In [1]:

```python
#데이터 시각화 및 프레임
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import koreanize_matplotlib


# 비지도 학습 모델
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler
```

In [2]:

```python
# 데이터 읽기
df = pd.read_csv('./CC GENERAL.csv')
df
```

Out[2]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASE
0	C10001	40.900749	0.818182	95.40	0.00	95.40	0.000000	0.166667	0.000000	
1	C10002	3202.467416	0.909091	0.00	0.00	0.00	6442.945483	0.000000	0.000000	
2	C10003	2495.148862	1.000000	773.17	773.17	0.00	0.000000	1.000000	1.000000	
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.00	205.788017	0.083333	0.083333	
4	C10005	817.714335	1.000000	16.00	16.00	0.00	0.000000	0.083333	0.083333	
...	...	...	...	...	...	...	...	...	...	
8945	C19186	28.493517	1.000000	291.12	0.00	291.12	0.000000	1.000000	0.000000	
8946	C19187	19.183215	1.000000	300.00	0.00	300.00	0.000000	1.000000	0.000000	
8947	C19188	23.398673	0.833333	144.40	0.00	144.40	0.000000	0.833333	0.000000	
8948	C19189	13.457564	0.833333	0.00	0.00	0.00	36.558778	0.000000	0.000000	
8949	C19190	372.708075	0.666667	1093.25	1093.25	0.00	127.040008	0.666667	0.666667	

8950 rows × 18 columns

In [3]:

```python
#데이터 정보
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   CUST_ID                           8950 non-null   object 
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64  
 12  PURCHASES_TRX                     8950 non-null   int64  
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64  
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

In [4]:

```python
# 결측치 개수
df.isnull().sum()
```

Out[4]:

```
CUST_ID                               0
BALANCE                               0
BALANCE_FREQUENCY                     0
PURCHASES                             0
ONEOFF_PURCHASES                      0
INSTALLMENTS_PURCHASES                0
CASH_ADVANCE                          0
PURCHASES_FREQUENCY                   0
ONEOFF_PURCHASES_FREQUENCY            0
PURCHASES_INSTALLMENTS_FREQUENCY      0
CASH_ADVANCE_FREQUENCY                0
CASH_ADVANCE_TRX                      0
PURCHASES_TRX                         0
CREDIT_LIMIT                          1
PAYMENTS                              0
MINIMUM_PAYMENTS                    313
PRC_FULL_PAYMENT                      0
TENURE                                0
dtype: int64
```

데이터 전처리

In [3]:

```python
df2 = df.dropna()
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8636 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   CUST_ID                           8636 non-null   object 
 1   BALANCE                           8636 non-null   float64
 2   BALANCE_FREQUENCY                 8636 non-null   float64
 3   PURCHASES                         8636 non-null   float64
 4   ONEOFF_PURCHASES                  8636 non-null   float64
 5   INSTALLMENTS_PURCHASES            8636 non-null   float64
 6   CASH_ADVANCE                      8636 non-null   float64
 7   PURCHASES_FREQUENCY               8636 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8636 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8636 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8636 non-null   float64
 11  CASH_ADVANCE_TRX                  8636 non-null   int64  
 12  PURCHASES_TRX                     8636 non-null   int64  
 13  CREDIT_LIMIT                      8636 non-null   float64
 14  PAYMENTS                          8636 non-null   float64
 15  MINIMUM_PAYMENTS                  8636 non-null   float64
 16  PRC_FULL_PAYMENT                  8636 non-null   float64
 17  TENURE                            8636 non-null   int64  
dtypes: float64(14), int64(3), object(1)
memory usage: 1.3+ MB
```

In [4]:

```python
#drop ID column

dm = df2.drop('CUST_ID',axis = 1)

#normalize values
scaler = StandardScaler()
dscaler = scaler.fit_transform(dm)
dscaler.shape
```

Out[4]:

```
(8636, 17)
```

In [6]:

```python
data = pd.DataFrame(dscaler,columns = dm.columns)
data.head()
```

Out[6]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMEN
0	-0.744625	-0.370047	-0.429184	-0.359160	-0.354826	-0.468655	-0.820769	-0.686280	
1	0.764152	0.067679	-0.473208	-0.359160	-0.458839	2.568556	-1.236139	-0.686280	
2	0.426602	0.505405	-0.116413	0.099909	-0.458839	-0.468655	1.256077	2.646651	
3	-0.373910	0.505405	-0.465825	-0.349660	-0.458839	-0.468655	-1.028455	-0.408536	
4	0.099551	0.505405	0.142062	-0.359160	0.994815	-0.468655	0.425339	-0.686280	

DBSCAN VS KMeans

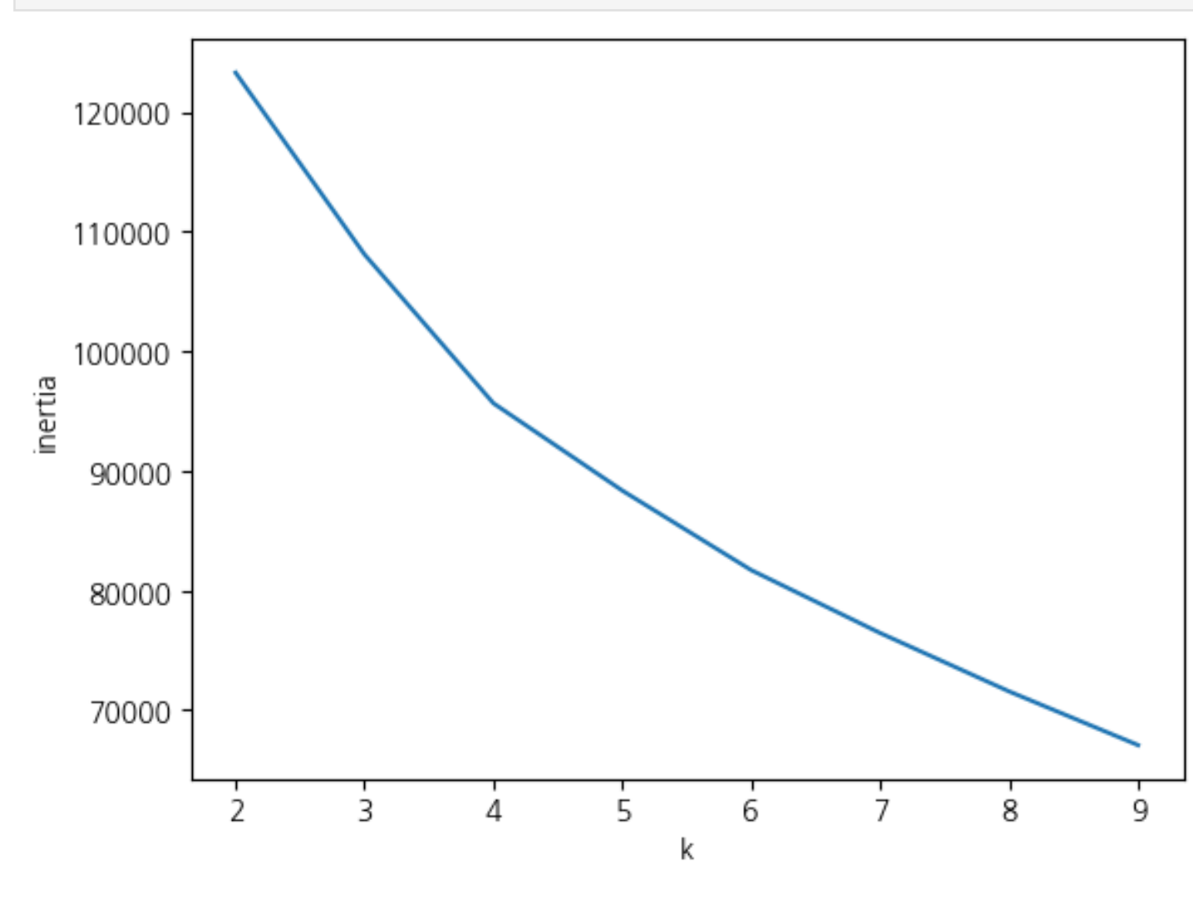
In [7]:

```python
#최적의 K 찾기 (데이터 시각화)

inertia = []

for k in range(2,10):
    km = KMeans(n_clusters = k, random_state = 42, n_init = 10)
    km.fit(data)
    inertia.append(km.inertia_)

plt.plot(range(2,10),inertia)
plt.xlabel('k')
plt.ylabel('inertia')
plt.show()
```



In [8]:

```python
#Kmeans 학습

km = KMeans(n_clusters = 4,random_state = 42,n_init = 10)
data['km'] = km.fit_predict(data)
data.head(3)
```

Out[8]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMEN
0	-0.744625	-0.370047	-0.429184	-0.359160	-0.354826	-0.468655	-0.820769	-0.686280	
1	0.764152	0.067679	-0.473208	-0.359160	-0.458839	2.568556	-1.236139	-0.686280	
2	0.426602	0.505405	-0.116413	0.099909	-0.458839	-0.468655	1.256077	2.646651	

In [22]:

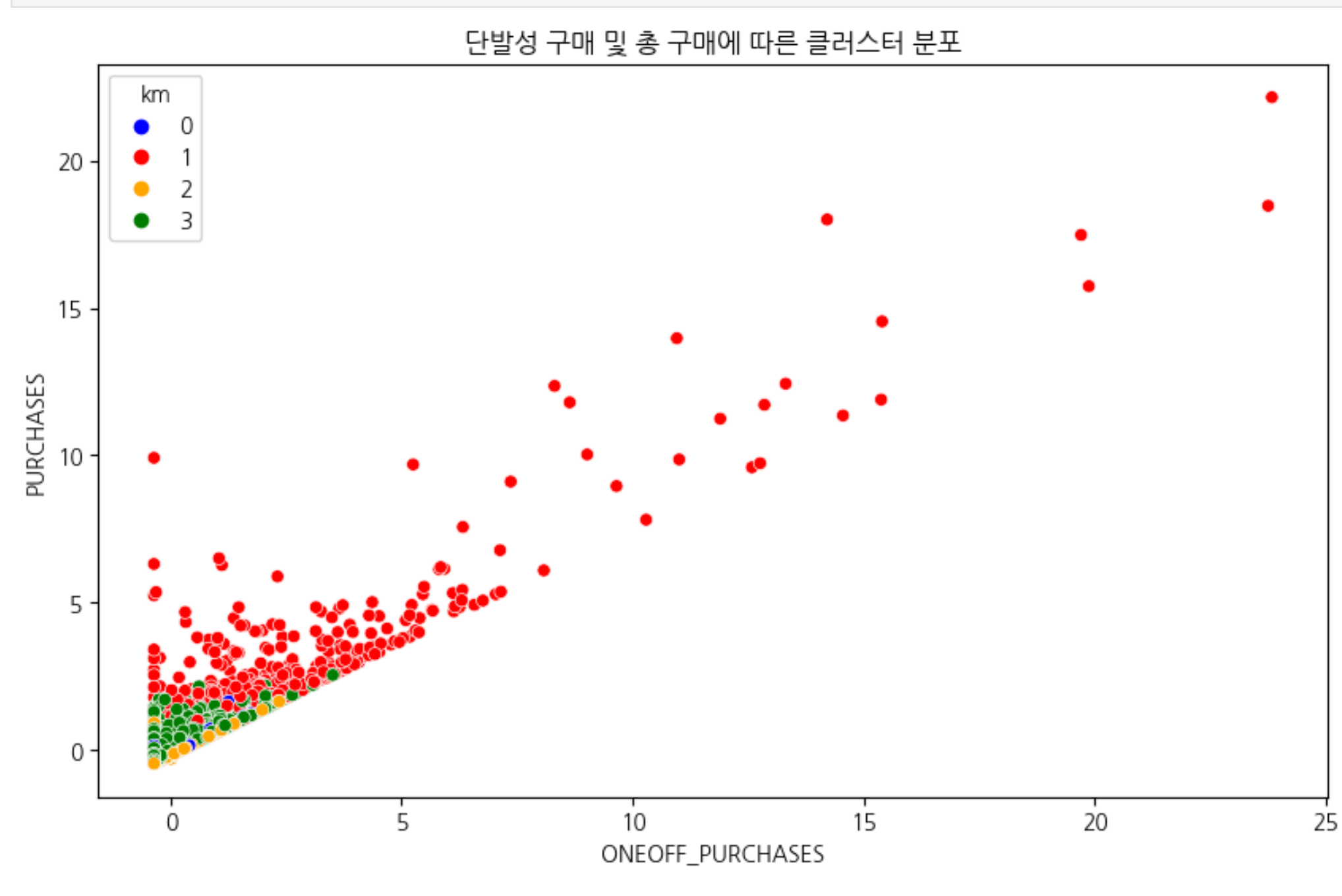
```python
data['km'].value_counts()
```

Out[22]:

```
km
2    3805
3    3282
0    1155
1     394
Name: count, dtype: int64
```

In [23]:

```python
#scatterplot
#제목 : 단발성 구매 및 총 구매에 따른 클러스터 분포
plt.figure(figsize = (10,6))
sns.scatterplot(data = data , x = 'ONEOFF_PURCHASES', y = 'PURCHASES',hue = 'km',palette={0:'Blue', 1:'Red', 2:'Orange',3:'Green'})
plt.title('단발성 구매 및 총 구매에 따른 클러스터 분포')
plt.show()
```



DBSCAN

In [9]:

```python
#DBSCAN
df3 = data.drop('km', axis = 1)
df3.head(3)
```

Out[9]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMEN
0	-0.744625	-0.370047	-0.429184	-0.359160	-0.354826	-0.468655	-0.820769	-0.686280	
1	0.764152	0.067679	-0.473208	-0.359160	-0.458839	2.568556	-1.236139	-0.686280	
2	0.426602	0.505405	-0.116413	0.099909	-0.458839	-0.468655	1.256077	2.646651	

In [10]:

```python
db = DBSCAN(eps = 3.5, min_samples = 4)
df3['dbscan'] = db.fit_predict(df3)
df3.head(3)
```

Out[10]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMEN
0	-0.744625	-0.370047	-0.429184	-0.359160	-0.354826	-0.468655	-0.820769	-0.686280	
1	0.764152	0.067679	-0.473208	-0.359160	-0.458839	2.568556	-1.236139	-0.686280	
2	0.426602	0.505405	-0.116413	0.099909	-0.458839	-0.468655	1.256077	2.646651	

In [51]:

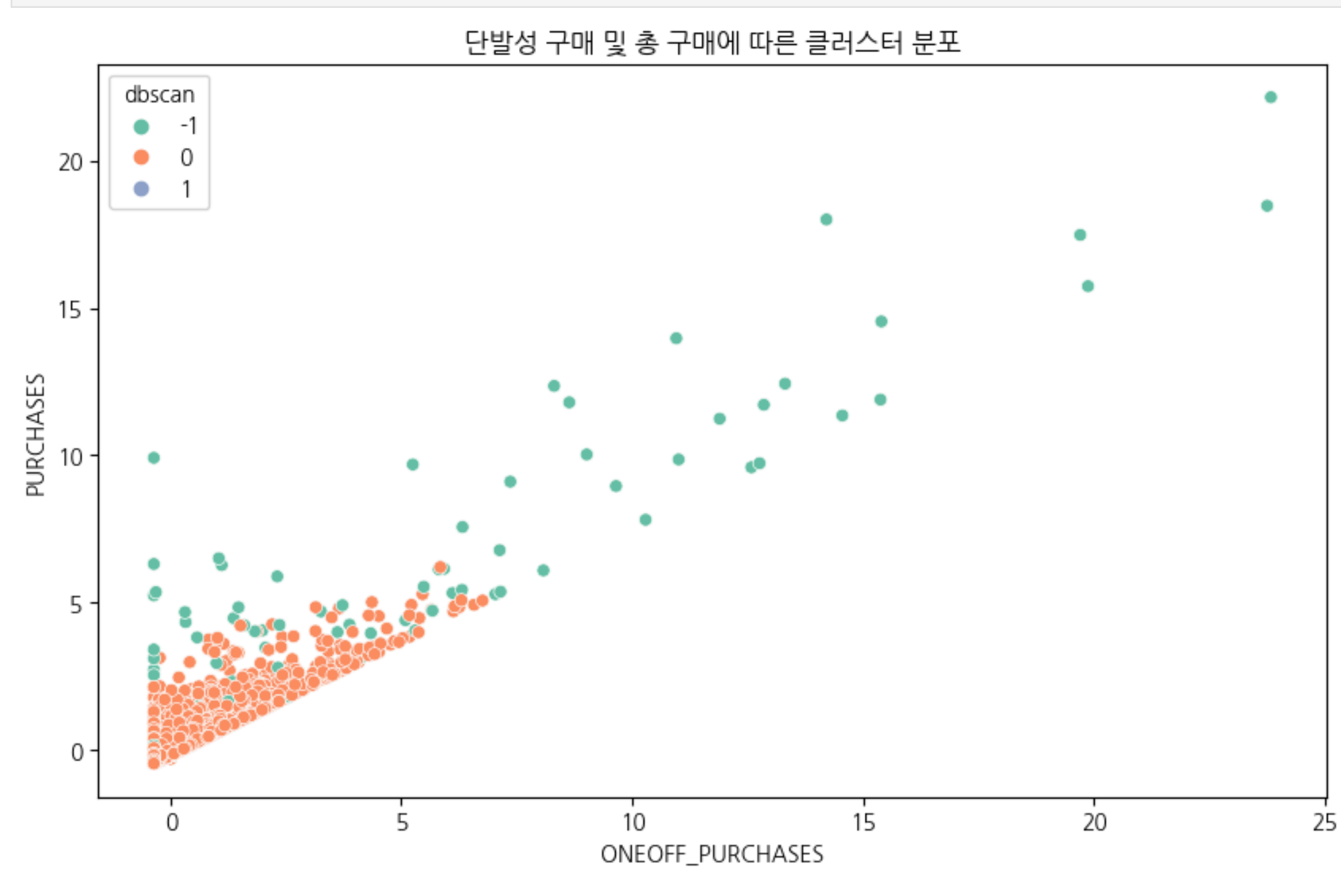
```python
df3['dbscan'].value_counts()
```

Out[51]:

```
dbscan
 0    8498
-1     135
 1       3
Name: count, dtype: int64
```

In [53]:

```python
#scatterplot
#제목 : 단발성 구매 및 총 구매에 따른 클러스터 분포
plt.figure(figsize = (10,6))
sns.scatterplot(data = df3 , x = 'ONEOFF_PURCHASES', y = 'PURCHASES',hue = 'dbscan', palette="Set2")
plt.title('단발성 구매 및 총 구매에 따른 클러스터 분포')
plt.show()
```



PCA

In [11]:

```python
data = data.drop('km', axis = 1 )
data.head(5)
```

Out[11]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMEN
0	-0.744625	-0.370047	-0.429184	-0.359160	-0.354826	-0.468655	-0.820769	-0.686280	
1	0.764152	0.067679	-0.473208	-0.359160	-0.458839	2.568556	-1.236139	-0.686280	
2	0.426602	0.505405	-0.116413	0.099909	-0.458839	-0.468655	1.256077	2.646651	
3	-0.373910	0.505405	-0.465825	-0.349660	-0.458839	-0.468655	-1.028455	-0.408536	
4	0.099551	0.505405	0.142062	-0.359160	0.994815	-0.468655	0.425339	-0.686280	

In [12]:

```python
dp = pd.DataFrame(PCA(n_components = 2).fit_transform(data),columns = ['pc1','pc2'])
dp.head(3)
```

Out[12]:

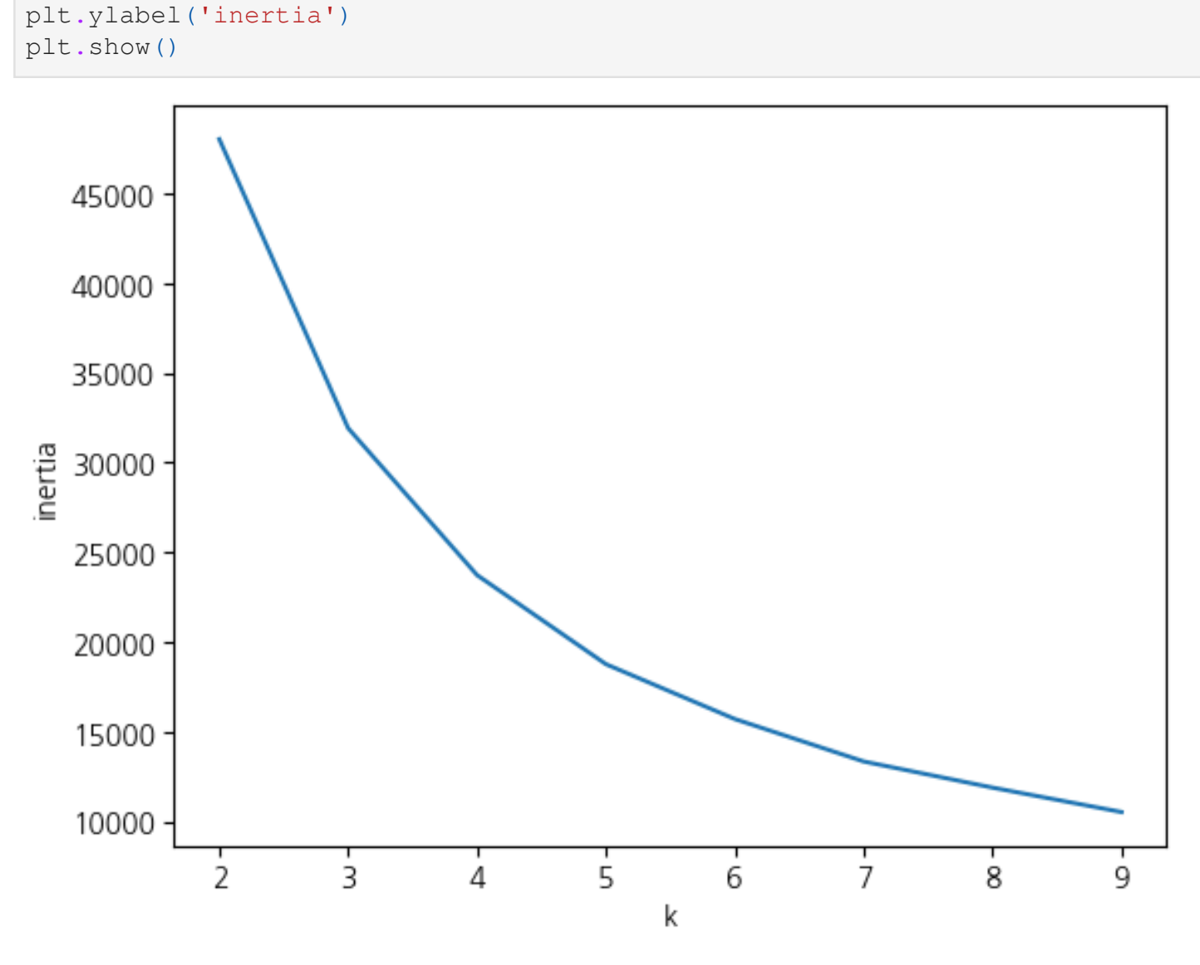
	pc1	pc2
0	-1.696396	-1.122583
1	-1.215681	2.435637
2	0.935856	-0.385216

In [13]:

```python
inertia = []

for k in range(2,10):
    km = KMeans(n_clusters = k, random_state = 42, n_init = 10)
    km.fit(dp)
    inertia.append(km.inertia_)

plt.plot(range(2,10),inertia)
plt.xlabel('k')
plt.ylabel('inertia')
plt.show()
```



In [14]:

```python
kmeans = KMeans(n_clusters = 4,random_state = 42, n_init = 10)
kmeans.fit(dp)
dp['labels'] = kmeans.predict(dp)
```

In [65]:

```python
dp.head()
```

Out[65]:

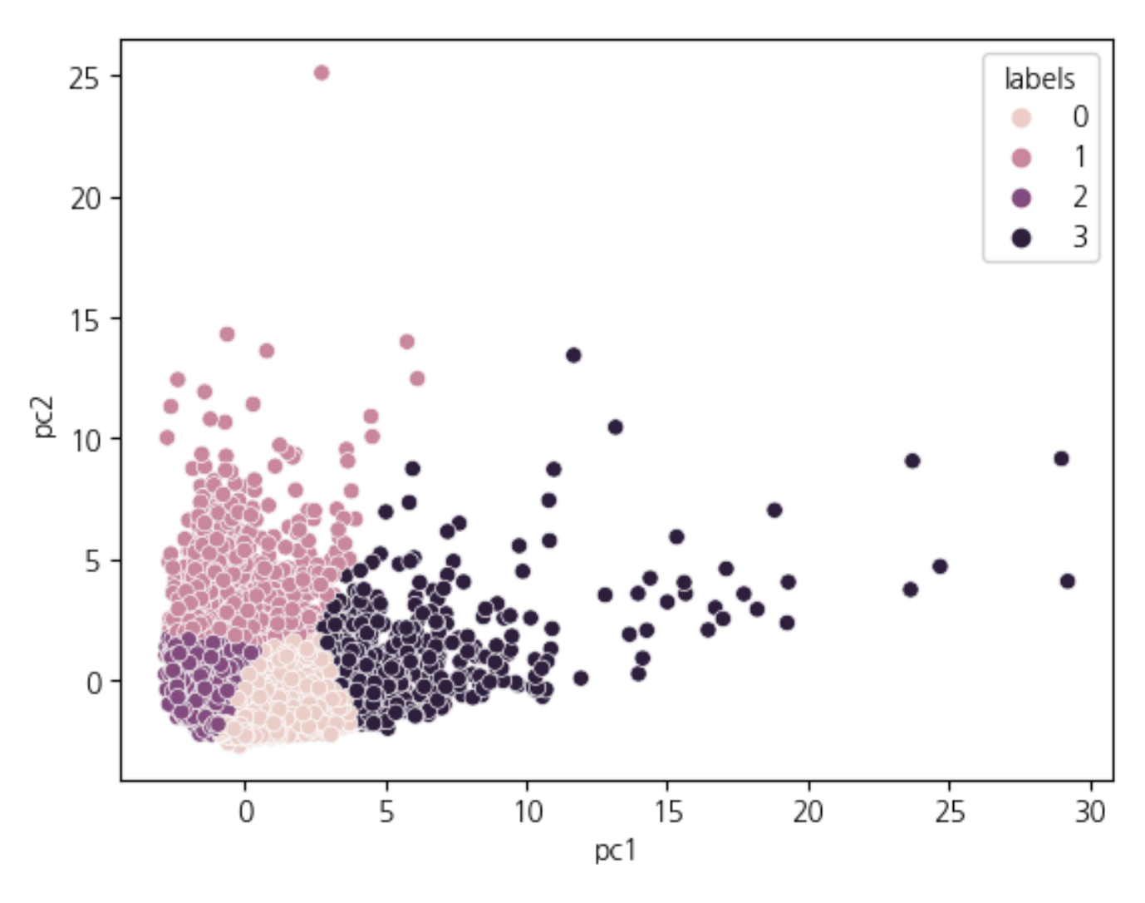
	pc1	pc2	labels
0	-1.696395	-1.122580	2
1	-1.215681	2.435663	1
2	0.935853	-0.385198	0
3	-1.614638	-0.724585	2
4	0.223701	-0.783605	0

In [66]:

```python
sns.scatterplot(data = dp, x = 'pc1',y = 'pc2', hue = 'labels')
```

Out[66]:

```
<Axes: xlabel='pc1', ylabel='pc2'>
```



t-SNE

In [15]:

```python
tsne = TSNE(n_components = 2, random_state = 42)
tdata = pd.DataFrame(tsne.fit_transform(data),columns = ['tsn1','tsn2'])
tdata
```

Out[15]:

	tsn1	tsn2
0	-40.560528	-25.943663
1	-16.982489	55.920586
2	30.747356	16.414604
3	-18.391907	-7.582566
4	8.846025	-20.203932
...	...	...
8631	-50.297924	-14.256378
8632	-22.131054	-62.668770
8633	-23.890818	-60.336018
8634	-48.640320	-6.631445
8635	-43.376095	-11.644857

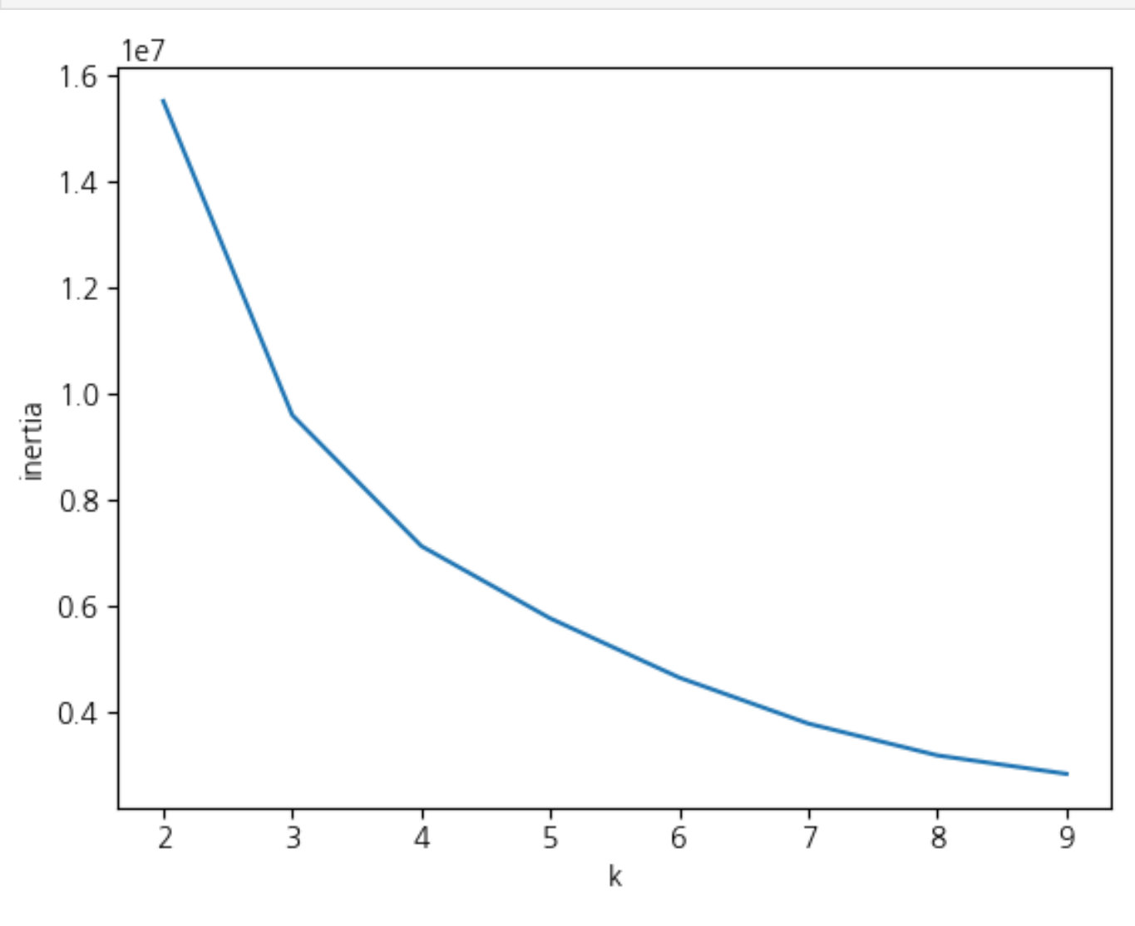
8636 rows × 2 columns

In [16]:

```python
inertia = []

for k in range(2,10):
    km = KMeans(n_clusters = k, random_state = 42, n_init = 10)
    km.fit(tdata)
    inertia.append(km.inertia_)

plt.plot(range(2,10),inertia)
plt.xlabel('k')
plt.ylabel('inertia')
plt.show()
```



In [19]:

```python
kmeans = KMeans(n_clusters = 4,random_state = 42, n_init = 10)
kmeans.fit(tdata)

tdata['tsne'] = kmeans.predict(tdata)
tdata.head(7)
```

Out[19]:

	tsn1	tsn2	tsne
0	-40.560528	-25.943663	1
1	-16.982489	55.920586	2
2	30.747356	16.414604	0
3	-18.391907	-7.582566	1
4	8.846025	-20.203932	3
5	74.675865	2.756452	0
6	24.698959	-26.956549	3

In [20]:

```python
sns.scatterplot(data = tdata, x = 'tsn1',y = 'tsn2', hue = 'tsne')
```

Out[20]:

```
<Axes: xlabel='tsn1', ylabel='tsn2'>
```