	OCCP_GRP_1	OCCP_GRP_2	TOTALPREM	MINCRDT	MAXCRDT	WEDD_YN	MATE_OCCP_GRP_1	MATE_OCCP_GRP_2	CHLD_CNT	LTBN_CHLD_AGE
0	1	1.0	N	2.0	4.0	21111.0	20.0	N	199910.0	3	사무직	146980441.0	NaN	NaN	Y	3.사무직	2차산업종사자	2.0	13.0
1	2	1.0	N	1.0	5.0	40000.0	20.0	N	199910.0	3	사무직	94600109.0	1.0	6.0	Y	1.주부	주부	2.0	17.0
2	3	1.0	N	1.0	6.0	0.0	NaN	N	199910.0	5	2차산업종사자	18501269.0	NaN	NaN	N	NaN	NaN	0.0	0.0
3	4	1.0	N	2.0	6.0	12861.0	40.0	Y	199910.0	2	3차산업종사자	317223657.0	2.0	99.0	N	NaN	NaN	0.0	0.0
4	5	1.0	N	2.0	5.0	0.0	NaN	Y	199910.0	2	3차산업종사자	10506072.0	8.0	8.0	Y	3.사무직	고위공무원	3.0	19.0

열 이름들을 확인해 봅니다.

```
result.columns
```

```
Index(['CUST_ID', 'DIVIDED_SET', 'SIU_CUST_YN', 'SEX', 'AGE', 'RESI_COST',
       'RESI_TYPE_CODE', 'FP_CAREER', 'CUST_RGST', 'OCCP_GRP_1', 'OCCP_GRP_2',
       'TOTALPREM', 'MINCRDT', 'MAXCRDT', 'WEDD_YN', 'MATE_OCCP_GRP_1',
       'MATE_OCCP_GRP_2', 'CHLD_CNT', 'LTBN_CHLD_AGE', 'MAX_PAYM_YM', 'MAX_PRM',
       'CUST_INCM', 'RCBASE_HSHD_INCM', 'JPBASE_HSHD_INCM', '강원', '경기', '경남',
       '경북', '광주', '대구', '대전', '부산', '서울', '세종', '울산', '인천',
       '전남', '전북', '제주', '충남', '충북', 'HOSP_DAY', '1_1', '1_2', '1_3',
       '1_4', '1_5', '1_6', '1_7', '1_9', '2_1', '2_2', '2_3', '2_4', '2_5', '2_6',
       '3_1', '3_2', '3_3', '3_4', '3_5', '3_6', '3_7', '3_9'],
      dtype='object')
```

5.2. 열 삭제

'FP_CAREER', 'DIVIDED_SET', 'RESI_TYPE_CODE', 'OCCP_GRP_2', 'MATE_OCCP_GRP_1', 'MATE_OCCP_GRP_2' 열을 삭제해야 합니다.

```
result = result.drop(['FP_CAREER', 'DIVIDED_SET', 'RESI_TYPE_CODE',
'OCCP_GRP_2', 'MATE_OCCP_GRP_1', 'MATE_OCCP_GRP_2'], axis=1)
```

5.3. 결과 파일 저장

최종 결과 파일을 저장합니다. 저장할 때에 인덱스는 파일에 저장하지 않도록 설정합니다.

```
result.to_csv("홍길동_result.csv", index=False, encoding="utf-8")
result.head()
```

	CUST_ID	SIU_CUST_YN	SEX	AGE	RESI_COST	CUST_RGST	OCCP_GRP_1	TOTALPREM	MINCRDT	MAXCRDT	WEDD_YN	CHLD_CNT	LTBN_CHLD_AGE	MAX_PAYM_YM	MAX_PRM	CUST_INCM	RCBASE_HSHD_INCM	JPBASE_HSHD_INCM	강원	경기	경남	경북	광주	대구
0	1	N	2.0	4.0	21111.0	199910.0	3	1469880441.0	NaN	NaN	Y	2.0	13.0	200811.0	319716.0	4879.0	10094.0	11337.0	0.0	0.0	0.0	0.0	0.0	0.0
1	2	N	1.0	5.0	40000.0	199910.0	3	946001109.0	1.0	6.0	Y	2.0	17.0	200012.0	341341.0	6509.0	9143.0	6509.0	0.0	0.0	0.0	0.0	0.0	0.0
2	3	N	1.0	6.0	0.0	199910.0	5	185012269.0	NaN	NaN	N	0.0	0.0	200305.0	131300.0	4180.0	0.0	4180.0	0.0	0.0	0.0	0.0	0.0	0.0
3	4	N	2.0	6.0	12861.0	199910.0	2	317223657.0	2.0	99.0	N	0.0	0.0	201009.0	1493184.0	NaN	4270.0	5914.0	0.0	1.0	0.0	0.0	0.0	0.0
4	5	N	2.0	5.0	0.0	199910.0	2	10509072.0	8.0	8.0	Y	3.0	19.0	200807.0	166760.0	3894.0	0.0	8885.0	0.0	0.0	0.0	0.0	1.0	0.0

6절. 요약정보 정답

요약 정보 정답 데이터입니다.

번호	요약 정보	값
1	나이를 연령대로 변화 후 평균	4.014509
2	나이를 연령대로 변환 후 표준편차	1.577022
3	지역 결측치 제거 후 행의 수	21779
4	단순 레이블 인코딩 후 평균	6.264521
5	단순 레이블 인코딩 후 표준편차	4.676853
6	원 핫 인코딩 후 서울 지역의 평균	0.168373
7	원 핫 인코딩 후 서울 지역의 표준편차	0.374206
8	가장 많은 직업의 코드	8
9	가장 많은 직업의 수	4868
10	HOSP_DAY 열의 평균	6.717302
11	HOSP_DAY 열의 표준편차	7.738851
12	HOSP_DAY 열의 MAX	289
13	교통재해(2)와 입원(2)의 횟수에 대한 평균	0.350200
14	교통재해(2)와 입원(2)의 횟수에 대한 표준편차	1.062947