In [76]:
```
#import relevant libraries

import sklearn as sk
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

df=pd.read_csv(r"C:\Users\goex1\Desktop\Data Science Customer Segmentation\customer_segmentation.csv")

df.head(10)
```

Out[76]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumV
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	58	635	...	
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	38	11	...	
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	26	426	...	
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	26	11	...	
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	94	173	...	
5	7446	1967	Master	Together	62513.0	0	1	09-09-2013	16	520	...	
6	965	1971	Graduation	Divorced	55635.0	0	1	13-11-2012	34	235	...	
7	6177	1985	PhD	Married	33454.0	1	0	08-05-2013	32	76	...	
8	4855	1974	PhD	Together	30351.0	1	0	06-06-2013	19	14	...	
9	5899	1950	PhD	Together	5648.0	1	1	13-03-2014	68	28	...	

10 rows × 29 columns

In [77]:
```
#descriptive statistics

df.describe()
```

Out[77]:

	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMea
count	2240.000000	2240.000000	2216.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	22
mean	5592.159821	1968.805804	52247.251354	0.444196	0.506250	49.109375	303.935714	26.302232	1
std	3246.662198	11.984069	25173.076661	0.538398	0.544538	28.962453	336.597393	39.773434	2
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2828.250000	1959.000000	35303.000000	0.000000	0.000000	24.000000	23.750000	1.000000	
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000	49.000000	173.500000	8.000000	
75%	8427.750000	1977.000000	68522.000000	1.000000	1.000000	74.000000	504.250000	33.000000	2
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	17

8 rows × 26 columns

In [ ]:
```
#grafical overview from numeric and categoric columns

numeric_columns = df.select_dtypes(include=np.number).columns
categoric_columns = df.select_dtypes(include=['object', 'category']).columns

for feature in categoric_columns:
    sns.countplot(x=df[feature])
    plt.xlabel(feature)
    plt.show()


for feature in numeric_columns:
    sns.boxplot(y=df[feature])
    plt.xlabel(feature)
    plt.show()
```

In [78]:
```
#datacleaning

print(df.isna().sum())
```

```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                24
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
dtype: int64
```

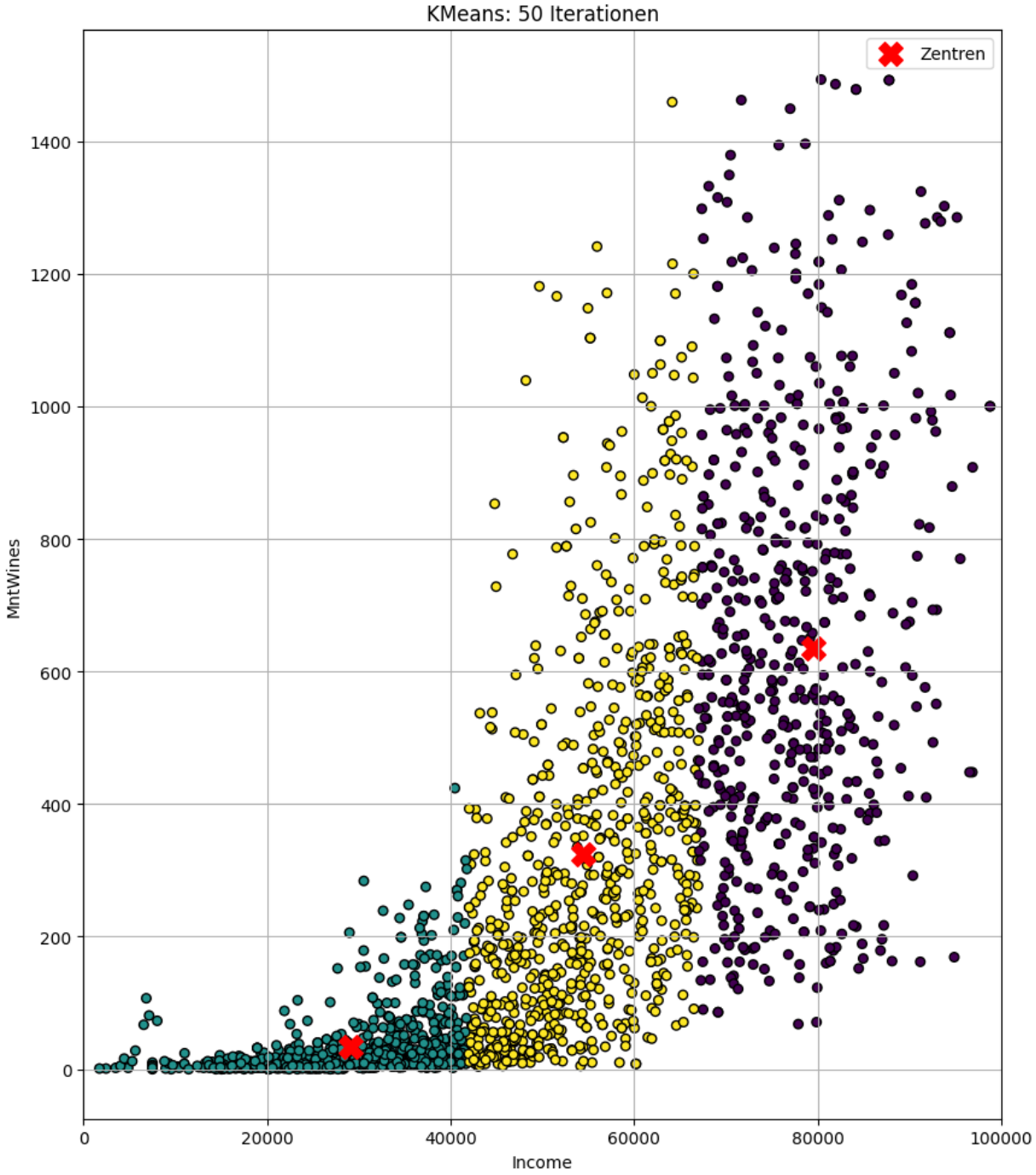
In [79]:
```
#datacleaning

df_clean = df.dropna(subset=["Income"])

df_clean.isna().sum()
```

Out[79]:
```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                 0
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
dtype: int64
```

In [80]:
```
#calculating KMeans with k=3#
X=df_clean[["Income","MntWines"]]

from sklearn.cluster import KMeans
k=3
kmeans3=KMeans(n_clusters=k, random_state=42)
y_pred=kmeans3.fit_predict(X)


#3 Clustercenters are found
kmeans3.cluster_centers_
```

```
kmeans_iter1 = KMeans(n_clusters=3, init="random", n_init=1,
                      algorithm="lloyd", max_iter=1, random_state=1)
kmeans_iter2 = KMeans(n_clusters=3, init="random", n_init=1,
                      algorithm="lloyd", max_iter=5, random_state=1)
kmeans_iter3 = KMeans(n_clusters=3, init="random", n_init=1,
                      algorithm="lloyd", max_iter=50, random_state=1)
kmeans_iter1.fit(X)
kmeans_iter2.fit(X)
kmeans_iter3.fit(X)


print("Clusterzentren nach 50 Iterationen:\n", kmeans_iter3.cluster_centers_)


print("\nLabels nach 50 Iterationen:\n", kmeans_iter3.labels_)

#Kmeans grafical overview

import matplotlib.pyplot as plt

def plot_clusters(X, kmeans, title):
    plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=kmeans.labels_, cmap='viridis', s=30, edgecolor='k')
    plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
                marker='X', s=200, c='red', label='Zentren')
    plt.title(title)
    plt.xlabel(X.columns[0])
    plt.ylabel(X.columns[1])
    plt.legend()
    plt.xlim([0,100000])
    plt.grid(True)

plt.figure(figsize=(25,10))

plt.subplot(1, 3, 3)
plot_clusters(X, kmeans_iter3, 'KMeans: 50 Iterationen')

plt.tight_layout()
plt.show()
```

```
Clusterzentren nach 50 Iterationen:
 [[7.95735907e+04 6.34435940e+02]
 [2.91037792e+04 3.44987310e+01]
 [5.44406324e+04 3.23581620e+02]]

Labels nach 50 Iterationen:
 [2 2 0 ... 2 0 2]
```



In [81]:
```
print("\nLabels nach 50 Iterationen:\n", kmeans_iter3.labels_)

#save clusters in dataframe
df_clean['Cluster'] = kmeans_iter3.labels_
```

```
Labels nach 50 Iterationen:
 [2 2 0 ... 2 0 2]
C:\Users\goex1\AppData\Local\Temp\ipykernel_16380\2988419621.py:4: 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#retur
ning-a-view-versus-a-copy
  df_clean['Cluster'] = kmeans_iter3.labels_
```

In [82]:
```
df_clean.head(10)

df_clean.to_csv("df_with_cluster.csv",index=False)
```