In [20]:
```
import requests
from bs4 import BeautifulSoup
# Step 1: Send an HTTP request to the website
url = 'http://quotes.toscrape.com/'
response = requests.get(url)
```

In [21]:
```
# Check if the request was successful
if response.status_code == 200:
    # Step 2: Parse the HTML content of the page with BeautifulSoup
    soup = BeautifulSoup(response.text, 'html.parser')

    # Step 3: Find all the quotes and authors
    quotes_data = []  # To store the quotes and authors

    # In the website, quotes are inside <span> tags with class 'text' and authors in <small> t
    quotes = soup.find_all('span', class_='text')
    authors = soup.find_all('small', class_='author')

    # Step 4: Loop through the quotes and authors, and store them in a dictionary
    for i in range(len(quotes)):
        quote_text = quotes[i].text
        author = authors[i].text
        quotes_data.append({'quote': quote_text, 'author': author})

    # Step 5: Display the scraped data
    for entry in quotes_data:
        print(f"Quote: {entry['quote']}")
        print(f"Author: {entry['author']}")
        print('---')
else:
    print(f"Failed to retrieve the webpage. Status code: {response.status_code}")
```

```
Quote: “The world as we have created it is a process of our thinking. It cannot be changed wit
hout changing our thinking.”
Author: Albert Einstein
---
Quote: “It is our choices, Harry, that show what we truly are, far more than our abilities.”
Author: J.K. Rowling
---
Quote: “There are only two ways to live your life. One is as though nothing is a miracle. The
other is as though everything is a miracle.”
Author: Albert Einstein
---
Quote: “The person, be it gentleman or lady, who has not pleasure in a good novel, must be int
olerably stupid.”
Author: Jane Austen
---
Quote: “Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous
than absolutely boring.”
Author: Marilyn Monroe
---
Quote: “Try not to become a man of success. Rather become a man of value.”
Author: Albert Einstein
---
Quote: “It is better to be hated for what you are than to be loved for what you are not.”
Author: André Gide
---
Quote: “I have not failed. I've just found 10,000 ways that won't work.”
Author: Thomas A. Edison
---
Quote: “A woman is like a tea bag; you never know how strong it is until it's in hot water.”
Author: Eleanor Roosevelt
---
Quote: “A day without sunshine is like, you know, night.”
Author: Steve Martin
---
```

In [22]:
```python
# Replace with your actual OpenWeatherMap API key
API_KEY ='ff397c60609dbfd1e4df25ab1a6dddae'
city = 'mumbai'  # Adding country code to avoid ambiguity
# Define the API endpoint and include your API key
url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_KEY}&units=metric"
# Send a GET request to the API
response = requests.get(url)

# Check if the request was successful
if response.status_code == 200:
    data = response.json()
    print(f"City: {data['name']}")
    print(f"Temperature: {data['main']['temp']}°C")
    print(f"Weather: {data['weather'][0]['description']}")
else:
    print(f"Failed to retrieve data. Status code: {response.status_code}, Reason: {response.r
```

```
City: Mumbai
Temperature: 25.99°C
Weather: mist
```

In [ ]:
```python
#practical 4
```

In [23]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

In [25]:
```
# Step 1: Load the Titanic Dataset
titanic = sns.load_dataset('titanic')
# Step 2: Inspect the Dataset
print("First 5 rows of the Titanic dataset:")
print(titanic.head())
```

```
First 5 rows of the Titanic dataset:
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

In [26]:
```
# Check for missing values
print("\nMissing values in each column:")
print(titanic.isnull().sum())
```

```
Missing values in each column:
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [28]:
```
# Check data types
print("\nData types and basic statistics:")
print(titanic.info())
```

```
Data types and basic statistics:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
None
```

In [30]:
```python
# Check stastical summary
print("\n statistical Summary:")
print(titanic.describe())
# Check shape of dataset
print("\n Shape of dataset:")
print(titanic.shape)
```

```
 statistical Summary:
        survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200

 Shape of dataset:
(891, 15)
```

In [32]:
```python
# Step 3: Handle Missing Values
# Fill missing 'Age' values with the median of the 'Age' column
#titanic['age'].fillna(titanic['age'].median(),inplace=True)
#titanic['age'] = titanic['age'].fillna(titanic['age'].median())
titanic.fillna({'age': titanic['age'].median()}, inplace=True)
```

In [33]:
```python
# Fill missing 'embarked' values with the most common value (mode)
#titanic['embarked'].fillna(titanic['embarked'].mode()[0], inplace=True)
titanic['embarked'] = titanic['embarked'].fillna(titanic['embarked'].mode()[0])
```

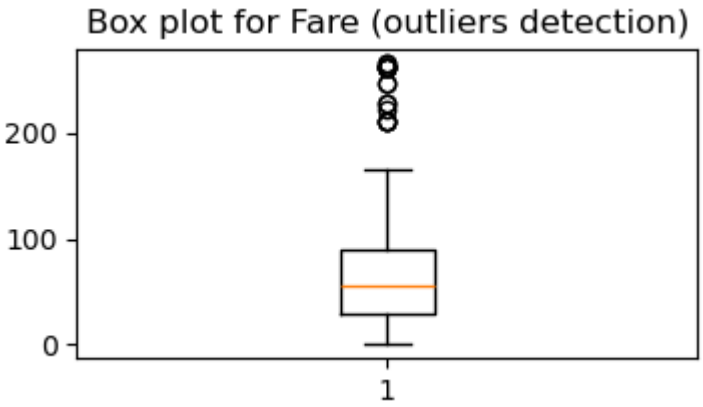
In [34]:
```python
# Drop any remaining rows with missing values
titanic.dropna(inplace=True)
print("\nMissing values after cleaning:")
print(titanic.isnull().sum())
```

```
Missing values after cleaning:
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
deck           0
embark_town    0
alive          0
alone          0
dtype: int64
```

In [35]:
```python
# Step 4: Handle Categorical Variables
# Convert 'sex' and 'embarked' into numerical labels using LabelEncoder
label_encoder = LabelEncoder()
titanic['sex'] = label_encoder.fit_transform(titanic['sex'])
titanic['embarked'] = label_encoder.fit_transform(titanic['embarked'])
# Convert 'who' into binary (man: 1, woman: 0)
titanic['who'] = titanic['who'].apply(lambda x: 1 if x == 'man' else 0)
```

In [36]:
```python
# Step 5: Feature Engineering
# Create a new feature: 'family_size' = 'sibsp' + 'parch' + 1
titanic['family_size'] = titanic['sibsp'] + titanic['parch'] + 1
# Step 6: Remove Duplicates
titanic_cleaned = titanic.drop_duplicates()
```

In [39]:
```python
# Step 7: Handle Outliers
# Inspecting outliers in the 'fare' column
plt.figure(figsize=(4,2))
plt.boxplot(titanic_cleaned['fare'])
plt.title("Box plot for Fare (outliers detection)")
plt.show()
```




In [48]:
```python
# Cap outliers in 'fare' to the 99th percentile
fare_cap = titanic_cleaned['fare'].quantile(0.99)
titanic_cleaned['fare'] = np.where(titanic_cleaned['fare'] > fare_cap, fare_cap, titanic_clea

print("\nFare column statistics after handling outliers:")
print(titanic_cleaned['fare'].describe())
```

```
Fare column statistics after handling outliers:
count    2.000000e+02
mean    -3.917534e-08
std      1.002509e+00
min     -1.160423e+00
25%     -7.100730e-01
50%     -2.961599e-01
75%      2.538254e-01
max      2.972324e+00
Name: fare, dtype: float64
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_2812\423033087.py:3: 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
  titanic_cleaned['fare'] = np.where(titanic_cleaned['fare'] > fare_cap, fare_cap, titanic_cleaned['fare'])
```

In [49]:
```
# Step 8: Normalize Numerical Data
# Normalize 'age' and 'fare' using StandardScaler
scaler = StandardScaler()
titanic_cleaned[['age', 'fare']] = scaler.fit_transform(titanic_cleaned[['age', 'fare']])
print("\nFirst 5 rows of the cleaned and wrangled data:")
print(titanic_cleaned.head())
```

```
First 5 rows of the cleaned and wrangled data:
    survived  pclass  sex       age  sibsp  parch      fare  embarked  class  \
1          1       1    0  0.202112      1      0 -0.040286         0  First
3          1       1    0  0.002849      1      0 -0.326016         2  First
6          0       1    1  1.264844      0      0 -0.345462         2  First
10         1       3    0 -2.056194      1      1 -0.898001         2  Third
11         1       1    0  1.530527      0      0 -0.743219         2  First

    who  adult_male deck  embark_town alive  alone  family_size
1     0       False    C    Cherbourg   yes  False            2
3     0       False    C  Southampton   yes  False            2
6     1        True    E  Southampton    no   True            1
10    0       False    G  Southampton   yes  False            3
11    0       False    C  Southampton   yes   True            1
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_2812\142412215.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#returning-a-view-versus-a-copy
  titanic_cleaned[['age', 'fare']] = scaler.fit_transform(titanic_cleaned[['age', 'fare']])
```

In [50]:
```
# Step 9: Save the cleaned data to a CSV file
titanic_cleaned.to_csv('titanic_cleaned.csv', index=False)
print("\nCleaned dataset saved to 'titanic_cleaned.csv'")
```

```
Cleaned dataset saved to 'titanic_cleaned.csv'
```

In [ ]: