유통고객 구매패턴 데이터 전처리

단계1. 데이터 탐색 및 준비

0.미션

(1) 미션① : 비즈니스 정보 파악

- 예제 질문 9개 조회를 수행해냅니다.

(2) 미션 ② : 데이터 구조 만들기

- 고객 이탈을 정의하고, 데이터를 생성합니다.
  - 대상 고객
    - 2014 ~ 2016년 신규 가입 고객 이면서,
    - 2016년 하반기에 한번 이상 방문한 고객을 대상 고객으로 정의합니다.
  - Labeling
    - 위 대상 고객 중, 2017년 1~3월(3개월)동안 방문(구매)하지 않은 사람은 이탈로 간주합니다.
  - feature 추가하기
    - 주어진 기본 feature 3가지를 생성합니다.

1.환경설정

(1) 라이브러리 설치하기

아래를 실행해주세요.

In [334…

```
!pip install matplotlib
!pip install --upgrade matplotlib

import matplotlib.pyplot as plt
```

```
Requirement already satisfied: matplotlib in c:\users\user\anaconda3\lib\site-packages (3.8.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\user\anaconda3\lib\site-packages
(from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\user\anaconda3\lib\site-packages (from
matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\user\anaconda3\lib\site-packages
(from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\user\anaconda3\lib\site-packages
(from matplotlib) (1.4.4)
Requirement already satisfied: numpy<2,>=1.21 in c:\users\user\anaconda3\lib\site-packages (fr
om matplotlib) (1.24.3)
Requirement already satisfied: packaging>=20.0 in c:\users\user\anaconda3\lib\site-packages (f
rom matplotlib) (23.1)
Requirement already satisfied: pillow>=8 in c:\users\user\anaconda3\lib\site-packages (from ma
tplotlib) (10.0.1)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\user\anaconda3\lib\site-packages
(from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\user\anaconda3\lib\site-packag
es (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\lib\site-packages (from pyt
hon-dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: matplotlib in c:\users\user\anaconda3\lib\site-packages (3.8.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\user\anaconda3\lib\site-packages
(from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\user\anaconda3\lib\site-packages (from
matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\user\anaconda3\lib\site-packages
(from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\user\anaconda3\lib\site-packages
(from matplotlib) (1.4.4)
Requirement already satisfied: numpy<2,>=1.21 in c:\users\user\anaconda3\lib\site-packages (fr
om matplotlib) (1.24.3)
Requirement already satisfied: packaging>=20.0 in c:\users\user\anaconda3\lib\site-packages (f
rom matplotlib) (23.1)
Requirement already satisfied: pillow>=8 in c:\users\user\anaconda3\lib\site-packages (from ma
tplotlib) (10.0.1)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\user\anaconda3\lib\site-packages
(from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\user\anaconda3\lib\site-packag
es (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\lib\site-packages (from pyt
hon-dateutil>=2.7->matplotlib) (1.16.0)
```

(2) 라이브러리 불러오기

- 세부 요구사항
  - 기본적으로 필요한 라이브러리를 import 하도록 코드가 작성되어 있습니다.
  - 필요하다고 판단되는 라이브러리를 추가하세요.

In [335…

```
#[문제1] pandas, numpy, matplotlib 라이브러리를 임포트하세요.
```

In [336…

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

(3) 데이터 불러오기

- 주어진 데이터셋
  - customers.csv : 고객정보
  - sales.csv : 판매정보
  - products.csv : 상품정보
- 다음과 같이 데이터를 저장하세요.
  - 주피터랩 실행
    - 제공된 압축파일 '미프 1차_구매'를 다운받아 압축을 푼다.
    - anaconda의 root directory(보통 C:\Users\ 에 '미프 1차_구매' 폴더를 만들고, 복사하고 붙여넣습니다.
    - '미프 1차_구매' 폴더에 필요한 파일들을 넣고, 본 파일 '데이터 전처리_교육생용' 실습파일을 불러옵니다
- 세부 요구사항
  - 데이터셋을 각각 불러와서 데이터프레임으로 저장합니다.
  - 날짜 데이터들은 날짜형식으로 변환합니다. (customers['RegisterDate'], sales['OrderDate'])
  - 기본 정보를 확인합니다.(.shape, .info(), .head())

In [337…
```
#[문제2] '미프 1차_구매' 폴더에서 본 파일들을 열어주세요.
```

In [338…
```
# 전체 데이터 불러오기

# 데이터 3개를 pd.read_csv로 불러오기
customers = pd.read_csv('customers.csv')
sales = pd.read_csv('sales.csv')
products = pd.read_csv('products.csv')
```

In [339…
```
#[문제3] 'customers', 'sales' 데이터의 상단 일부 행을 출력해보세요.
```

In [340…
```
#customers
customers.head()
```

Out[340]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2
0	c328222	2014-09-25	F	1960	강원도	원주시
1	c281448	2013-06-18	F	1974	강원도	원주시
2	c038336	2003-10-10	F	1968	강원도	춘천시
3	c084237	2007-03-09	F	1982	강원도	강릉시
4	c162600	2010-06-14	F	1978	강원도	속초시

In [341…
```
#sales
sales.head()
```

Out[341]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID
0	107	2	2016-01-02	p1036481	2	2100	c150417
1	69	1	2016-01-02	p1152861	1	1091	c212716
2	69	7	2016-01-02	p1013161	1	2600	c212716
3	69	8	2016-01-02	p1005771	1	1650	c212716
4	69	11	2016-01-02	p1089531	1	2600	c212716

In [342…
```
#[문제4] 'customers'와 'sales' 데이터프레임의 구조와 기본 통계정보를 출력해보세요.
```

In [343…
```
#customers
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2243 entries, 0 to 2242
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   CustomerID    2243 non-null   object
 1   RegisterDate  2243 non-null   object
 2   Gender        2243 non-null   object
 3   BirthYear     2243 non-null   int64 
 4   Addr1         2243 non-null   object
 5   Addr2         2243 non-null   object
dtypes: int64(1), object(5)
memory usage: 105.3+ KB
```

In [344…
```
#sales
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70012 entries, 0 to 70011
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   OrderID     70012 non-null  int64 
 1   Seq         70012 non-null  int64 
 2   OrderDate   70012 non-null  object
 3   ProductID   70012 non-null  object
 4   Qty         70012 non-null  int64 
 5   Amt         70012 non-null  int64 
 6   CustomerID  70012 non-null  object
dtypes: int64(4), object(3)
memory usage: 3.7+ MB
```

In [345…
```
#[문제5] 날짜 데이터를 날짜형식으로 변환해주세요. 날짜데이터 : customers['RegisterDate'], sale
```

In [346…
```
# pd.to_datetime 활용
customer_date = pd.to_datetime(customers['RegisterDate'])
sales_date = pd.to_datetime(sales['OrderDate'])
```

In [347…
```
#[문제6] 'customers'의 데이터타입이 날짜형식으로 잘 변환되었는지 확인해주세요.
print(customer_date.dtype)
```

```
datetime64[ns]
```

In [348…
```
#[문제7] 'sales'의 데이터타입이 날짜형식으로 변환되었는지 확인해주세요.
```

In [349…
```
print(sales_date.dtype)
```

```
datetime64[ns]
```

In [350…
```
#[문제8] 'sales' 데이터의 'OrderDate'의 최소값과 최대값을 표시해주세요.
```

In [351…
```
# min, max 함수 활용
print(sales['OrderDate'].min())
print(sales['OrderDate'].max())
```

```
2016-01-02
2017-03-31
```

2.데이터 탐색

주어진 데이터에 대해서 다음의 요건에 맞게 조회하면서 데이터를 파악해 봅시다.

- **상세요구사항**
  - 아래 예제 질문 9개에 대해 조회를 수행합니다.
  - 예제 질문
    - Q01) 일별 매출액
    - Q02) 월별 매출액
    - Q03) 요일별 매출액 평균
    - Q04) 일별 고객 1인당 평균 구매액(객단가)
    - Q05) 일별 방문 고객수(구매 고객수)
    - Q06) 매출 상위 top 10 상품
    - Q07) 요일별 매출상위 Top 10 상품
    - Q08) 카테고리별 매출 비중
    - Q09) 고객 나이대

Q01) 일별 매출액

In [352…
```
#[문제9] 'sales' 데이터프레임에서 일별 매출액의 합계를 'daily_sales' 데이터 프레임에 할당하세요
daily_sales = sales.groupby(by='OrderDate')[['Amt']].sum() # 일별 합계
daily_sales.head()
```

Out[352]:

OrderDate	Amt
2016-01-02	503234
2016-01-03	211202
2016-01-04	705195
2016-01-05	502803
2016-01-06	485984

In [353…
```
# 'sales'에서 주문일자별 주문 금액 합산 : groupby
sales.groupby(by='OrderDate')[['Amt']].sum()
```

Out[353]:

OrderDate	Amt
2016-01-02	503234
2016-01-03	211202
2016-01-04	705195
2016-01-05	502803
2016-01-06	485984
…	…
2017-03-27	465265
2017-03-28	529778
2017-03-29	451900
2017-03-30	425932
2017-03-31	471847

447 rows × 1 columns

In [354…
```
#[문제10] 'sales' 데이터프레임에서 부분인 2016년 1월~3월까지 3개월의 범위로 데이터를 선택하고
```

In [355…
```
# loc 함수 활용
daily_sales = sales.loc[(sales_date.dt.year == 2016) & (sales_date.dt.month <= 3)]
daily_sales
```

Out[355]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID
0	107	2	2016-01-02	p1036481	2	2100	c150417
1	69	1	2016-01-02	p1152861	1	1091	c212716
2	69	7	2016-01-02	p1013161	1	2600	c212716
3	69	8	2016-01-02	p1005771	1	1650	c212716
4	69	11	2016-01-02	p1089531	1	2600	c212716
…	…	…	…	…	…	…	…
14069	63	2	2016-03-31	p1175481	1	1300	c304973
14070	67	1	2016-03-31	p1178011	1	8800	c115575
14071	67	2	2016-03-31	p1162631	1	4600	c115575
14072	67	3	2016-03-31	p1002841	1	11000	c115575
14073	71	1	2016-03-31	p1178011	1	8800	c222420

14074 rows × 7 columns

In [356…
```
#[문제11] 일자별 매출액을 'daily_sales' 데이터 프레임에 저장해주세요.
```

In [357…
```
#'OrderDate' 기준 'Amt'의 합계 계산
sales.groupby(by='OrderDate', as_index=False)[['Amt']].sum()
```

Out[357]:

	OrderDate	Amt
0	2016-01-02	503234
1	2016-01-03	211202
2	2016-01-04	705195
3	2016-01-05	502803
4	2016-01-06	485984
…	…	…
442	2017-03-27	465265
443	2017-03-28	529778
444	2017-03-29	451900
445	2017-03-30	425932
446	2017-03-31	471847

447 rows × 2 columns

In [358…
```
#[문제12] 'daily_sales' 데이터를 확인해보세요.
```

In [359…
```
print(daily_sales.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 14074 entries, 0 to 14073
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   OrderID     14074 non-null  int64 
 1   Seq         14074 non-null  int64 
 2   OrderDate   14074 non-null  object
 3   ProductID   14074 non-null  object
 4   Qty         14074 non-null  int64 
 5   Amt         14074 non-null  int64 
 6   CustomerID  14074 non-null  object
dtypes: int64(4), object(3)
memory usage: 879.6+ KB
None
```

Q02) 월별 매출액

In [360…
```
#[문제13] 기간별 조회를 위해 'sales' 데이터프레임에 주문일자의 연도와 월의 변수를 추가해주세요
```

In [361…
```
# 연도는 'Year', 월은 'Month'로 변수를 추가
# 'OrderDate' 컬럼에서 dt.year, dt.month 함수 활용
Year = sales_date.dt.year
Month = sales_date.dt.month
```

```
sales['Year'] = Year
sales['Month'] = Month
sales.head()
```

Out[361]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1
1	69	1	2016-01-02	p1152861	1	1091	c212716	2016	1
2	69	7	2016-01-02	p1013161	1	2600	c212716	2016	1
3	69	8	2016-01-02	p1005771	1	1650	c212716	2016	1
4	69	11	2016-01-02	p1089531	1	2600	c212716	2016	1

In [362…

```
#[문제14] 월별 매출액의 합계를 구하고 'monthly_sales'에 저장해주세요.
```

In [363…

```
# groupby 함수 활용, as_index = False
monthly_sales = sales.groupby(by=['Year', 'Month'], as_index=False)[['Amt']].sum()
monthly_sales
```

Out[363]:

	Year	Month	Amt
0	2016	1	14209990
1	2016	2	14367451
2	2016	3	17111120
3	2016	4	15129917
4	2016	5	13976941
5	2016	6	13113610
6	2016	7	10717743
7	2016	8	10176702
8	2016	9	10496653
9	2016	10	10003648
10	2016	11	9767765
11	2016	12	13061191
12	2017	1	14721755
13	2017	2	12559903
14	2017	3	15585079

In [364…

```
#[문제15] 'monthly_sales'를 출력하고 확인해주세요.
```

In [365…

```
monthly_sales.head()
```

Out[365]:

	Year	Month	Amt
0	2016	1	14209990
1	2016	2	14367451
2	2016	3	17111120
3	2016	4	15129917
4	2016	5	13976941

Q03) 요일별 매출액 평균

In [366…
```
#[문제16] 'sales' 데이터프레임에서 주문일자 기준 일별 매출액의 합계를 구해 'day_sales'에 저장하
```

In [367…
```
# groupby 함수 활용, 'OrderDate' 기준 'Amt'의 합산
# sum 함수 활용
day_sales = sales.groupby(by='OrderDate', as_index=False)[['Amt']].sum()
```

In [368…
```
#[문제17] 'day_sales'에서 'OrderDate' 열의 날짜 정보를 활용하여 'Weekday' 열을 추가하세요.
```

In [369…
```
# dt.day_name() 함수
day_sales['weekday'] = sales_date.dt.day_name()
```

In [370…
```
#[문제18] day_sales 데이터프레임의 'Weekday' 열을 출력하세요.
```

In [371…
```
day_sales['weekday']
```

Out[371]:
```
0      Saturday
1      Saturday
2      Saturday
3      Saturday
4      Saturday
         ...
442      Monday
443      Monday
444      Monday
445      Monday
446      Monday
Name: weekday, Length: 447, dtype: object
```

Q04) 일별 고객 1인당 평균 구매액(객단가)

In [372…
```
sales.head()
```

Out[372]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1
1	69	1	2016-01-02	p1152861	1	1091	c212716	2016	1
2	69	7	2016-01-02	p1013161	1	2600	c212716	2016	1
3	69	8	2016-01-02	p1005771	1	1650	c212716	2016	1
4	69	11	2016-01-02	p1089531	1	2600	c212716	2016	1

In [373…

```
#[문제19] 'sales' 데이터프레임에서 주문일자,고객ID별 주문금액을 합산하여 'order_amt'에 저장하시
order_amt = sales.groupby(by=['OrderDate', 'CustomerID'], as_index=False)['Amt'].sum()
order_amt.head()
#sales['order_amt'] = sales_2['Amt']
#sales.head()
```

Out[373]:

	OrderDate	CustomerID	Amt
0	2016-01-02	c029643	2650
1	2016-01-02	c047907	1650
2	2016-01-02	c068640	3300
3	2016-01-02	c082866	12350
4	2016-01-02	c083862	9504

In [374…

```
# 'OrderDate'와 'CustomerID' 기준 groupby 함수 활용
# sum 함수 활용
sales.groupby(by=['CustomerID', 'OrderDate'], as_index=False)[['Amt']].sum()
```

Out[374]:

	CustomerID	OrderDate	Amt
0	c017487	2016-01-14	3900
1	c017487	2016-01-20	3682
2	c017487	2016-01-25	11363
3	c017487	2016-01-28	1500
4	c017487	2016-02-06	6950
…	…	…	…
32649	c401555	2017-03-22	6900
32650	c401555	2017-03-29	5040
32651	c401584	2017-03-28	6509
32652	c402010	2017-03-25	20650
32653	c402634	2017-03-30	4850

32654 rows × 3 columns

In [375…

```
#[문제20] 'order_amt'의 상단 10행을 출력하세요.
```

In [376…
```
# head 함수
order_amt.head(10)
```

Out[376]:

	OrderDate	CustomerID	Amt
0	2016-01-02	c029643	2650
1	2016-01-02	c047907	1650
2	2016-01-02	c068640	3300
3	2016-01-02	c082866	12350
4	2016-01-02	c083862	9504
5	2016-01-02	c085558	12710
6	2016-01-02	c090006	2841
7	2016-01-02	c090821	12500
8	2016-01-02	c104962	1650
9	2016-01-02	c105415	1300

In [377…
```
#[문제21] 'order_amt'에서 주문일자별 고객 구매액의 총 평균을 구하고 'order_amt2'에 저장하세요.
```

In [378…
```
# 'OrderDate'기준 groupby 함수 활용
# mean 함수 활용
order_amt2 = order_amt.groupby(by='OrderDate', as_index=False)[['Amt']].mean()
order_amt2.head()
```

Out[378]:

	OrderDate	Amt
0	2016-01-02	7624.757576
1	2016-01-03	5415.435897
2	2016-01-04	6716.142857
3	2016-01-05	6364.594937
4	2016-01-06	6844.845070

Q05) 일별 방문 고객수(구매 고객수)

In [379…
```
#[문제22] 'order_amt'에서 일별 방문 고객수를 구하고 'daily_visit'으로 저장해주세요.
```

In [380…
```
# CustomerID 컬럼의 데이터 개수 계산
daily_visit = order_amt.groupby(by='CustomerID', as_index=False)[['CustomerID']].count()
daily_visit.head()
```

Out[380]:

	CustomerID
0	49
1	25
2	59
3	40
4	10

In [381…
```
#[문제23] 'order_amt2'에 'daily_visit' 컬럼을 주문일자 기준 추가해주세요.
```

In [382…
```
order_amt2['daily_visit'] = daily_visit
order_amt2
```

Out[382]:

	OrderDate	Amt	daily_visit
0	2016-01-02	7624.757576	49
1	2016-01-03	5415.435897	25
2	2016-01-04	6716.142857	59
3	2016-01-05	6364.594937	40
4	2016-01-06	6844.845070	10
…	…	…	…
442	2017-03-27	5410.058140	4
443	2017-03-28	6160.209302	27
444	2017-03-29	5868.831169	2
445	2017-03-30	5679.093333	18
446	2017-03-31	5972.746835	3

447 rows × 3 columns

In [383…
```
#[문제24] 'order_amt2' 데이터를 확인해보세요.
```

In [384…
```
order_amt2.head()
```

Out[384]:

	OrderDate	Amt	daily_visit
0	2016-01-02	7624.757576	49
1	2016-01-03	5415.435897	25
2	2016-01-04	6716.142857	59
3	2016-01-05	6364.594937	40
4	2016-01-06	6844.845070	10

Q06) 매출 상위 top 10 상품

In [385…
```
sales.head()
```

Out[385]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1
1	69	1	2016-01-02	p1152861	1	1091	c212716	2016	1
2	69	7	2016-01-02	p1013161	1	2600	c212716	2016	1
3	69	8	2016-01-02	p1005771	1	1650	c212716	2016	1
4	69	11	2016-01-02	p1089531	1	2600	c212716	2016	1

In [386…
```
products.head()
```

Out[386]:

	ProductID	ProductName	Category	SubCategory
0	p1052661	새우깡	간식	과자
1	p1054261	고구마스틱	간식	과자
2	p1097821	짱구	간식	과자
3	p1097831	감자칩	간식	과자
4	p1119071	뿌셔뿌셔	간식	과자

In [387…
```
#[문제25] 'sales' 데이터와 'products' 데이터를 합쳐서 'top_amt'에 할당하세요.
```

In [388…
```
# merge 함수 활용
top_amt = pd.merge(sales, products, on='ProductID', how='outer')
```

In [389…
```
top_amt.head()
```

Out[389]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month	ProductName	Categ
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1	순두부	반
1	137	4	2016-01-02	p1036481	2	2100	c280590	2016	1	순두부	반
2	63	16	2016-01-03	p1036481	1	1050	c037915	2016	1	순두부	반
3	135	3	2016-01-04	p1036481	3	3150	c100815	2016	1	순두부	반
4	63	13	2016-01-06	p1036481	10	10500	c048405	2016	1	순두부	반

In [390…
```
#[문제26] 상품명을 기준으로 매출 합계를 구하여 'top_amt2'에 저장하세요
```

In [391…
```
# groupby 함수 활용, sum함수 활용
top_amt2 = top_amt.groupby(by='ProductName', as_index=False)[['Amt']].sum()
```

In [392…
```
top_amt2
```

Out[392]:

	ProductName	Amt
0	감귤컵푸딩	677154
1	감자칩	3676518
2	고구마스틱	1620986
3	깻잎	2930359
4	낱개캔디	479197
5	느타리버섯	4163035
6	당근	4453387
7	두부_대	10119037
8	두부_소	7319963
9	딸기_대	11011541
10	딸기_소	7517664
11	딸기아이스크림	940786
12	딸기요거트	955633
13	마늘	3354827
14	무	1260905
15	무농약시금치	2469392
16	배아이스크림	623612
17	백오이	2036463
18	부추	1070348
19	부침용두부	1241545
20	브로컬리	3491762
21	뿌셔뿌셔	1215673
22	사과_대	5399285
23	사과_소	14078818
24	사과아이스크림	794018
25	상추	1494844
26	새우깡	2340237
27	소보루빵	536294
28	숙주나물	2601703
29	순두부	820101
30	시금치	2665640
31	쌈모음	3229161
32	애호박	3421170
33	양배출	1478820

	ProductName	Amt
34	양파	762702
35	연두부	1017937
36	열무	1867542
37	우유1000	18129067
38	우유200	2359250
39	유기농우유	6437323
40	유부	2113187
41	저지방우유	4404124
42	짱구	2161256
43	참외	5491030
44	청오이	3960918
45	초코아이스크림	1158483
46	초코우유	2456346
47	초코콘	374057
48	콘칩	942184
49	콩나물	6971347
50	토마토	6291001
51	통단팥빵	830162
52	파	4518592
53	팥아이스크림	759973
54	팽이버섯	691947
55	포도컵푸딩	777637
56	포토아이스크림	672267
57	플레인요거트	5197017
58	피망	3164241

In [393…
```
#[문제27] 'top_amt2' 데이터프레임을 'Amt'열 기준 상위 10개의 데이터를 선택하여 내림차순으로 정
```

In [394…
```
# sort_values 함수 활용
top10_amt = top_amt2.sort_values(by='Amt', ascending=False)
top10_amt = top_amt2.head(10)
```

In [395…
```
#[문제28] 'top10_amt' 데이터프레임을 확인하세요.
```

In [396…
```
top10_amt
```

Out[396]:

	ProductName	Amt
0	감귤컵푸딩	677154
1	감자칩	3676518
2	고구마스틱	1620986
3	깻잎	2930359
4	낱개캔디	479197
5	느타리버섯	4163035
6	당근	4453387
7	두부_대	10119037
8	두부_소	7319963
9	딸기_대	11011541

Q07) 카테고리별 매출 비중

In [397…
```
#[문제29] 'sales'와 'products' 데이터프레임을 합치고 'cate_amt'에 저장하세요.
```

In [398…
```
sales.head()
```

Out[398]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1
1	69	1	2016-01-02	p1152861	1	1091	c212716	2016	1
2	69	7	2016-01-02	p1013161	1	2600	c212716	2016	1
3	69	8	2016-01-02	p1005771	1	1650	c212716	2016	1
4	69	11	2016-01-02	p1089531	1	2600	c212716	2016	1

In [399…
```
products.head()
```

Out[399]:

	ProductID	ProductName	Category	SubCategory
0	p1052661	새우깡	간식	과자
1	p1054261	고구마스틱	간식	과자
2	p1097821	짱구	간식	과자
3	p1097831	감자칩	간식	과자
4	p1119071	뿌셔뿌셔	간식	과자

In [400…
```
# merge 함수
cate_amt = pd.merge(sales, products, on='ProductID', how='outer')
```

In [401…
```
cate_amt.head()
```

Out[401]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month	ProductName	Categ
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1	순두부	반
1	137	4	2016-01-02	p1036481	2	2100	c280590	2016	1	순두부	반
2	63	16	2016-01-03	p1036481	1	1050	c037915	2016	1	순두부	반
3	135	3	2016-01-04	p1036481	3	3150	c100815	2016	1	순두부	반
4	63	13	2016-01-06	p1036481	10	10500	c048405	2016	1	순두부	반

In [402…
```
#[문제30] 카테고리를 기준으로 매출 합계를 구하여 'cate_amt2'에 저장하세요.
```

In [403…
```
# groupby, sum 함수 활용
cate_amt2 = cate_amt.groupby(by='Category', as_index=False)[['Amt']].sum()
```

In [404…
```
cate_amt2
```

Out[404]:

	Category	Amt
0	간식	12920570
1	과일	49789339
2	반찬류	32204820
3	유제품	45261956
4	채소	54822783

In [405…
```
#[문제31] 카테고리별 매출 비중을 구하여 'Rate' 변수를 'cate_amt2'에 추가해주세요.
```

In [406…
```
# 매출 비중 : 'Amt' / Amt.sum()
Rate = cate_amt2['Amt'] / cate_amt2['Amt'].sum() * 100
cate_amt2['Rate'] = Rate
```

In [407…
```
# 'cate_amt2'를 출력하세요.
```

In [408…
```
cate_amt2.head()
```

Out[408]:

	Category	Amt	Rate
0	간식	12920570	6.625951
1	과일	49789339	25.533064
2	반찬류	32204820	16.515337
3	유제품	45261956	23.211323
4	채소	54822783	28.114324

Q08) 요일별 매출 비중

In [409…
```
sales.head()
```

Out[409]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1
1	69	1	2016-01-02	p1152861	1	1091	c212716	2016	1
2	69	7	2016-01-02	p1013161	1	2600	c212716	2016	1
3	69	8	2016-01-02	p1005771	1	1650	c212716	2016	1
4	69	11	2016-01-02	p1089531	1	2600	c212716	2016	1

In [410…
```
products.head()
```

Out[410]:

	ProductID	ProductName	Category	SubCategory
0	p1052661	새우깡	간식	과자
1	p1054261	고구마스틱	간식	과자
2	p1097821	짱구	간식	과자
3	p1097831	감자칩	간식	과자
4	p1119071	뿌셔뿌셔	간식	과자

In [411…
```
#[문제32] 'sales'와 'products' 데이터프레임을 합치고 'day_amt'에 저장하세요.
```

In [412…
```
day_amt = pd.concat([sales, products], axis=1, join='outer')
```

In [413…
```
day_amt.head()
```

Out[413]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month	ProductID	ProductNa
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1	p1052661	새
1	69	1	2016-01-02	p1152861	1	1091	c212716	2016	1	p1054261	고구마
2	69	7	2016-01-02	p1013161	1	2600	c212716	2016	1	p1097821	짱
3	69	8	2016-01-02	p1005771	1	1650	c212716	2016	1	p1097831	감
4	69	11	2016-01-02	p1089531	1	2600	c212716	2016	1	p1119071	뿌셔

In [414…
```
#[문제33] 'day_amt' 데이터에서 'OrderDate' 열을 날짜 형식의 데이터타입으로 변환해주세요
```

In [415…
```
# pd.to_datetime 함수 활용
day_amt_name = pd.to_datetime(day_amt['OrderDate'])
```

In [416…
```
print(day_amt_name.dtypes)
```

```
datetime64[ns]
```

In [417…
```
#[문제34] 'day_amt' 데이터에서 'OrderDate' 열의 날짜 정보를 활용하여 요일을 나타내는 'Weekday'
```

In [418…
```
# dt.day_name() 함수 활용
day_amt['weekday'] = day_amt_name.dt.day_name()
```

In [419…
```
day_amt.head()
```

Out[419]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month	ProductID	ProductNa
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1	p1052661	새
1	69	1	2016-01-02	p1152861	1	1091	c212716	2016	1	p1054261	고구마
2	69	7	2016-01-02	p1013161	1	2600	c212716	2016	1	p1097821	정
3	69	8	2016-01-02	p1005771	1	1650	c212716	2016	1	p1097831	감
4	69	11	2016-01-02	p1089531	1	2600	c212716	2016	1	p1119071	뿌셔

In [420…
```
#[문제35] day_amt를 요일 기준 매출액의 합을 계산하고 day_amt2에 저장하세요.
```

In [421…
```
# groupby, sum 함수 활용
day_amt2 = day_amt.groupby(by='weekday', as_index=False)[['Amt']].sum()
```

In [422…
```
#[문제36] day_amt2를 출력하세요.
day_amt2
```

Out[422]:

	weekday	Amt
0	Friday	30838811
1	Monday	34484281
2	Saturday	28683620
3	Sunday	18358536
4	Thursday	26608397
5	Tuesday	28097823
6	Wednesday	27928000

In [423…
```
#[문제37] 요일별 매출 비중을 구하여 'Rate' 변수를 'day_amt2'에 추가해주세요.
```

In [424…
```
Rate = day_amt2['Amt'] / day_amt2['Amt'].sum() * 100
day_amt2['Rate'] = Rate
```

In [425…
```
#[문제38] day_amt2를 다시 출력하여 변수가 추가되었는지 확인해주세요.
day_amt2.head()
```

Out[425]:

	weekday	Amt	Rate
0	Friday	30838811	15.814818
1	Monday	34484281	17.684295
2	Saturday	28683620	14.709589
3	Sunday	18358536	9.414660
4	Thursday	26608397	13.645369

Q09) 고객 나이대

In [426…
```
customers.head()
```

Out[426]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2
0	c328222	2014-09-25	F	1960	강원도	원주시
1	c281448	2013-06-18	F	1974	강원도	원주시
2	c038336	2003-10-10	F	1968	강원도	춘천시
3	c084237	2007-03-09	F	1982	강원도	강릉시
4	c162600	2010-06-14	F	1978	강원도	속초시

In [427…
```
#[문제39] 고객의 태어난 연도를 활용하여 'Age'(고객 나이) 컬럼을 추가 합니다. (현재 : 2016년)
```

In [428…
```
# 2016년 - 고객의 태어난 연도
customers['Age'] = 2016 - customers['BirthYear']
```

In [429…
```
customers.head()
```

Out[429]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age
0	c328222	2014-09-25	F	1960	강원도	원주시	56
1	c281448	2013-06-18	F	1974	강원도	원주시	42
2	c038336	2003-10-10	F	1968	강원도	춘천시	48
3	c084237	2007-03-09	F	1982	강원도	강릉시	34
4	c162600	2010-06-14	F	1978	강원도	속초시	38

In [430…
```
#[문제40] 고객이 몇십대 인지 'Age' 컬럼을 활용하여 'AgeGroup'(연령대) 컬럼을 추가 합니다.
```

In [431…
```
# 연령대 = 나이 / 10 * 10
customers['AgeGroup'] = customers['Age'] // 10 * 10
```

In [432…
```
customers.head()
```

Out[432]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50
1	c281448	2013-06-18	F	1974	강원도	원주시	42	40
2	c038336	2003-10-10	F	1968	강원도	춘천시	48	40
3	c084237	2007-03-09	F	1982	강원도	강릉시	34	30
4	c162600	2010-06-14	F	1978	강원도	속초시	38	30

In [433…
```
#[문제41] 'customers' 데이터의 상단 5행을 출력하세요.
```

In [434…
```
customers.head()
```

Out[434]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50
1	c281448	2013-06-18	F	1974	강원도	원주시	42	40
2	c038336	2003-10-10	F	1968	강원도	춘천시	48	40
3	c084237	2007-03-09	F	1982	강원도	강릉시	34	30
4	c162600	2010-06-14	F	1978	강원도	속초시	38	30

In [435…
```
#[추가문제42] 고객의 성별과 가입연도에 대한 가변수를 만들고 'encoded_data'에 저장하세요.
```

In [436…
```
# customers, Sales 데이터프레임을 CustomerID 컬럼 기준 병합, 'merged_data'에 할당
encoded_data = pd.merge(customers, sales, on='CustomerID', how='outer')
# Gender 컬럼 기반 가변수화된 데이터프레임 'encoded_data' 생성(각 범주에 해당하면 1, 아니면 0)
dumm_cols = ['Gender']
encoded_data = pd.get_dummies(encoded_data, columns=dumm_cols, drop_first=True, dtype=int)

# Year 컬럼 기반 다시 한번 'encoded_data'를 가변수화
dumm_cols2 = ['Year']
encoded_data = pd.get_dummies(encoded_data, columns=dumm_cols2, drop_first=True, dtype=int)

encoded_data
```

Out[436]:

	CustomerID	RegisterDate	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID	Seq	OrderDat
0	c328222	2014-09-25	1960	강원도	원주시	56	50	155	1	2016-02 2
1	c328222	2014-09-25	1960	강원도	원주시	56	50	143	5	2016-03 0
2	c328222	2014-09-25	1960	강원도	원주시	56	50	143	8	2016-03 0
3	c328222	2014-09-25	1960	강원도	원주시	56	50	143	10	2016-03 0
4	c328222	2014-09-25	1960	강원도	원주시	56	50	143	11	2016-03 0
...	...	...	...	...	...	...	...	...	...	
70007	c181797	2011-01-26	1981	경기도	화성시	35	30	73	1	2016-06 2
70008	c181797	2011-01-26	1981	경기도	화성시	35	30	73	2	2016-06 2
70009	c181797	2011-01-26	1981	경기도	화성시	35	30	73	5	2016-06 2
70010	c181797	2011-01-26	1981	경기도	화성시	35	30	73	7	2016-06 2
70011	c181797	2011-01-26	1981	경기도	화성시	35	30	125	2	2016-09 0

70012 rows × 16 columns

3.고객 이탈 정의

(1) Label 만들기

- **Label이란?**
  - 라벨링이란 데이터의 의미 있는 값을 부여하여 구분하거나 분류하는 작업입니다.
  - 라벨링을 통해 데이터를 분석하거나 모델링하는 과정을 더 의미있게 만듭니다.
  - 예를 들면 이메일이 '스팸'인지 '정상'인지를 분류하거나, 데이터를 그룹화하거나 참인지 거짓인지 이진 분류하거나, 순서를 표현하거나, 카테고리화하는 등 데이터의 특성에 따라 라벨링을 적용합니다.
- **Target이란?**
  - 타겟은 데이터 분석에서 예측하거나 분류하고자 하는 대상을 말합니다. 즉 우리가 예측하려는 값입니다.
  - 즉 타겟은 예측하거나 분류하려는 대상이나 범주이며, 타겟변수는 그 값을 의미합니다.
  - 라벨과 타겟은 비슷한 개념이지만 라벨은 데이터의 특성이나 카테고리를 설명하기 위해 나타내는 값이며, 타겟은 모델이 예측하려는 대상 결과값이라고 이해해주세요.

- **세부 요구사항**
  - 고객의 이탈여부에 해당하는 Target변수를 생성하시오.
  - 대상 고객
    - 2014 ~ 2016년 신규 가입 고객 이면서,
    - 2016년 하반기에 한번 이상 방문한 고객이 대상 고객입니다.
  - Labeling
    - 위 대상 고객 중, 2017년 1~3월(3개월)동안 방문(구매)하지 않은 사람은 이탈로 간주합니다.
    - 이탈고객은 1, 이탈하지 않은 고객은 0으로 정의

1) 대상고객 만들기

In [437…
```
customers.head()
```

Out[437]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50
1	c281448	2013-06-18	F	1974	강원도	원주시	42	40
2	c038336	2003-10-10	F	1968	강원도	춘천시	48	40
3	c084237	2007-03-09	F	1982	강원도	강릉시	34	30
4	c162600	2010-06-14	F	1978	강원도	속초시	38	30

In [438…
```
#[문제43] 'customers' 데이터에서 2014 ~ 2016년 신규 가입 고객을 'cust01'에 할당하세요.
```

In [439…
```
# loc함수, between('2014-01-01', '2016-12-31') 활용
cust01 = customers.loc[customers['RegisterDate'].between('2014-01-01', '2016-12-31', inclusive
```

In [440…
```
#[문제44] 'cust01'을 출력하세요.
```

In [441…
```
cust01
```

Out[441]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50
12	c354310	2015-07-16	F	1964	강원도	원주시	52	50
16	c390828	2016-10-15	F	1984	강원도	원주시	32	30
28	c386399	2016-08-18	F	1972	강원도	원주시	44	40
30	c367116	2015-12-29	F	1970	강원도	원주시	46	40
...	...	...	...	...	...	...	...	...
2221	c316736	2014-05-17	F	1976	인천광역시	중구	40	40
2225	c310508	2014-03-12	F	1981	전라남도	해남군	35	30
2228	c352708	2015-06-26	F	1980	충청남도	당진시	36	30
2236	c395061	2016-12-08	F	1958	충청북도	청원군	58	50
2240	c314668	2014-04-25	F	1977	충청북도	청주시	39	30

726 rows × 8 columns

In [442...]
```
sales.head()
```

Out[442]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1
1	69	1	2016-01-02	p1152861	1	1091	c212716	2016	1
2	69	7	2016-01-02	p1013161	1	2600	c212716	2016	1
3	69	8	2016-01-02	p1005771	1	1650	c212716	2016	1
4	69	11	2016-01-02	p1089531	1	2600	c212716	2016	1

In [443...]
```
#[문제45] 'sales' 데이터에서 2016년 하반기에 방문한 고객을 'cust02'에 할당하세요.
```

In [444...]
```
# loc 함수, between('2016-07-01', '2016-12-31')
cust02 = sales.loc[sales['OrderDate'].between('2016-07-01', '2016-12-31', inclusive='both')]
```

In [445...]
```
cust02
```

Out[445]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
29519	89	1	2016-07-01	p1030071	1	2450	c127482	2016	7
29520	89	2	2016-07-01	p1144371	1	1250	c127482	2016	7
29521	89	4	2016-07-01	p1159481	1	1650	c127482	2016	7
29522	101	1	2016-07-01	p1011291	1	273	c217179	2016	7
29523	103	1	2016-07-01	p1175481	1	1300	c222561	2016	7
...	...	...	...	...	...	...	...	...	...
55971	97	5	2016-12-31	p1005891	2	3900	c259362	2016	12
55972	99	1	2016-12-31	p1012751	1	1850	c350918	2016	12
55973	99	2	2016-12-31	p1159481	1	2200	c350918	2016	12
55974	99	3	2016-12-31	p1207281	1	3300	c350918	2016	12
55975	103	3	2016-12-31	p1002841	1	15800	c153641	2016	12

26457 rows × 9 columns

In [446…
```
#[문제46] 2016년 하반기에 한번 이상 방문한 고객을 'cust02'에 할당하세요.
```

In [447…
```
# drop_duplicates 함수, 중복된 고객 제거
cust02 = cust02.drop_duplicates(subset='CustomerID', keep='first')
```

In [448…
```
#[문제47] 'cust02'을 출력하세요.
```

In [449…
```
cust02
```

Out[449]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
29519	89	1	2016-07-01	p1030071	1	2450	c127482	2016	7
29522	101	1	2016-07-01	p1011291	1	273	c217179	2016	7
29523	103	1	2016-07-01	p1175481	1	1300	c222561	2016	7
29526	107	9	2016-07-01	p1011291	2	546	c333911	2016	7
29527	111	3	2016-07-01	p1030071	1	2450	c373441	2016	7
...	...	...	...	...	...	...	...	...	...
55591	37	5	2016-12-29	p1175481	1	1150	c153641	2016	12
55627	97	2	2016-12-30	p1005771	2	3700	c323093	2016	12
55670	25	2	2016-12-30	p1005891	1	1950	c313636	2016	12
55836	67	5	2016-12-31	p1207281	1	3300	c188728	2016	12
55866	165	2	2016-12-31	p1097821	1	1000	c394007	2016	12

1711 rows × 9 columns

In [450…
```
cust01.head()
```

Out[450]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50
12	c354310	2015-07-16	F	1964	강원도	원주시	52	50
16	c390828	2016-10-15	F	1984	강원도	원주시	32	30
28	c386399	2016-08-18	F	1972	강원도	원주시	44	40
30	c367116	2015-12-29	F	1970	강원도	원주시	46	40

In [451…
```
cust02.head()
```

Out[451]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
29519	89	1	2016-07-01	p1030071	1	2450	c127482	2016	7
29522	101	1	2016-07-01	p1011291	1	273	c217179	2016	7
29523	103	1	2016-07-01	p1175481	1	1300	c222561	2016	7
29526	107	9	2016-07-01	p1011291	2	546	c333911	2016	7
29527	111	3	2016-07-01	p1030071	1	2450	c373441	2016	7

In [452…
```
#[문제48] 대상고객 목록인 'cust01', 'cust02' 데이터를 합쳐서 'cust_churn0'으로 할당하세요.
```

In [453…
```
# merge 함수
cust_churn0 = pd.merge(cust01, cust02, on='CustomerID', how='inner')
```

In [454…
```
#[문제49] 'cust_churn0'을 상단 5행 출력하세요.
```

In [455…
```
cust_churn0
```

Out[455]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID	Seq	Or
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50	99	1	
1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	181	3	
2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	49	14	
3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	121	4	
4	c367116	2015-12-29	F	1970	강원도	원주시	46	40	169	3	
...	...	...	...	...	...	...	...	...	...	...	
542	c341674	2015-02-28	M	1973	인천광역시	중구	43	40	47	7	
543	c316736	2014-05-17	F	1976	인천광역시	중구	40	40	193	3	
544	c310508	2014-03-12	F	1981	전라남도	해남군	35	30	155	1	
545	c352708	2015-06-26	F	1980	충청남도	당진시	36	30	125	3	
546	c395061	2016-12-08	F	1958	충청북도	청원군	58	50	79	1	

547 rows × 16 columns

2) Label 만들기

In [456…
```
#[문제50] sales 데이터프레임에서 2017년 1~3월 구매 고객을 'cust03'에 할당하세요
```

In [457…
```
sales.head()
```

Out[457]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1
1	69	1	2016-01-02	p1152861	1	1091	c212716	2016	1
2	69	7	2016-01-02	p1013161	1	2600	c212716	2016	1
3	69	8	2016-01-02	p1005771	1	1650	c212716	2016	1
4	69	11	2016-01-02	p1089531	1	2600	c212716	2016	1

In [458…
```
# loc 함수 : 인덱싱을 사용하여 데이터프레임 내에서 조건을 만족하는 행 선택
# between('2017-01-01', '2017-03-31') 함수 활용, 중복된 CustomerID 삭제하기
```

```
# 'keep=first'로 중복된 값 중 첫번째 값 유지
cust03 = sales.loc[sales['OrderDate'].between('2017-01-01', '2017-03-31', inclusive='both')]
```

In [459…
```
cust03
```

Out[459]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
55976	5	2	2017-01-02	p1133371	7	7000	c271068	2017	1
55977	5	6	2017-01-02	p1012751	1	1850	c271068	2017	1
55978	5	10	2017-01-02	p1144661	1	1950	c271068	2017	1
55979	5	11	2017-01-02	p1207281	1	3300	c271068	2017	1
55980	5	12	2017-01-02	p1299491	1	1950	c271068	2017	1
...	...	...	...	...	...	...	...	...	...
70007	53	4	2017-03-31	p1072601	1	4600	c337999	2017	3
70008	53	6	2017-03-31	p1178011	1	8800	c337999	2017	3
70009	55	6	2017-03-31	p1054261	1	2091	c088320	2017	3
70010	59	4	2017-03-31	p1175481	1	1300	c238056	2017	3
70011	59	5	2017-03-31	p1013161	1	2950	c238056	2017	3

14036 rows × 9 columns

In [460…
```
cust03 = cust03.drop_duplicates(subset='CustomerID', keep='first')
```

In [461…
```
cust03
```

Out[461]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
55976	5	2	2017-01-02	p1133371	7	7000	c271068	2017	1
55982	9	15	2017-01-02	p1207281	1	3300	c146132	2017	1
55983	17	8	2017-01-02	p1012811	1	3250	c226242	2017	1
55984	21	1	2017-01-02	p1013161	2	5722	c140658	2017	1
55985	25	6	2017-01-02	p1255281	1	1182	c328839	2017	1
...	...	...	...	...	...	...	...	...	...
69871	35	2	2017-03-31	p1284851	1	2750	c400253	2017	3
69883	53	1	2017-03-31	p1002841	1	10700	c044830	2017	3
69888	63	2	2017-03-31	p1256521	1	1850	c346882	2017	3
69904	87	2	2017-03-31	p1005771	1	1850	c095005	2017	3
69957	193	2	2017-03-31	p1012811	1	2300	c017735	2017	3

1388 rows × 9 columns

In [462…
```
#[문제51] cust03에 'churn' 열을 추가하고, 해당 열의 모든 값에 0을 할당하세요.
```

In [463…
```
cust03['churn'] = 0
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_17792\2098238762.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cust03['churn'] = 0
```

In [464…
```
cust03
```

Out[464]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month	churn
55976	5	2	2017-01-02	p1133371	7	7000	c271068	2017	1	0
55982	9	15	2017-01-02	p1207281	1	3300	c146132	2017	1	0
55983	17	8	2017-01-02	p1012811	1	3250	c226242	2017	1	0
55984	21	1	2017-01-02	p1013161	2	5722	c140658	2017	1	0
55985	25	6	2017-01-02	p1255281	1	1182	c328839	2017	1	0
…	…	…	…	…	…	…	…	…	…	…
69871	35	2	2017-03-31	p1284851	1	2750	c400253	2017	3	0
69883	53	1	2017-03-31	p1002841	1	10700	c044830	2017	3	0
69888	63	2	2017-03-31	p1256521	1	1850	c346882	2017	3	0
69904	87	2	2017-03-31	p1005771	1	1850	c095005	2017	3	0
69957	193	2	2017-03-31	p1012811	1	2300	c017735	2017	3	0

1388 rows × 10 columns

In [465…
```
cust_churn0.head()
```

Out[465]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID	Seq	Orde
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50	99	1	20
1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	181	3	20
2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	49	14	20
3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	121	4	20
4	c367116	2015-12-29	F	1970	강원도	원주시	46	40	169	3	20

In [466…
```
cust03.head()
```

Out[466]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month	churn
55976	5	2	2017-01-02	p1133371	7	7000	c271068	2017	1	0
55982	9	15	2017-01-02	p1207281	1	3300	c146132	2017	1	0
55983	17	8	2017-01-02	p1012811	1	3250	c226242	2017	1	0
55984	21	1	2017-01-02	p1013161	2	5722	c140658	2017	1	0
55985	25	6	2017-01-02	p1255281	1	1182	c328839	2017	1	0

In [467…
```
#[문제52] 대상고객 목록인 'cust_churn0'과 2017년 1~3월 구매고객인 'cust03'을 합쳐서 'cust_chur
```

In [468…
```
# merge 함수, how = 'left'로 왼쪽 데이터프레임의 행은 유지
cust_churn0 = pd.merge(cust_churn0, cust03, on='CustomerID', how='left')
```

In [469…
```
#[문제53] 'cust_churn0'을 상단 5행 출력하세요.
```

In [470…
```
cust_churn0.head()
```

Out[470]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID_x	Seq_x	.
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50	99	1	
1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	181	3	
2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	49	14	
3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	121	4	
4	c367116	2015-12-29	F	1970	강원도	원주시	46	40	169	3	

5 rows × 25 columns

In [471…
```
#[문제54] cust_churn0 데이터의 churn 열의 NAN은 이탈 고객이다. 이탈고객은 1로 채워주세요.
```

In [472…
```
# fillna 함수 활용 : 데이터프레임에서 결측값(NaN)을 원하는 값으로 채우는 함수, inplace=True
cust_churn0.fillna(1, inplace=True)
```

In [473…
```
cust_churn0
```

Out[473]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID_x	Seq_x
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50	99	1
1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	181	3
2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	49	14
3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	121	4
4	c367116	2015-12-29	F	1970	강원도	원주시	46	40	169	3
...	...	...	...	...	...	...	...	...	...	...
542	c341674	2015-02-28	M	1973	인천광역시	중구	43	40	47	7
543	c316736	2014-05-17	F	1976	인천광역시	중구	40	40	193	3
544	c310508	2014-03-12	F	1981	전라남도	해남군	35	30	155	1
545	c352708	2015-06-26	F	1980	충청남도	당진시	36	30	125	3
546	c395061	2016-12-08	F	1958	충청북도	청원군	58	50	79	1

547 rows × 25 columns

In [474…
```
#[문제55] 데이터프레임 cust_churn0의 'churn' 열의 데이터 타입을 정수형(int64)으로 변환하세요.
```

In [475…
```
# astype('int64')
cust_churn0['churn'] = cust_churn0['churn'].astype('int64')
```

In [476…
```
cust_churn0['churn'].dtypes
```

Out[476]:
```
dtype('int64')
```

In [477…
```
#[문제56] 데이터프레임 cust_churn0의 각 값들의 개수를 반환하여 고객 이탈 여부의 빈도수를 확인하
```

In [478…
```
# value_counts() 함수 활용
#cust_churn0.value_counts('churn')
cust_churn0['churn'].value_counts()
```

Out[478]:
```
churn
0    363
1    184
Name: count, dtype: int64
```

(2) feature 추가하기

- **feature란**
  - Feature란 데이터에서 추출한 변수나 속성을 의미합니다.
  - 각각의 열을 feature라고 간주할 수 있습니다.
  - 올바른 feature를 선택하고 가공하여 생성하는 것은 데이터 분석에서 모델을 학습하거나 예측을 수행하는데 중요합니다.
- **세부 요구사항**
  - 기본 feature 3가지를 생성해봅니다.
    - 1) 가입연차
    - 2) 최근 3개월(2016년10~12월)간 방문 횟수
    - 3) 최근 3개월(2016년10~12월)간 구매금액

feature 3가지 생성

- ① 가입연수(RegDuration)
- ② 최근 3개월(2016년10~12월)간 방문 횟수
- ③ 최근 3개월(2016년10~12월)간 구매금액

In [479…
```
customers.head()
```

Out[479]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50
1	c281448	2013-06-18	F	1974	강원도	원주시	42	40
2	c038336	2003-10-10	F	1968	강원도	춘천시	48	40
3	c084237	2007-03-09	F	1982	강원도	강릉시	34	30
4	c162600	2010-06-14	F	1978	강원도	속초시	38	30

In [480…
```
#<① 가입연수>
#[문제57] 'RegisterDate' 열의 데이터는 datetime 형식으로 변환합니다.
```

In [481…
```
#pd.to_datetime 함수
RegisterDate = pd.to_datetime(customers['RegisterDate'])
```

In [482…
```
cust_churn0.head()
```

Out[482]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID_x	Seq_x	.
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50	99	1	
1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	181	3	
2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	49	14	
3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	121	4	
4	c367116	2015-12-29	F	1970	강원도	원주시	46	40	169	3	

5 rows × 25 columns

In [483…
```
#[문제58] 2016-12-31 기준으로 가입 연수 'RegDuration' 열을 'cust_churn0'에 추가하여 출력해주세
```

In [484…
```
#가입연수 : 2016년 - 등록연도 (RegisterDate.dt.year)
cust_churn0['RegDuration'] = 2016 - RegisterDate.dt.year
```

In [485…
```
cust_churn0.head()
```

Out[485]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID_x	Seq_x	.
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50	99	1	
1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	181	3	
2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	49	14	
3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	121	4	
4	c367116	2015-12-29	F	1970	강원도	원주시	46	40	169	3	

5 rows × 26 columns

In [486…
```
sales.head()
```

Out[486]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
0	107	2	2016-01-02	p1036481	2	2100	c150417	2016	1
1	69	1	2016-01-02	p1152861	1	1091	c212716	2016	1
2	69	7	2016-01-02	p1013161	1	2600	c212716	2016	1
3	69	8	2016-01-02	p1005771	1	1650	c212716	2016	1
4	69	11	2016-01-02	p1089531	1	2600	c212716	2016	1

In [487…
```
#<② 3개월간 방문 횟수>
#[문제59] sales에서 주문일자('OrderDate')가 '2016-10-01'부터 '2016-12-31' 사이인 데이터를 선택
```

In [488…
```
# loc 함수, between
tmp = sales.loc[sales['OrderDate'].between('2016-10-01', '2016-12-31', inclusive='both')]
```

In [489…
```
tmp
```

Out[489]:

	OrderID	Seq	OrderDate	ProductID	Qty	Amt	CustomerID	Year	Month
43455	57	6	2016-10-01	p1012951	1	1850	c155875	2016	10
43456	59	3	2016-10-01	p1054261	1	1864	c177682	2016	10
43457	63	3	2016-10-01	p1207281	1	3450	c314136	2016	10
43458	63	7	2016-10-01	p1072601	1	4600	c314136	2016	10
43459	63	8	2016-10-01	p1255281	1	1182	c314136	2016	10
…	…	…	…	…	…	…	…	…	…
55971	97	5	2016-12-31	p1005891	2	3900	c259362	2016	12
55972	99	1	2016-12-31	p1012751	1	1850	c350918	2016	12
55973	99	2	2016-12-31	p1159481	1	2200	c350918	2016	12
55974	99	3	2016-12-31	p1207281	1	3300	c350918	2016	12
55975	103	3	2016-12-31	p1002841	1	15800	c153641	2016	12

12521 rows × 9 columns

In [490…
```
#<② 3개월간 방문 횟수>
#[문제60] 'OrderID', 'OrderDate', 'CustomerID' 열만을 필터링해서 중복된 행을 제거하고 'temp'에
```

In [491…
```
# drop_duplicates() 함수 활용
del_dupli = ['OrderID', 'OrderDate', 'CustomerID']
temp = tmp[del_dupli].drop_duplicates()
#temp = tmp.drop_duplicates(subset=['OrderID', 'OrderDate', 'CustomerID'], keep='first')
```

In [492…
```
temp
```

Out[492]:

	OrderID	OrderDate	CustomerID
43455	57	2016-10-01	c155875
43456	59	2016-10-01	c177682
43457	63	2016-10-01	c314136
43461	65	2016-10-01	c319923
43463	69	2016-10-01	c302963
...	...	...	...
55964	91	2016-12-31	c038319
55965	95	2016-12-31	c132501
55968	97	2016-12-31	c259362
55972	99	2016-12-31	c350918
55975	103	2016-12-31	c153641

6372 rows × 3 columns

In [493…
```
#<② 3개월간 방문 횟수>
#[문제61] 'temp' 데이터에서 각 고객별 'OrderDate'열의 개수를 세어서 'temp2'에 저장하세요.
```

In [494…
```
# groupy() 함수
temp2 = temp.groupby(by='CustomerID', as_index=False)[['OrderDate']].count()
```

In [495…
```
temp2
```

Out[495]:

	CustomerID	OrderDate
0	c017487	9
1	c017503	2
2	c017517	12
3	c017522	7
4	c017526	1
...	...	...
1377	c395401	2
1378	c395638	2
1379	c395673	1
1380	c396022	1
1381	c396059	1

1382 rows × 2 columns

In [496…
```
#<② 3개월간 방문 횟수>
#[문제62] 'temp2' 데이터의 'OrderDate' 열의 이름을 'Visit_3M_Cnt'로 변경하고 확인해주세요.
```

In [497…
```
# rename() 함수
temp2.columns = ['CustomerID' , 'Visit_3M_Cnt']
```

In [498…
```
temp2
```

Out[498]:

	CustomerID	Visit_3M_Cnt
0	c017487	9
1	c017503	2
2	c017517	12
3	c017522	7
4	c017526	1
...	...	...
1377	c395401	2
1378	c395638	2
1379	c395673	1
1380	c396022	1
1381	c396059	1

1382 rows × 2 columns

In [499…
```
cust_churn0.head()
```

Out[499]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID_x	Seq_x	.
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50	99	1	
1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	181	3	
2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	49	14	
3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	121	4	
4	c367116	2015-12-29	F	1970	강원도	원주시	46	40	169	3	

5 rows × 26 columns

In [500…
```
temp2.head()
```

Out[500]:

	CustomerID	Visit_3M_Cnt
0	c017487	9
1	c017503	2
2	c017517	12
3	c017522	7
4	c017526	1

In [501…
```
#<② 3개월간 방문 횟수>
#[문제63] cust_churn0과 temp2을 합쳐서 'cust_churn1'으로 저장하세요. how= 'left'로 기존 데이터
```

In [502…
```
# pd.merge 함수
cust_churn1 = pd.merge(cust_churn0, temp2, on='CustomerID', how='left')
```

In [503…
```
#[문제64] cust_churn1 출력하고 확인해주세요.
```

In [504…
```
cust_churn1
```

Out[504]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID_x	Seq_x
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50	99	1
1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	181	3
2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	49	14
3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	121	4
4	c367116	2015-12-29	F	1970	강원도	원주시	46	40	169	3
…	…	…	…	…	…	…	…	…	…	…
542	c341674	2015-02-28	M	1973	인천광역시	중구	43	40	47	7
543	c316736	2014-05-17	F	1976	인천광역시	중구	40	40	193	3
544	c310508	2014-03-12	F	1981	전라남도	해남군	35	30	155	1
545	c352708	2015-06-26	F	1980	충청남도	당진시	36	30	125	3
546	c395061	2016-12-08	F	1958	충청북도	청원군	58	50	79	1

547 rows × 27 columns

In [505…
```
temp.head()
```

Out[505]:

	OrderID	OrderDate	CustomerID
43455	57	2016-10-01	c155875
43456	59	2016-10-01	c177682
43457	63	2016-10-01	c314136
43461	65	2016-10-01	c319923
43463	69	2016-10-01	c302963

In [506…
```
#<③ 3개월간 구매금액>
#[문제65] 최근 3개월(2016-10-01~2016-12-31)간 고객별 구매금액을 계산하여 'temp'에 저장해주세요
```

In [507…
```
# loc 함수, between
temp = cust_churn1.loc[cust_churn1['OrderDate_x'].between('2016-10-01', '2016-12-31', inclusiv
```

In [508…
```
temp
```

Out[508]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID_x	Seq_x
1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	181	3
2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	49	14
3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	121	4
12	c347315	2015-04-27	F	1971	강원도	횡성군	45	40	103	5
15	c341313	2015-02-24	F	1979	경기도	광명시	37	30	113	2
...	...	...	...	...	...	...	...	...	...	...
535	c366435	2015-12-19	F	1975	인천광역시	연수구	41	40	179	2
536	c391925	2016-10-27	F	1941	인천광역시	연수구	75	70	127	3
537	c315320	2014-05-02	M	1980	인천광역시	연수구	36	30	71	4
538	c332026	2014-11-01	F	1977	인천광역시	연수구	39	30	139	1
546	c395061	2016-12-08	F	1958	충청북도	청원군	58	50	79	1

118 rows × 27 columns

In [509…
```
#<③ 3개월간 구매금액>
#[문제66] 데이터프레임 temp를 'CustomerID'를 기준으로 그룹화하고 각 고객별로 'Amt' 열의 합을 ㅈ
```

In [510…
```
# groupby 함수
temp2 = temp.groupby(by='CustomerID', as_index=False)[['Amt_x']].sum()
```

In [511…
```
temp2
```

Out[511]:

	CustomerID	Amt_x
0	c305675	4600
1	c307519	1850
2	c307671	2091
3	c309295	1900
4	c309376	1050
...	...	...
113	c395401	5500
114	c395638	1455
115	c395673	4200
116	c396022	1455
117	c396059	8370

118 rows × 2 columns

In [512...

```
#<③ 3개월간 구매금액>
#[문제67] 'temp2'에서 'Amt'열의 이름을 'Amt_3M_sum'으로 변경해주세요.
```

In [513...

```
# renam 함수
temp2.columns = ['CustomerID', 'Amt_3M_sum']
```

In [514...

```
temp2
```

Out[514]:

	CustomerID	Amt_3M_sum
0	c305675	4600
1	c307519	1850
2	c307671	2091
3	c309295	1900
4	c309376	1050
...	...	...
113	c395401	5500
114	c395638	1455
115	c395673	4200
116	c396022	1455
117	c396059	8370

118 rows × 2 columns

In [515…
```
#<③ 3개월간 구매금액>
#[문제68] cust_churn1, temp2을 합쳐서 'cust_churn2'으로 저장하세요.
```

In [516…
```
cust_churn1.head()
```

Out[516]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID_x	Seq_x	.
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50	99	1	
1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	181	3	
2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	49	14	
3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	121	4	
4	c367116	2015-12-29	F	1970	강원도	원주시	46	40	169	3	

5 rows × 27 columns

In [517…
```
temp2.head()
```

Out[517]:

	CustomerID	Amt_3M_sum
0	c305675	4600
1	c307519	1850
2	c307671	2091
3	c309295	1900
4	c309376	1050

In [520…
```
# pd.merge 함수
cust_churn2 = pd.merge(cust_churn1, temp2, on='CustomerID', how='left')
```

In [521…
```
#[문제69] 'cust_churn2'를 출력하고 확인하세요.
```

In [522…
```
cust_churn2.head()
```

Out[522]:

	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	OrderID_x	Seq_x	.
0	c328222	2014-09-25	F	1960	강원도	원주시	56	50	99	1	
1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	181	3	
2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	49	14	
3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	121	4	
4	c367116	2015-12-29	F	1970	강원도	원주시	46	40	169	3	

5 rows × 28 columns

In [144…
```
#[문제70] cust_churn2를 cust_churn으로 저장하세요.
```

In [523…
```
cust_churn = cust_churn2
```

4.데이터셋 저장

- **세부 요구사항**
  - to_csv를 이용하여 전처리된 데이터셋을 저장하세요.
  - 저장할 파일의 확장자는 .csv 입니다.

In [144…
```
#[문제71] 전처리된 데이터프레임 'cust_churn'을 CSV 파일로 저장합니다.
```

In [524…
```
# 파일 : 'cust_churn.csv'
# to_csv 함수 활용
cust_churn.to_csv('cust_churn.csv_1')
```

In [144…
```
#[문제72] 데이터가 잘 저장되었는지 다시 한번 불러오고 확인해보세요.
```

In [527…
```
cust_c =  pd.read_csv('cust_churn.csv_1')
cust_c
```

Out[527]:

	Unnamed: 0	CustomerID	RegisterDate	Gender	BirthYear	Addr1	Addr2	Age	AgeGroup	Order
0	0	c328222	2014-09-25	F	1960	강원도	원주시	56	50	
1	1	c354310	2015-07-16	F	1964	강원도	원주시	52	50	
2	2	c390828	2016-10-15	F	1984	강원도	원주시	32	30	
3	3	c386399	2016-08-18	F	1972	강원도	원주시	44	40	
4	4	c367116	2015-12-29	F	1970	강원도	원주시	46	40	
...	...	...	...	...	...	...	...	...	...	
542	542	c341674	2015-02-28	M	1973	인천광역시	중구	43	40	
543	543	c316736	2014-05-17	F	1976	인천광역시	중구	40	40	
544	544	c310508	2014-03-12	F	1981	전라남도	해남군	35	30	
545	545	c352708	2015-06-26	F	1980	충청남도	당진시	36	30	
546	546	c395061	2016-12-08	F	1958	충청북도	청원군	58	50	

547 rows × 29 columns

In [144...

```
## 정말 고생 많으셨습니다.
## 시간이 남으시는 분은 중급용 파일을 다시 한번 복기해보세요.
```