Data Cleaning Exercise

Cleaning your data is crucial when starting a new data engineering project because it ensures the accuracy, consistency, and reliability of the dataset. Dirty data, which may include duplicates, missing values, and errors, can lead to incorrect analysis and insights, ultimately affecting the decision-making process. Data cleaning helps in identifying and rectifying these issues, providing a solid foundation for building effective data models and analytics. Additionally, clean data improves the performance of algorithms and enhances the overall efficiency of the project, leading to more trustworthy and actionable results.

Use Python, `numpy`, `pandas` and/or `matplotlib` to analyse and clean your batch data:

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Load Data

Link to data source: `<TODO>`

In [2]:
```
df = pd.read_csv('AAPL.csv')
```

Understand the Data

View the first few rows, get summary statistics and check data types

In [3]:
```
df.head(10)
```

Out[3]:

	symbol	date	open	high	low	close	volume	
0	AAPL	2024-01-02 00:00:00+00:00	187.149994	188.440002	183.889999	185.639999	82488700	1
1	AAPL	2024-01-03 00:00:00+00:00	184.220001	185.880005	183.429993	184.250000	58414500	1
2	AAPL	2024-01-04 00:00:00+00:00	182.149994	183.089996	180.880005	181.910004	71983600	1
3	AAPL	2024-01-05 00:00:00+00:00	181.990005	182.759995	180.169998	181.179993	62303300	1
4	AAPL	2024-01-08 00:00:00+00:00	182.089996	185.600006	181.500000	185.559998	59144500	1
5	AAPL	2024-01-09 00:00:00+00:00	183.919998	185.149994	182.729996	185.139999	42841800	1
6	AAPL	2024-01-10 00:00:00+00:00	184.350006	186.399994	183.919998	186.190002	46792900	1
7	AAPL	2024-01-11 00:00:00+00:00	186.539993	187.050003	183.619995	185.589996	49128400	1
8	AAPL	2024-01-12 00:00:00+00:00	186.059998	186.740005	185.190002	185.919998	40444700	1
9	AAPL	2024-01-16 00:00:00+00:00	182.160004	184.259995	180.929993	183.630005	65603000	1

In [4]:

```
print(df.dtypes)
```

```
symbol        object
date          object
open         float64
high         float64
low          float64
close        float64
volume         int64
adjclose     float64
dividends    float64
dtype: object
```

In [5]:

```
print("Allgemeine Informationen:")
df.info()

print("\nStatistische Zusammenfassung:")
print(df.describe(include='all'))
```

```
Allgemeine Informationen:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 251 entries, 0 to 250
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   symbol     251 non-null    object 
 1   date       251 non-null    object 
 2   open       251 non-null    float64
 3   high       251 non-null    float64
 4   low        251 non-null    float64
 5   close      251 non-null    float64
 6   volume     251 non-null    int64  
 7   adjclose   251 non-null    float64
 8   dividends  251 non-null    float64
dtypes: float64(6), int64(1), object(2)
memory usage: 17.8+ KB

Statistische Zusammenfassung:
       symbol                       date        open        high         low  \
count     251                        251  251.000000  251.000000  251.000000   
unique      1                        251         NaN         NaN         NaN   
top      AAPL  2024-01-02 00:00:00+00:00         NaN         NaN         NaN   
freq      251                          1         NaN         NaN         NaN   
mean      NaN                        NaN  206.771115  208.733546  205.040279   
std       NaN                        NaN   25.219399   25.477327   25.026478   
min       NaN                        NaN  165.350006  166.399994  164.080002   
25%       NaN                        NaN  183.735001  185.119995  182.180000   
50%       NaN                        NaN  213.929993  216.779999  211.919998   
75%       NaN                        NaN  227.320000  229.375000  225.110001   
max       NaN                        NaN  258.190002  260.100006  257.630005   

             close        volume    adjclose   dividends  
count   251.000000  2.510000e+02  251.000000  251.000000  
unique         NaN           NaN         NaN         NaN  
top            NaN           NaN         NaN         NaN  
freq           NaN           NaN         NaN         NaN  
mean    207.033745  5.719756e+07  206.369270    0.003944  
std      25.406784  3.087430e+07   25.589748    0.031061  
min     165.000000  2.323470e+07  164.224564    0.000000  
25%     184.199997  4.187125e+07  183.217468    0.000000  
50%     214.240005  4.994790e+07  213.522369    0.000000  
75%     227.424995  6.295815e+07  226.839203    0.000000  
max     259.019989  3.186799e+08  258.735504    0.250000  
```

Handle Missing Data

Identify missing values and fill or drop missing values

In [6]:
```python
print("Fehlende Werte pro Spalte:")
print(df.isnull().sum())
```

```
Fehlende Werte pro Spalte:
symbol       0
date         0
open         0
high         0
low          0
close        0
volume       0
adjclose     0
dividends    0
dtype: int64
```

Handle Duplicates

Identify duplicates and remove them

In [7]:
```
duplicates = df.duplicated()
print(f"Anzahl doppelter Zeilen: {duplicates.sum()}")
```

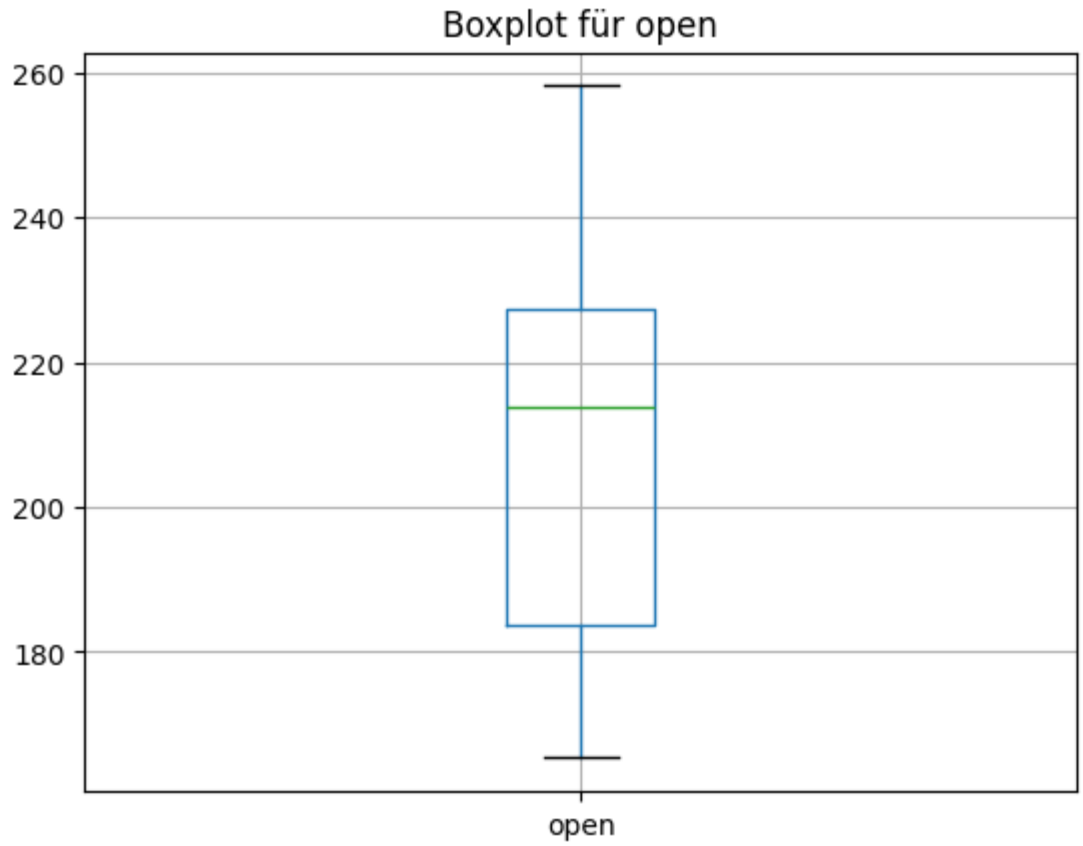
```
Anzahl doppelter Zeilen: 0
```

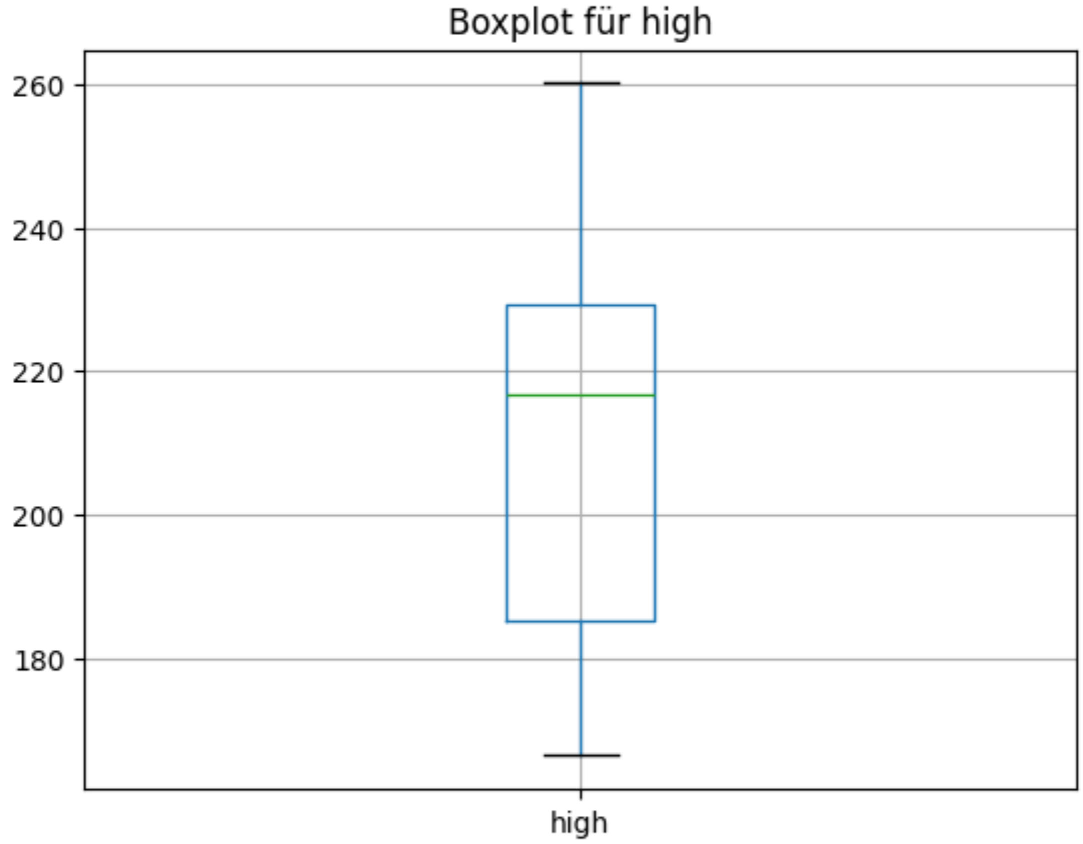
In [8]:
```
# If Duplicates: remove Duplicates
df = df.drop_duplicates()
```

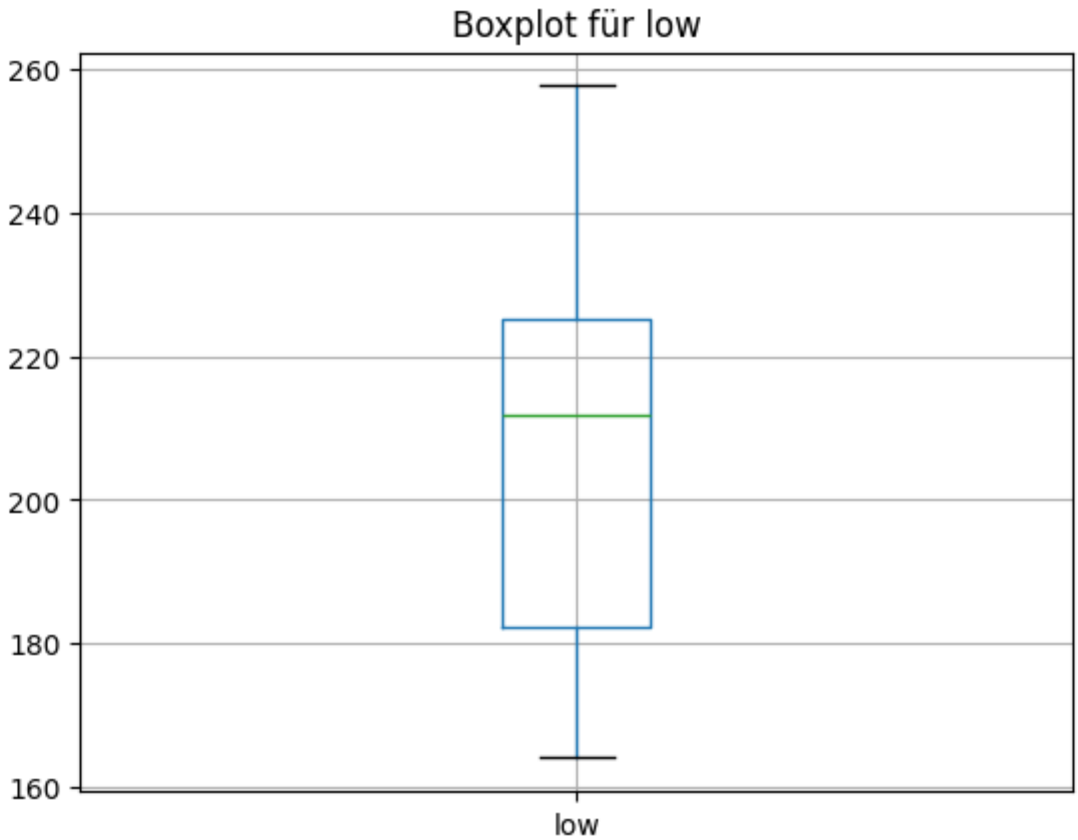
Handle Outliers

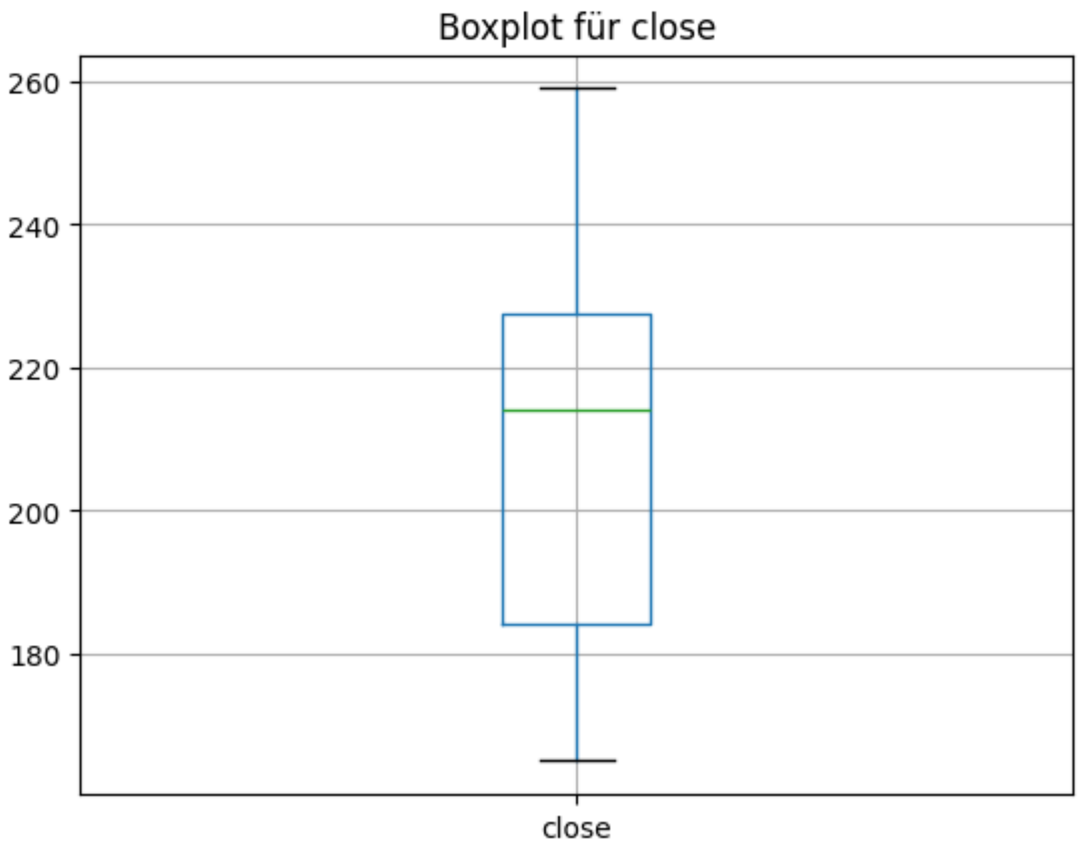
Identify outliers and remove or corret them

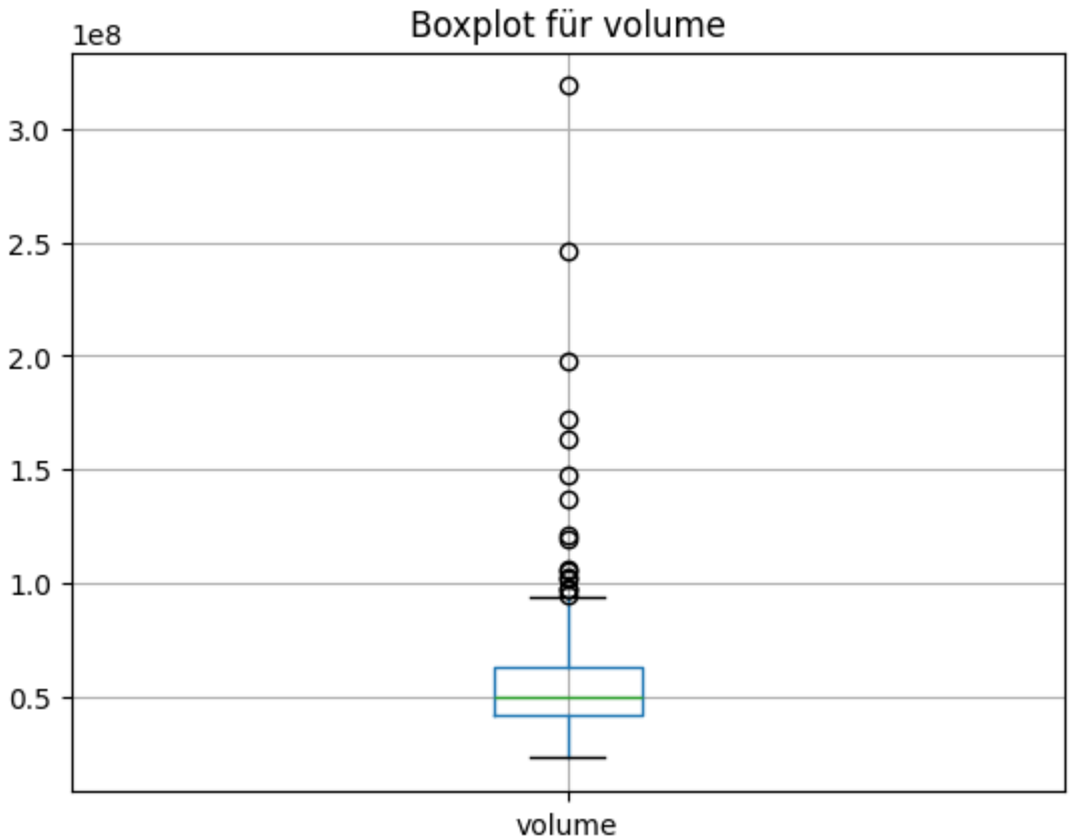
In [9]:
```
numeric_cols = ['open', 'high', 'low', 'close', 'volume', 'adjclose', 'dividends

# Boxplots
for col in numeric_cols:
    plt.figure()
    df.boxplot(column=col)
    plt.title(f'Boxplot für {col}')
    plt.show()
```

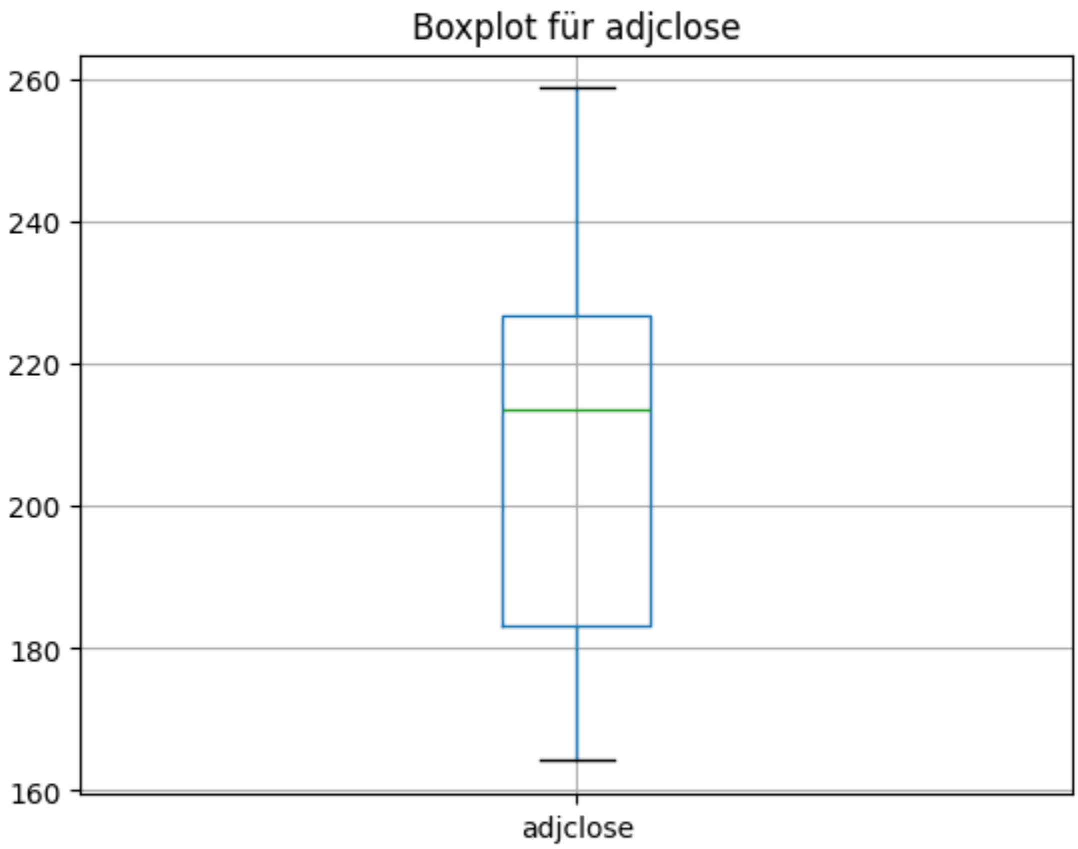

Boxplot für open
260
240
220
200
180
open

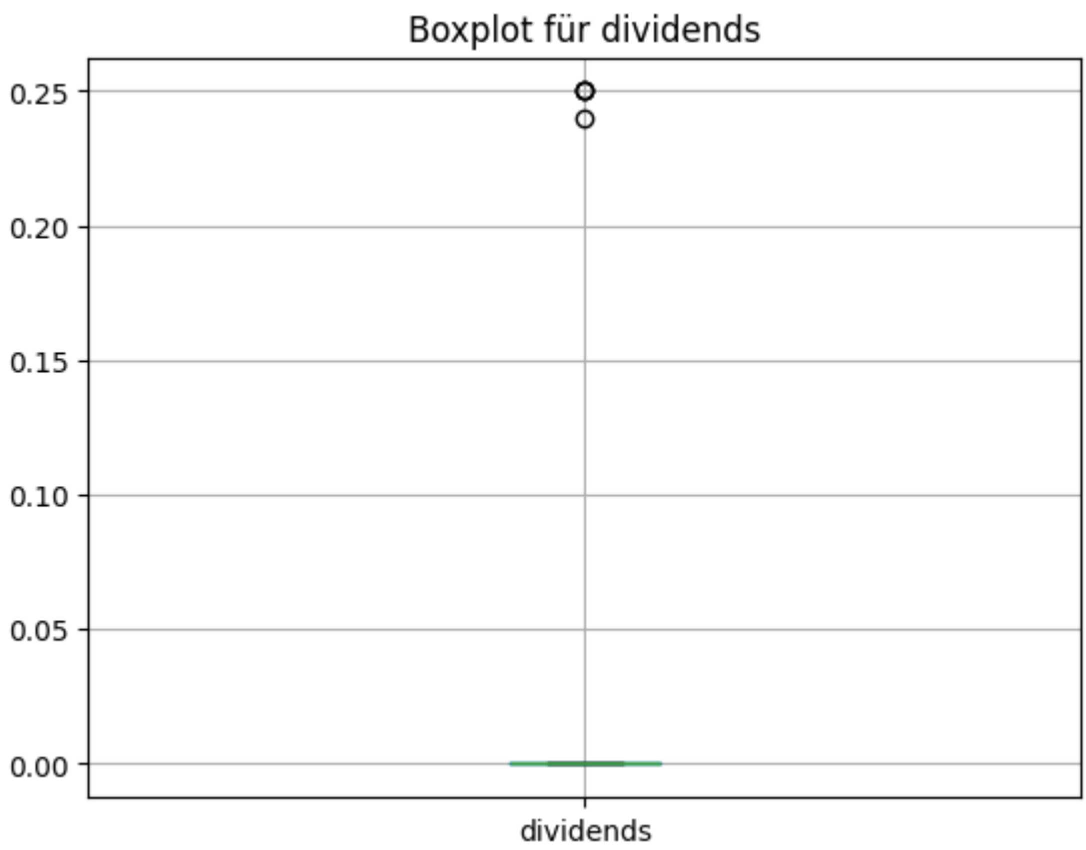

Boxplot für high
260
240
220
200
180
high


Boxplot für low
260
240
220
200
180
160
low


Boxplot für close
260
240
220
200
180
close


Boxplot für volume
1e8
3.0
2.5
2.0
1.5
1.0
0.5
volume


Boxplot für adjclose
260
240
220
200
180
160
adjclose



Handle Incorrect Data Types

In [10]:
```
# Umwandlung der Datumsspalte in datetime-Format
df['date'] = pd.to_datetime(df['date'])
print(df.dtypes)
```

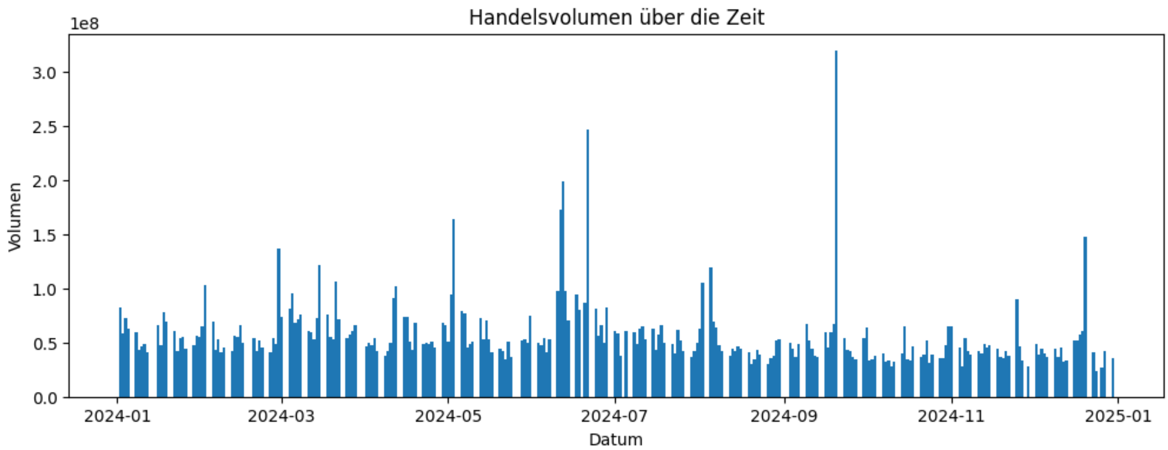
```
symbol                    object
date         datetime64[ns, UTC]
open                     float64
high                     float64
low                      float64
close                    float64
volume                     int64
adjclose                 float64
dividends                float64
dtype: object
```

Visualize Data

Use graphes, plots and/or diagrams to visualize the data

In [11]:
```
plt.figure(figsize=(12, 6))
plt.plot(df['date'], df['close'])
plt.title('AAPL Schlusskurs über die Zeit')
plt.xlabel('Datum')
plt.ylabel('Schlusskurs ($)')
plt.grid(True)
plt.show()
```

In [12]:
```
plt.figure(figsize=(12, 4))
plt.bar(df['date'], df['volume'], width=1.0)
plt.title('Handelsvolumen über die Zeit')
plt.xlabel('Datum')
plt.ylabel('Volumen')
plt.show()
```



Save Cleaned Data

In [13]:
```
df.to_json('AAPL_cleaned.json', orient='records', lines=True)

print("Saved as AAPL_cleaned.json")
```

```
Saved as AAPL_cleaned.json
```