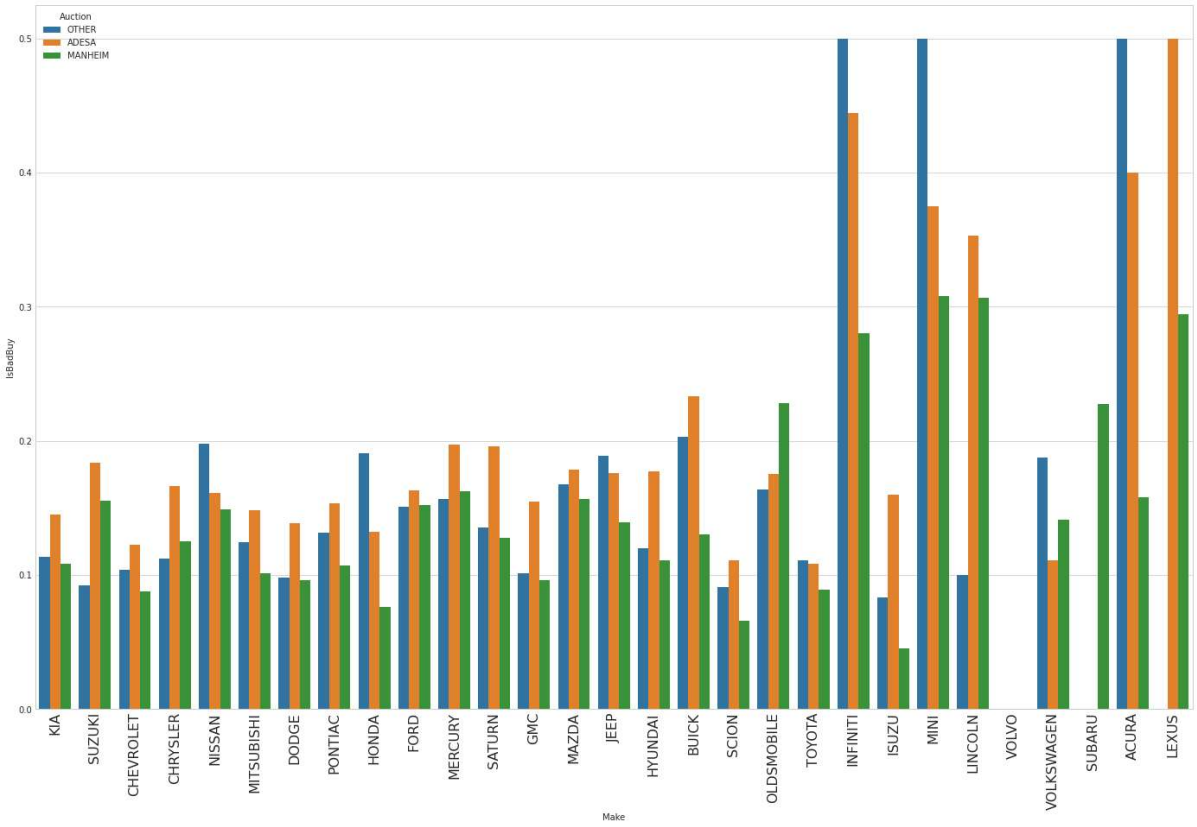
```
plt.show()
```



In [33]:
```
#Outliers Detection with RANSAC

#copy dataframe
df_for_outliers = df.copy()
df_for_outliers = df_for_outliers.dropna()
```

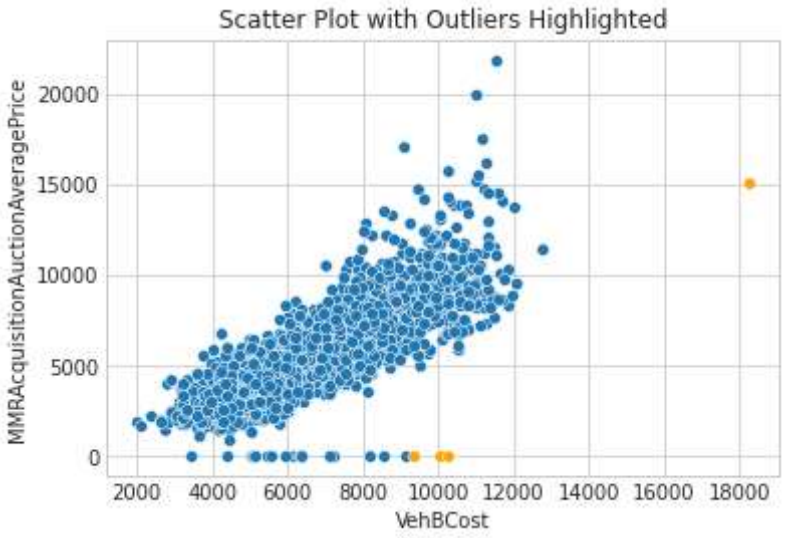
In [34]:
```
print(df['VehOdo'].min(), df['VehOdo'].max()) #problemlos
```

```
5368 115717
```

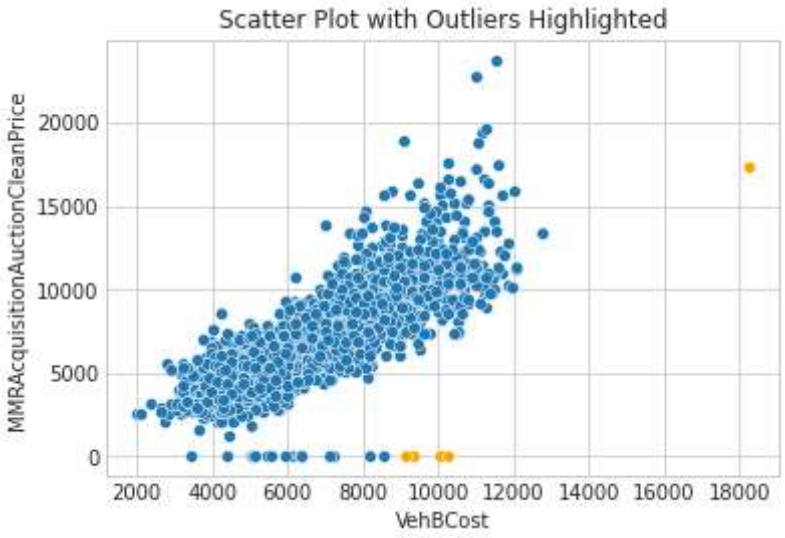
In [35]:
```
#import modules for outlier detection
from sklearn.linear_model import RANSACRegressor
from statsmodels.robust import mad
#RANSAC

numeric_columns = ['MMRAcquisitionAuctionAveragePrice',
                           'MMRAcquisitionAuctionCleanPrice',
                           'MMRAcquisitionRetailAveragePrice',
                           'MMRAcquisitonRetailCleanPrice',
                           'MMRCurrentAuctionAveragePrice',
                           'MMRCurrentAuctionCleanPrice',
                           'MMRCurrentRetailAveragePrice',
                           'MMRCurrentRetailCleanPrice',
                           'VehOdo',
                           'WarrantyCost']


for col in numeric_columns:
    d_distance = mad(df_for_outliers.loc[:, 'VehBCost']) * 3

    model_outlier = RANSACRegressor(residual_threshold=d_distance, random_state=0)
    model_outlier.fit(X=df_for_outliers.loc[:, [col]], y=df_for_outliers.loc[:, 'Ve

    print(col, (~model_outlier.inlier_mask_).sum())
```

```
    #Visualisierung
    ax = sns.scatterplot(x=df_for_outliers.loc[model_outlier.inlier_mask_, 'VehBCos
                         y=df_for_outliers.loc[model_outlier.inlier_mask_, col])

    sns.scatterplot(x=df_for_outliers.loc[~model_outlier.inlier_mask_, 'VehBCost'],
                    y=df_for_outliers.loc[~model_outlier.inlier_mask_, col],
                    color='orange',
                    ax=ax)
    ax.set_xlabel('VehBCost')
    ax.set_ylabel(col)
    ax.set_title('Scatter Plot with Outliers Highlighted')

    # Şimdi grafiği gösterelim
    plt.show()
```

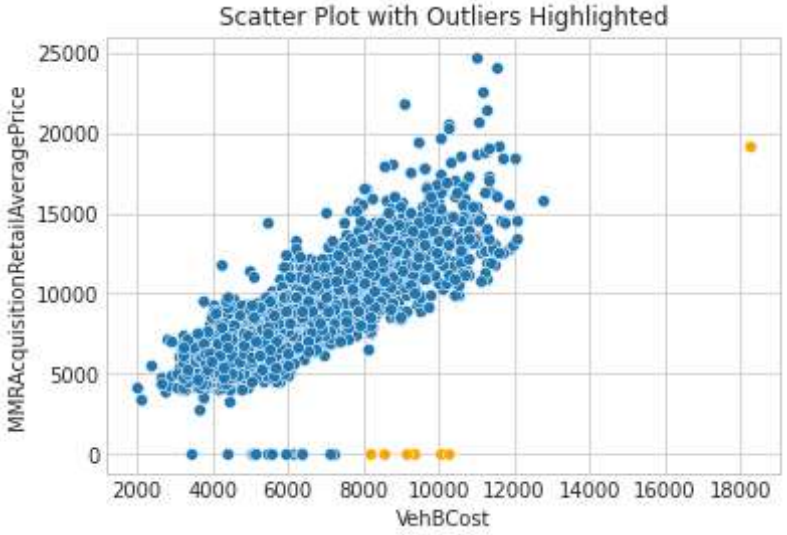
MMRAcquisitionAuctionAveragePrice 4



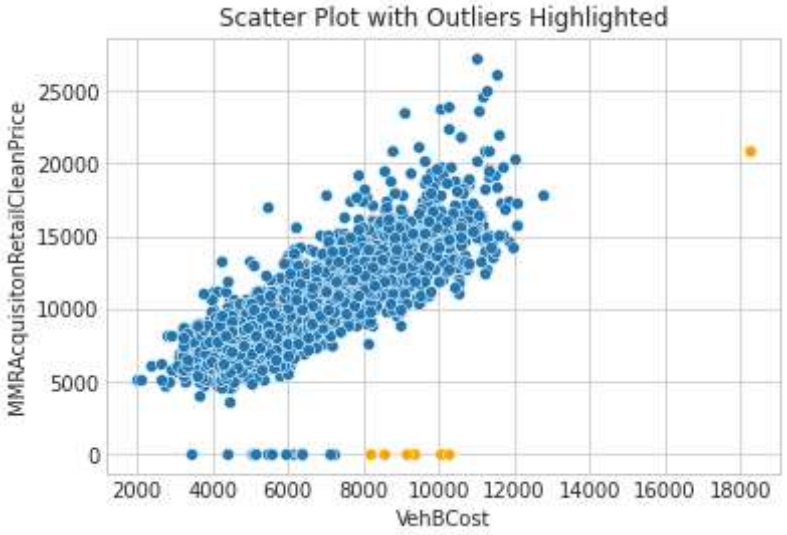
MMRAcquisitionAuctionCleanPrice 5



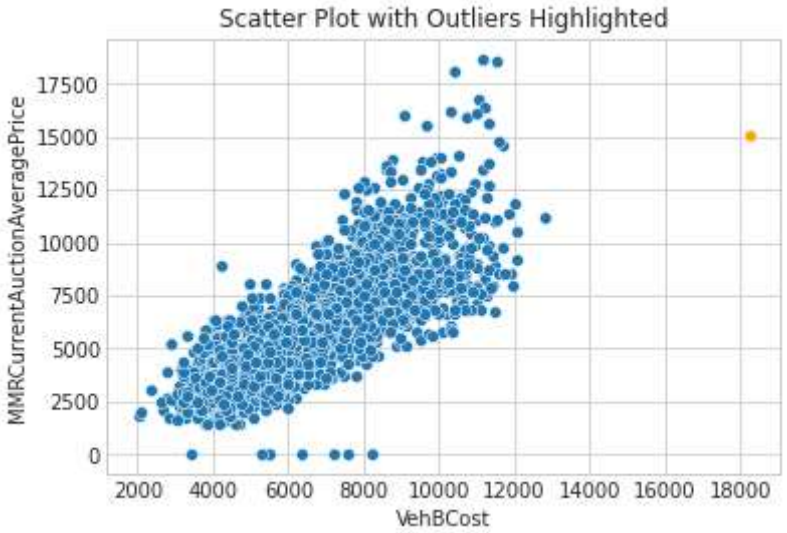
MMRAcquisitionRetailAveragePrice 7



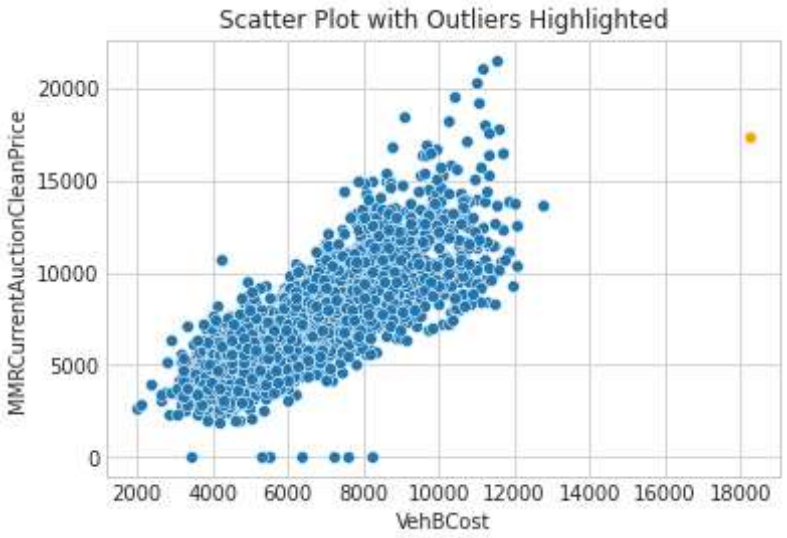
MMRAcquisitonRetailCleanPrice 7



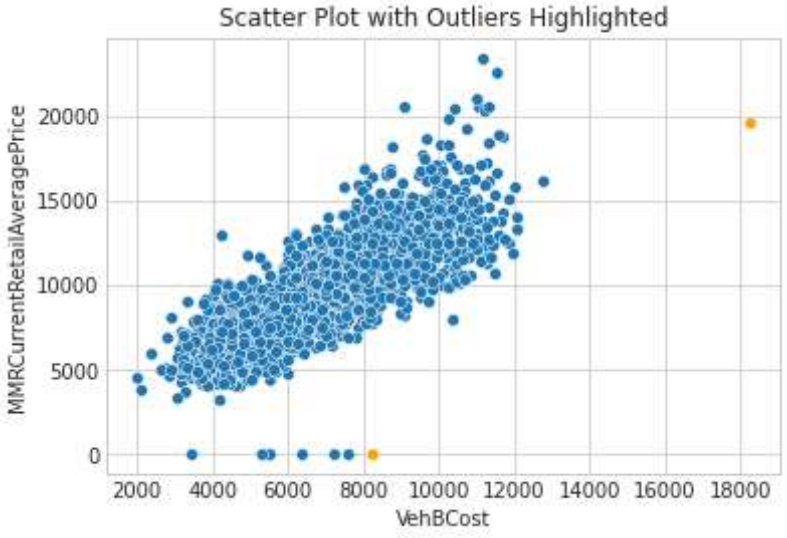
MMRCurrentAuctionAveragePrice 1



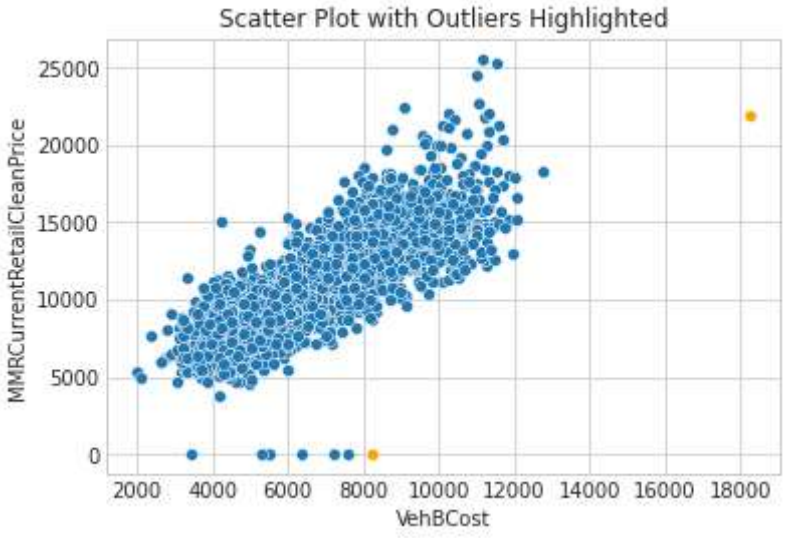
MMRCurrentAuctionCleanPrice 1



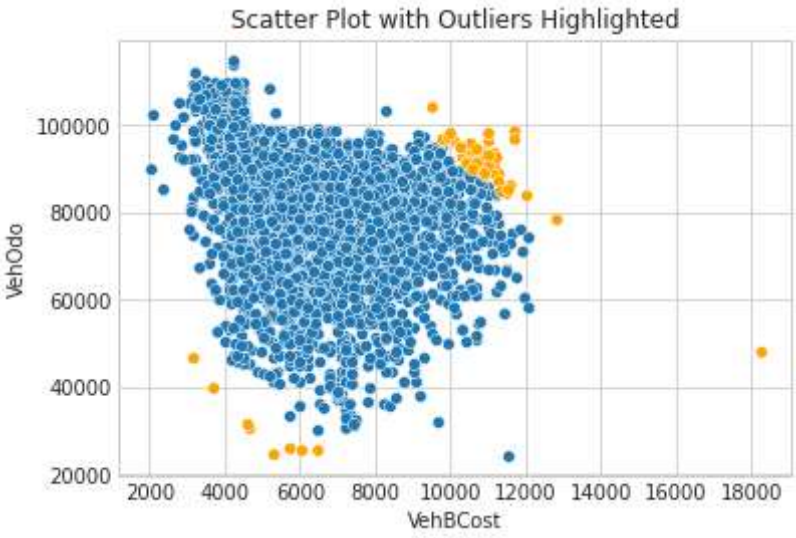
MMRCurrentRetailAveragePrice 2



MMRCurrentRetailCleanPrice 2



VehOdo 53



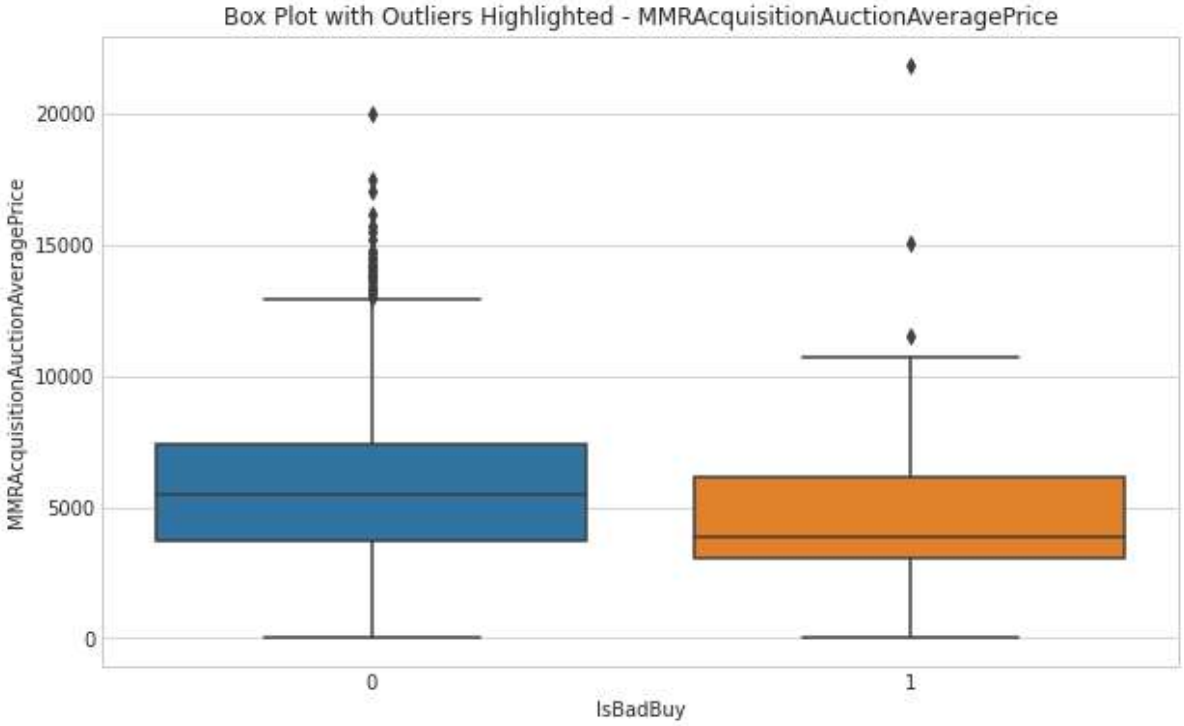
WarrantyCost 136

In [36]:

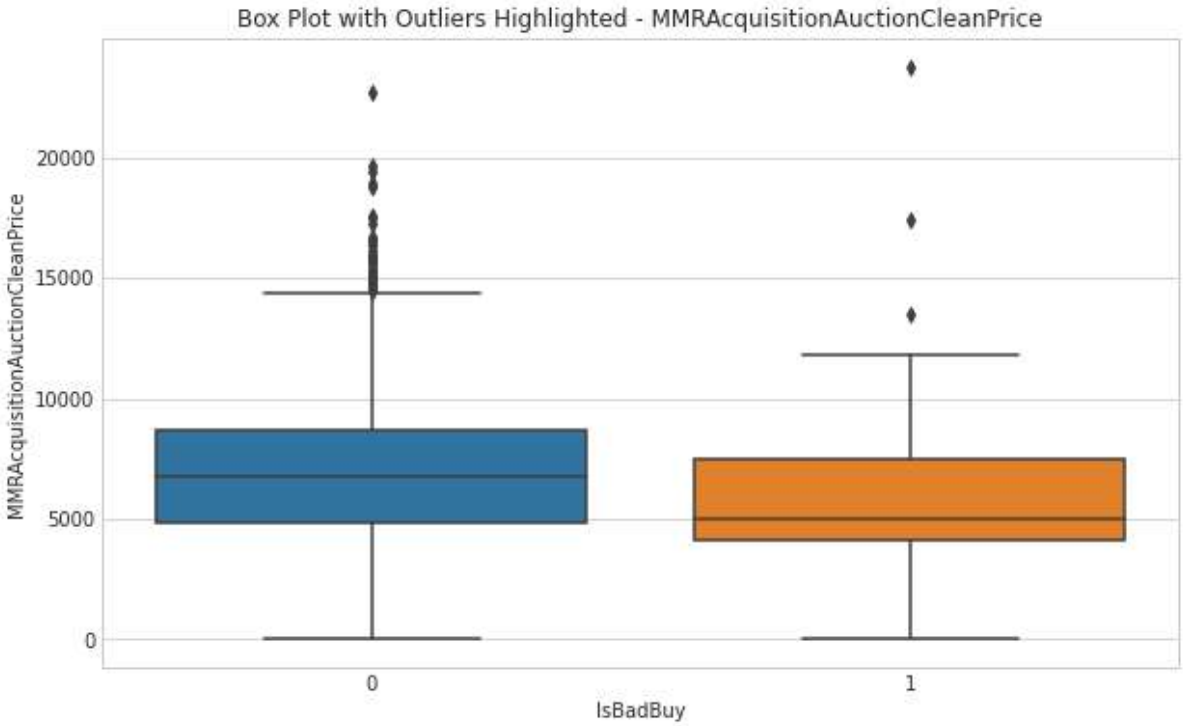
```
import seaborn as sns
import matplotlib.pyplot as plt
numeric_columns = ['MMRAcquisitionAuctionAveragePrice',
                   'MMRAcquisitionAuctionCleanPrice',
                   'MMRAcquisitionRetailAveragePrice',
                   'MMRAcquisitonRetailCleanPrice',
                   'MMRCurrentAuctionAveragePrice',
                   'MMRCurrentAuctionCleanPrice',
                   'MMRCurrentRetailAveragePrice',
                   'MMRCurrentRetailCleanPrice',
                   'VehBCost',
                   'WarrantyCost',
                   'VehOdo']

for col in numeric_columns:
    # IQR hesaplama
    Q1_0 = df_for_outliers.loc[df_for_outliers['IsBadBuy'] == 0, col].quantile(0.25
    Q3_0 = df_for_outliers.loc[df_for_outliers['IsBadBuy'] == 0, col].quantile(0.75
    IQR_0 = Q3_0 - Q1_0

    Q1_1 = df_for_outliers.loc[df_for_outliers['IsBadBuy'] == 1, col].quantile(0.25
    Q3_1 = df_for_outliers.loc[df_for_outliers['IsBadBuy'] == 1, col].quantile(0.75
    IQR_1 = Q3_1 - Q1_1

    # Alt ve üst sınır hesaplama
    lower_limit_0 = Q1_0 - 1.5 * IQR_0
    upper_limit_0 = Q3_0 + 1.5 * IQR_0
```

```
    lower_limit_1 = Q1_1 - 1.5 * IQR_1
    upper_limit_1 = Q3_1 + 1.5 * IQR_1

    # Outlier'ları filtreleme
    outliers_0 = df_for_outliers.loc[(df_for_outliers['IsBadBuy'] == 0) & ((df_for_
    outliers_1 = df_for_outliers.loc[(df_for_outliers['IsBadBuy'] == 1) & ((df_for_

    print(f"{col}: Class 0 - {len(outliers_0)} outliers, Class 1 - {len(outliers_1)

    # Box plot ile Outlier'ları görselleştirme
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='IsBadBuy', y=col, data=df_for_outliers)
    plt.xlabel('IsBadBuy')
    plt.ylabel(col)
    plt.title(f'Box Plot with Outliers Highlighted - {col}')
    plt.show()
```

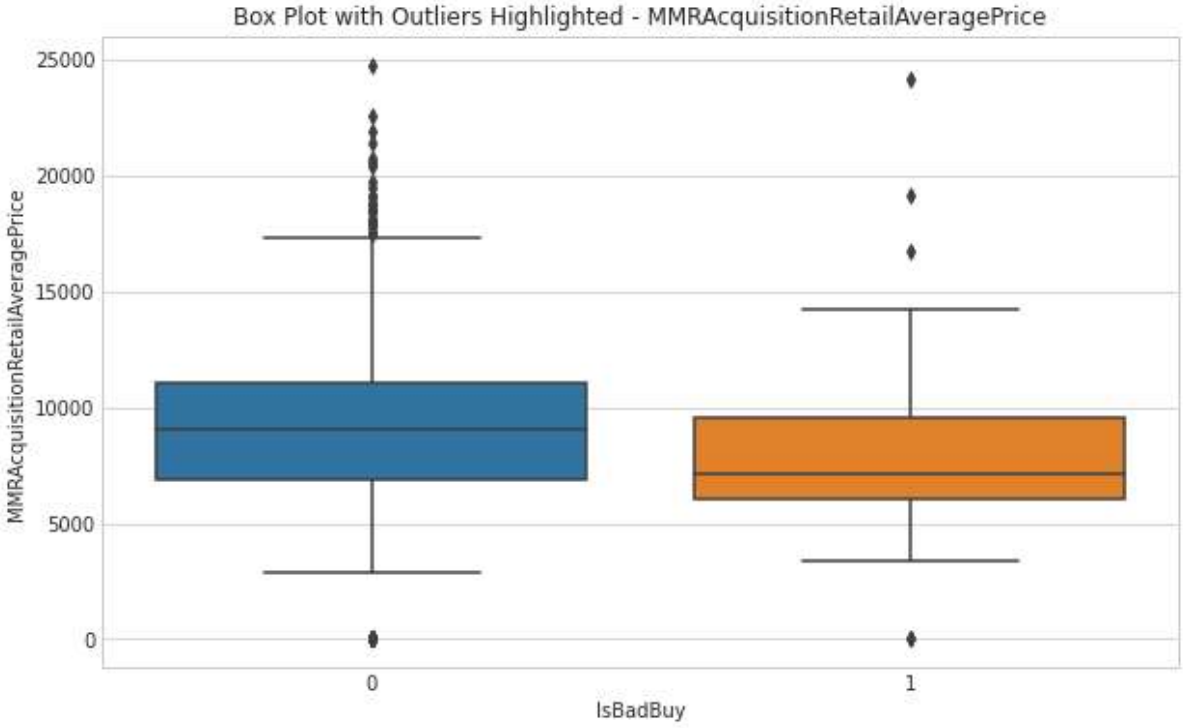
```
MMRAcquisitionAuctionAveragePrice: Class 0 - 25 outliers, Class 1 - 3 outliers
```



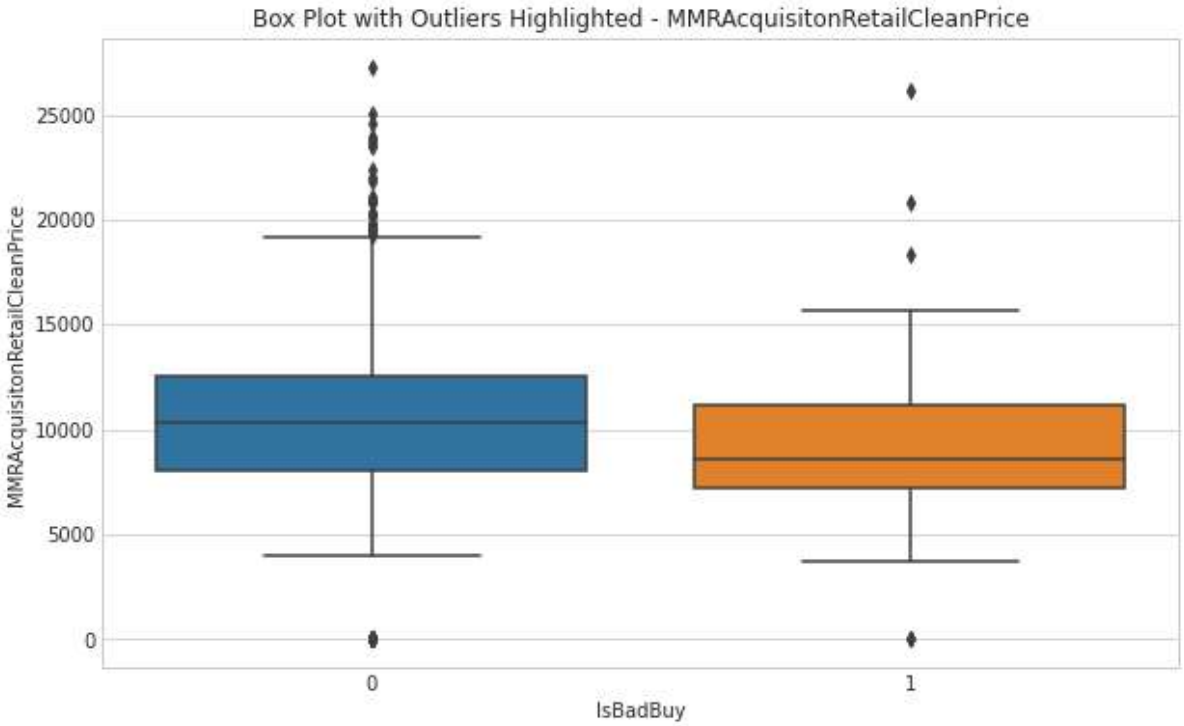

```
MMRAcquisitionAuctionCleanPrice: Class 0 - 35 outliers, Class 1 - 3 outliers
```



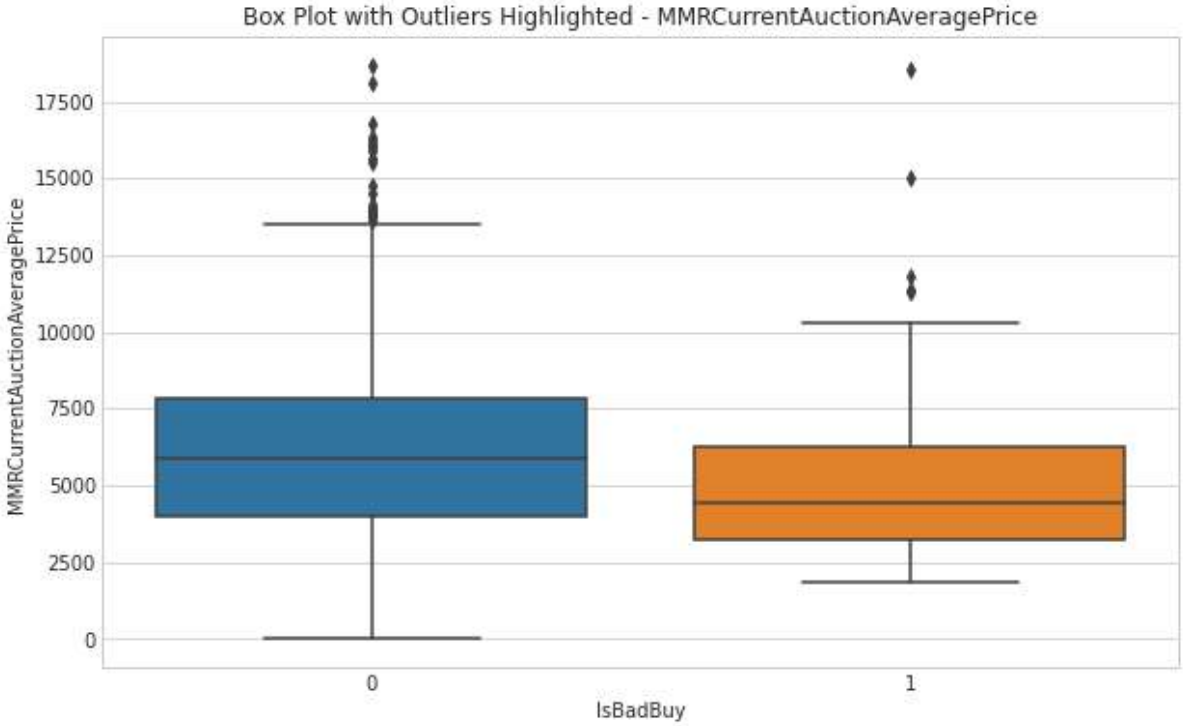
MMRAcquisitionRetailAveragePrice: Class 0 - 42 outliers, Class 1 - 5 outliers



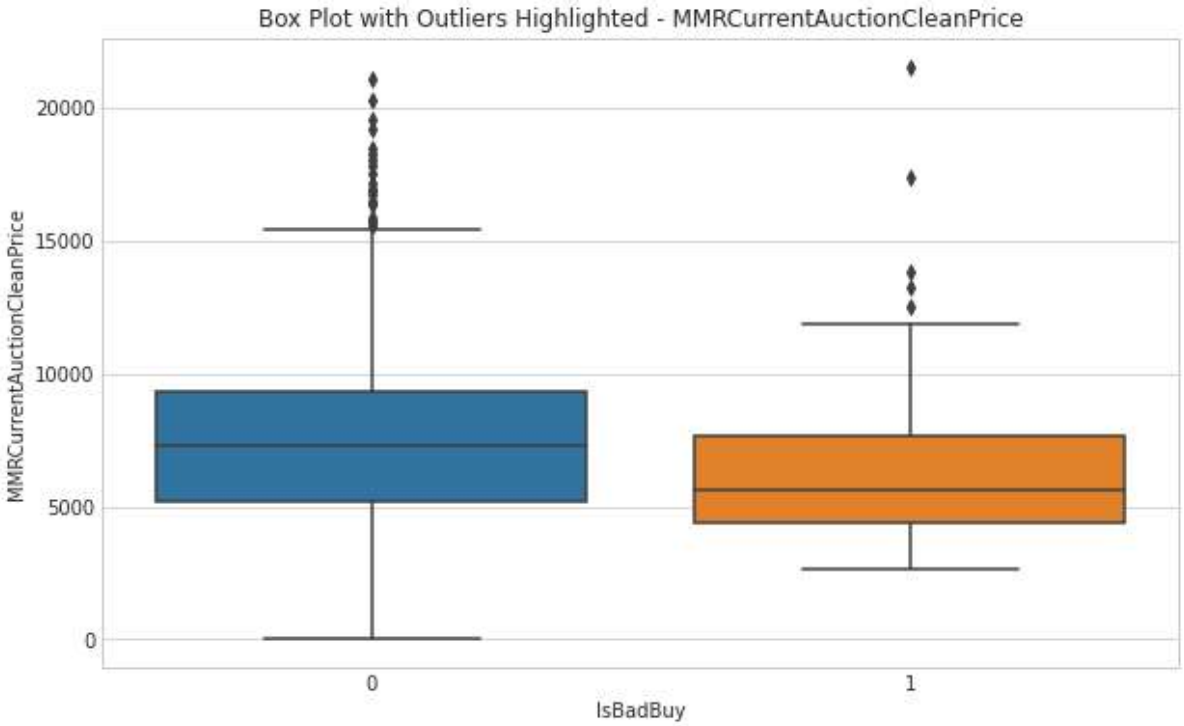
MMRAcquisitonRetailCleanPrice: Class 0 - 46 outliers, Class 1 - 5 outliers



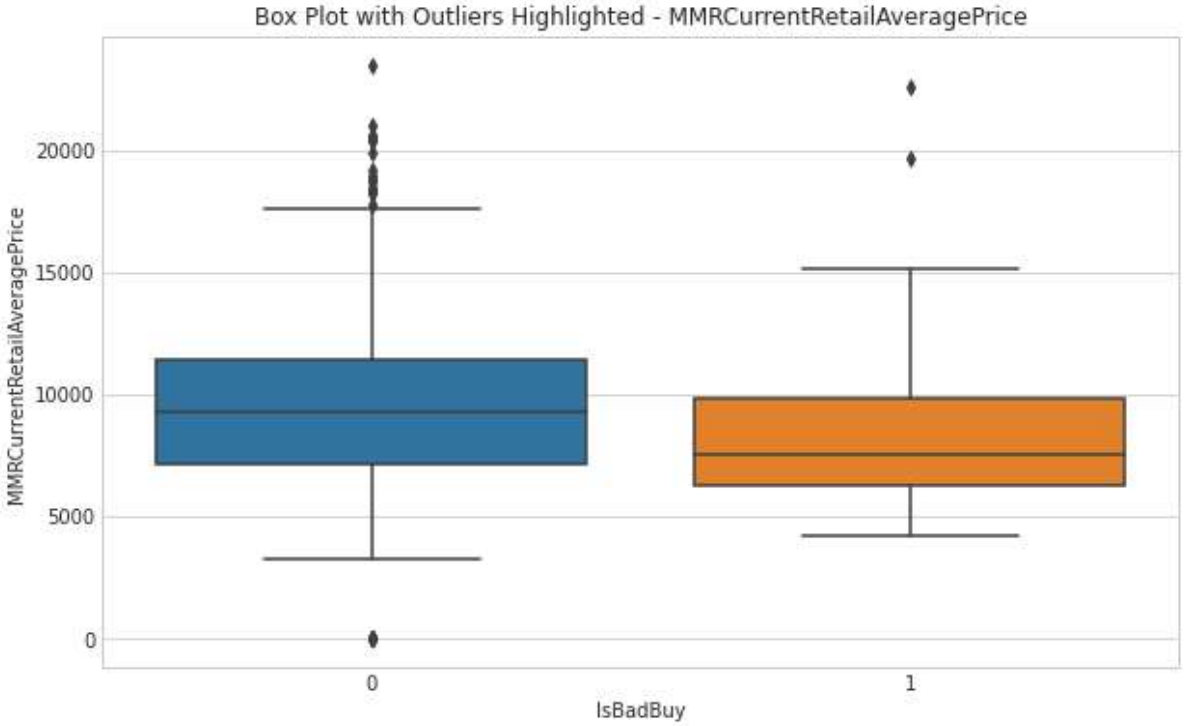
MMRCurrentAuctionAveragePrice: Class 0 - 20 outliers, Class 1 - 5 outliers



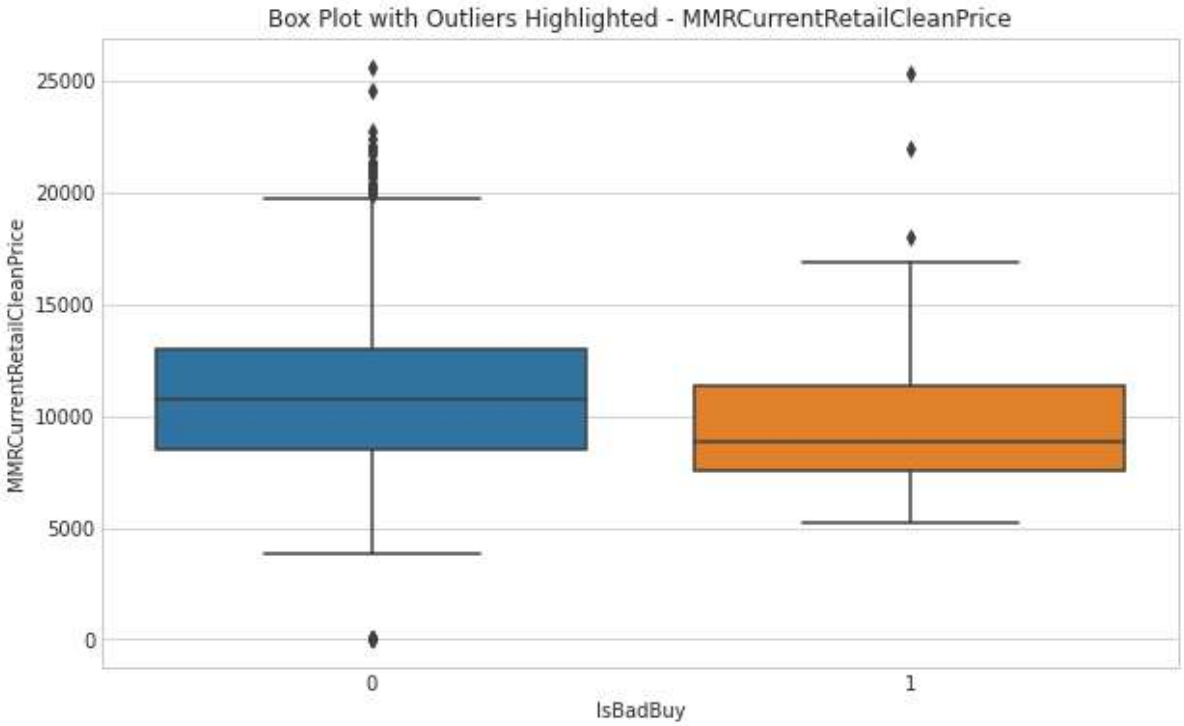
MMRCurrentAuctionCleanPrice: Class 0 - 22 outliers, Class 1 - 5 outliers



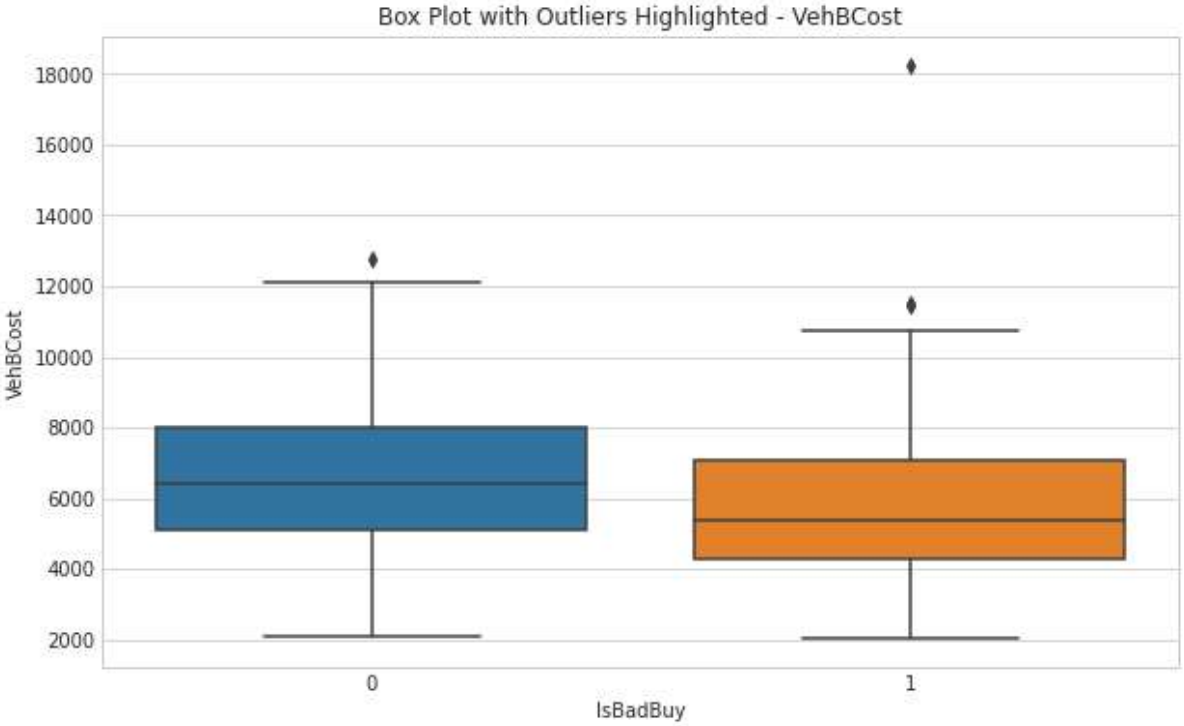
MMRCurrentRetailAveragePrice: Class 0 - 25 outliers, Class 1 - 2 outliers



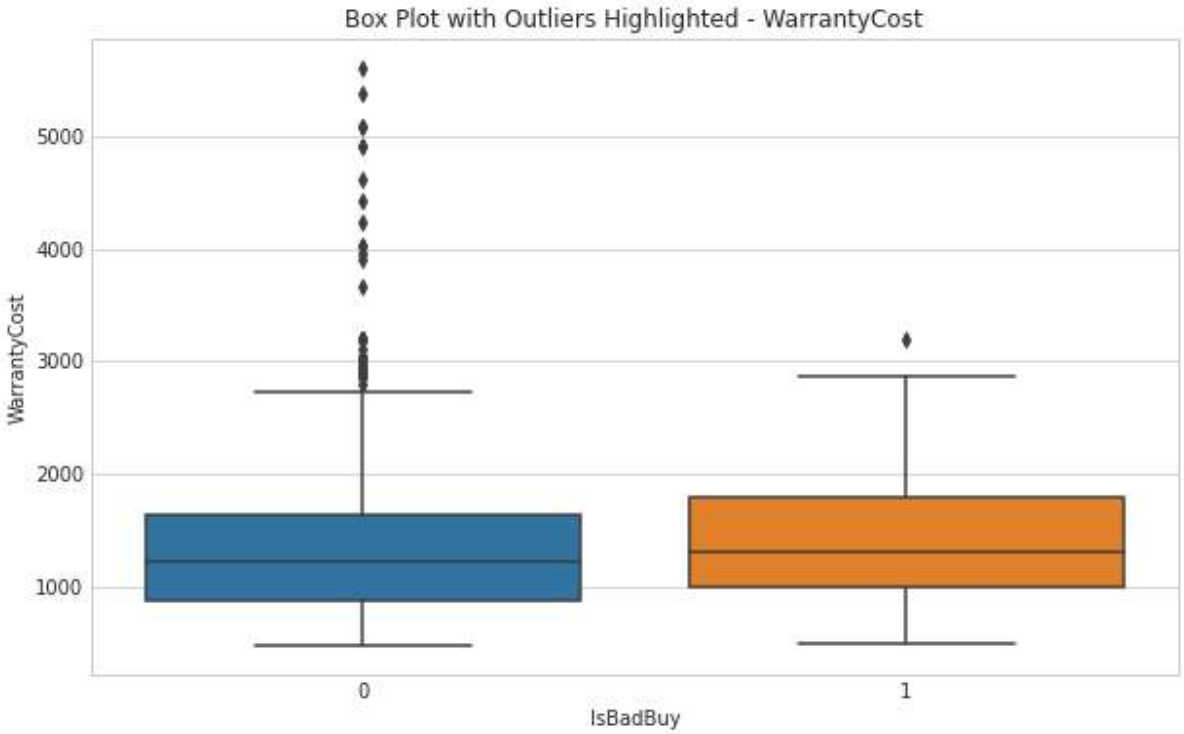
MMRCurrentRetailCleanPrice: Class 0 - 32 outliers, Class 1 - 3 outliers



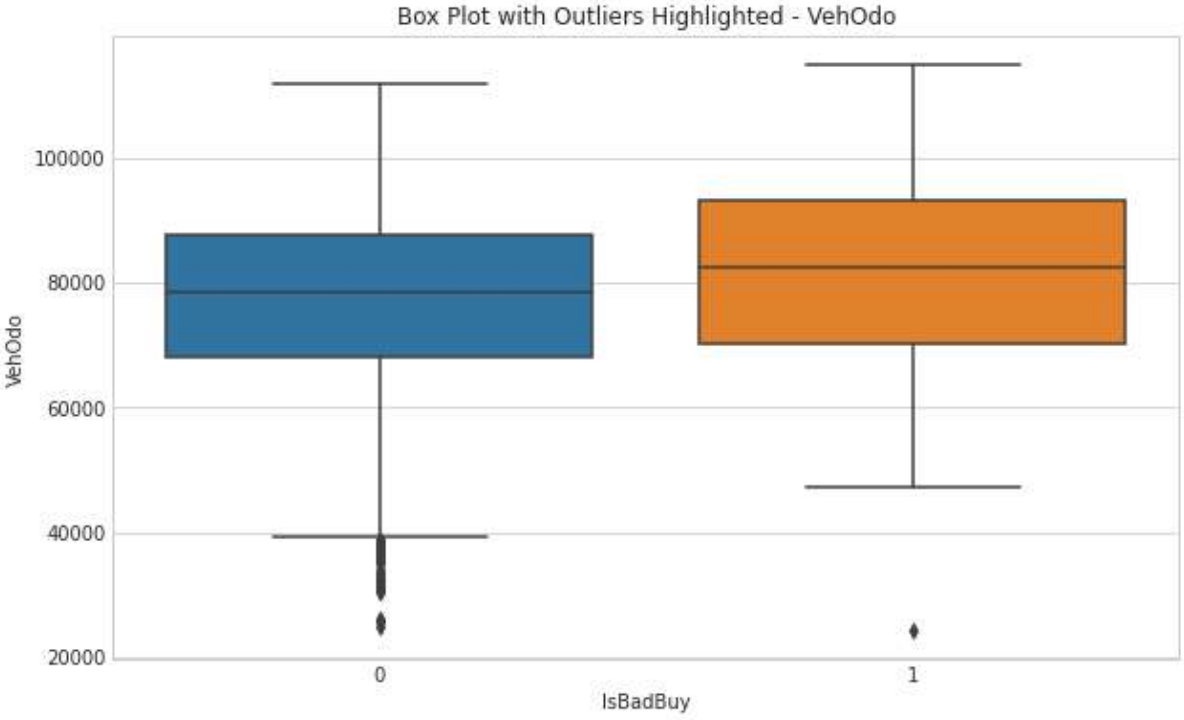
VehBCost: Class 0 - 1 outliers, Class 1 - 3 outliers



WarrantyCost: Class 0 - 43 outliers, Class 1 - 1 outliers



VehOdo: Class 0 - 36 outliers, Class 1 - 1 outliers



In [37]:
```
#chech negativ values
numeric_columns = list(df.select_dtypes(include='number').columns)
for col in numeric_columns:
    mask = df[col]<0
    print(mask.sum())
```