Preparation of data

In [1]:
```
# Import neccessary libraries
import pandas as pd
import zipfile
import io
import datetime
import requests
import matplotlib.pyplot as plt

# Download the ZIP file with the data from the European Central Bank
url = 'https://www.ecb.europa.eu/stats/eurofxref/eurofxref-hist.zip'
response = requests.get(url)

# Extract the CSV from the ZIP and load it correctly
with zipfile.ZipFile(io.BytesIO(response.content)) as z:
    with z.open('eurofxref-hist.csv') as f:
        # Read with correct delimiter (comma)
        df = pd.read_csv(f)

# Rename first column to 'Date' to make sure rhere is no error occurring
df.rename(columns={df.columns[0]: 'Date'}, inplace=True)

# Convert Date to datetime and set as index
df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d')
df.set_index('Date', inplace=True)

# Keep only the desired currencies: USD, AUD, CHF and CNY
selected_currencies = ['USD', 'AUD', 'CHF', 'CNY']
df = df[selected_currencies]

# Sort by date
df = df.sort_index()

# Show last few rows to confirm it worked
print("The euro exchange reference rates are as follows:")
print(df.tail())
```

```
The euro exchange reference rates are as follows:
              USD     AUD     CHF     CNY
Date
2025-05-08  1.1297  1.7605  0.9325  8.1764
2025-05-09  1.1252  1.7572  0.9353  8.1470
2025-05-12  1.1106  1.7384  0.9369  7.9999
2025-05-13  1.1112  1.7341  0.9358  8.0021
2025-05-14  1.1214  1.7340  0.9390  8.0794
```

In [2]:
```
# Drop columns with NaN values and modify DataFrame
df.dropna(axis=0, inplace=True)
df
```

Out[2]:

Date	USD	AUD	CHF	CNY
2005-04-01	1.2959	1.6803	1.5527	10.7255
2005-04-04	1.2883	1.6777	1.5535	10.6626
2005-04-05	1.2810	1.6758	1.5541	10.6022
2005-04-06	1.2860	1.6804	1.5508	10.6436
2005-04-07	1.2923	1.6798	1.5497	10.6957
...	...	...	...	...
2025-05-08	1.1297	1.7605	0.9325	8.1764
2025-05-09	1.1252	1.7572	0.9353	8.1470
2025-05-12	1.1106	1.7384	0.9369	7.9999
2025-05-13	1.1112	1.7341	0.9358	8.0021
2025-05-14	1.1214	1.7340	0.9390	8.0794

5151 rows × 4 columns

Goal 1: Simple currency converter: Input a quantity in CHF, USD, CNY or AUD and convert to Euro or vice versa for the date of yesterday

In [4]:
```
# Get yesterday's date
today = datetime.date.today()
yesterday = today - datetime.timedelta(days=1)

# Convert to pandas Timestamp to match DataFrame index
yesterday = pd.Timestamp(yesterday)
```

```
# Find the latest available date in the DataFrame that is ≤ yesterday
conversion_date = df.index[df.index <= yesterday].max()

# Extract the exchange rates for that specific day (row from the DataFrame)
rate_on_date = df.loc[conversion_date]  # This gives you a Series with currency values on that day



# Define a simple currency converter function
def convert_currency(amount, from_currency, to_currency='EUR'):

    # Convert TO Euro (from another currency)
    if to_currency == 'EUR':
        rate = rate_on_date[from_currency]  # Get the exchange rate from currency to EUR
        return round(amount / rate, 2)      # Divide amount by rate to get EUR

    # Convert FROM Euro (to another currency)
    elif from_currency == 'EUR':
        rate = df.loc[conversion_date][to_currency]   # Get rate from EUR to target currency
        return round(amount * rate, 2)                # Multiply amount by rate to convert

    # If trying to convert between non-EUR currencies, raise an error
    else:
        raise ValueError("Conversion only supports to/from Euro.")


# Define valid currencies
valid_currencies = ['USD', 'CHF', 'CNY', 'AUD', 'EUR']

# Ask the user for input. Try again if an error occurs
while True:
    try:
        print(f"Currency converter: Please insert your conversion request:")
        from_currency = input("Enter the currency you are converting from (USD, CHF, CNY, AUD, EUR): ").upper()
        amount = float(input("Enter the amount to convert: "))
        to_currency = input("Enter the currency you want to convert to (USD, CHF, CNY, AUD, EUR): ").upper()

        # Check, if currencies are valid
        if from_currency not in valid_currencies or to_currency not in valid_currencies:
            print("Error: Invalid currency input. Please try again.\n")
            continue

        # Check if conversion is valid (must involve EUR and not be EUR to EUR)
        if (from_currency != 'EUR' and to_currency != 'EUR') or (from_currency == 'EUR' and to_currency == 'EUR
            print("Error: One of the currencies must be EUR, and conversion from EUR to EUR is not allowed. Ple
            continue

        # Perform the conversion
        result = convert_currency(amount, from_currency, to_currency)
        print(f"The exchange rate according to your input was as the following:")
        print(f"{amount} {from_currency} was equal to {result} {to_currency} on {conversion_date}.")

         # Successful conversion → exit loop
        break

    except ValueError as e:
        print("There was an error:", e)
        print("Please try again.\n")
```

```
Currency converter: Please insert your conversion request:
Enter the currency you are converting from (USD, CHF, CNY, AUD, EUR): AUD
Enter the amount to convert: 9378
Enter the currency you want to convert to (USD, CHF, CNY, AUD, EUR): EUR
The exchange rate according to your input was as the following:
9378.0 AUD was equal to 5408.3 EUR on 2025-05-14 00:00:00.
```

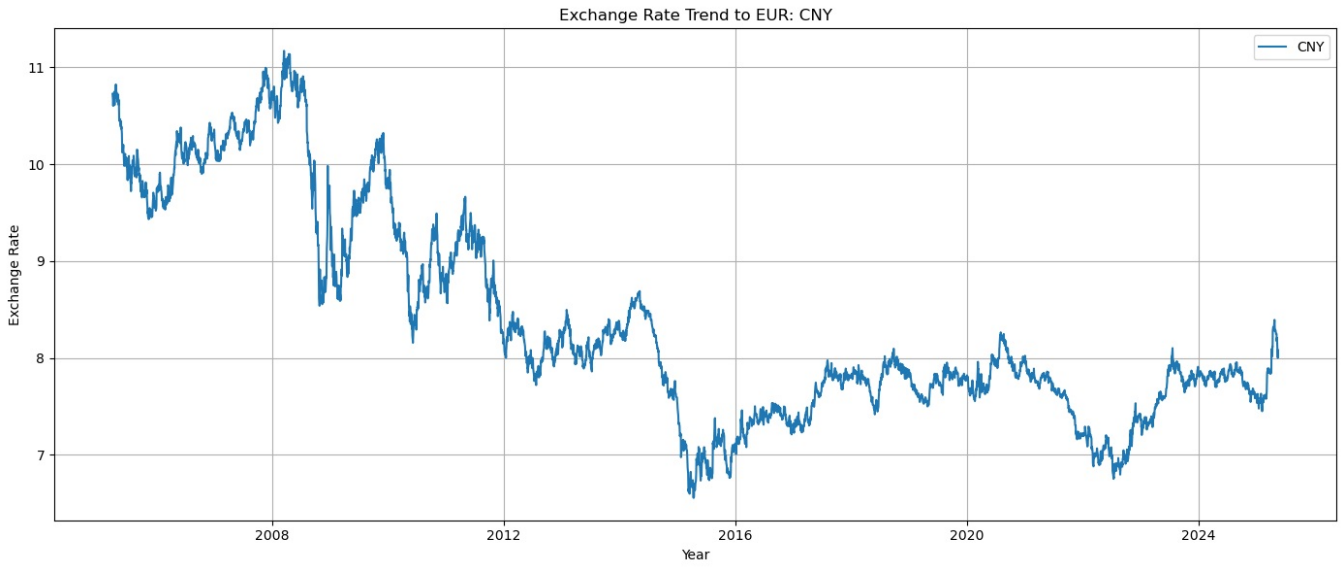
Goal 2: Historical exchange rate analysis: Line chart of how individual exchange rates have developed over the years and displaying KPIs such as maximum, minimum, mean value and volatility (range of fluctuation) of each exchange rate

In [5]:

```
# Define valid currencies
valid_currencies = ['USD', 'CHF', 'CNY', 'AUD']


while True:
    # Ask the user which currency to display
    selected_currency = input("Which currency would you like to display (USD, CHF, CNY, AUD)? ").upper()

    # Valid input:
    if selected_currency in valid_currencies:
        break

    # Error if input is not in line with the valid currencies
    else:
        print("Invalid input. Please enter one of: USD, CHF, CNY, AUD.")

# Plot the selected currency
plt.figure(figsize=(14, 6))
plt.plot(df.index, df[selected_currency], label=selected_currency)
```

```
# Modify plot
plt.title(f'Exchange Rate Trend to EUR: {selected_currency}')
plt.xlabel('Year')
plt.ylabel('Exchange Rate')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Which currency would you like to display (USD, CHF, CNY, AUD)? CNY
```



In [6]:

```
# List to collect KPI rows for each currency
kpi_rows = []

# Loop through each currency to calculate KPIs
for currency in ['USD', 'CHF', 'CNY', 'AUD']:
    series = df[currency]

    # Basic statistics
    max_val = series.max()
    min_val = series.min()
    mean_val = series.mean()
    volatility = max_val - min_val

    # Dates for max and min
    max_date = series.idxmax()
    min_date = series.idxmin()

    # Append the row as a dictionary
    kpi_rows.append({
        'Currency': currency,
        'Max': round(max_val, 4),
        'Max Date': max_date.strftime('%Y-%m-%d'),
        'Min': round(min_val, 4),
        'Min Date': min_date.strftime('%Y-%m-%d'),
        'Mean': round(mean_val, 4),
        'Volatility': round(volatility, 4),
    })

# Create the final KPI DataFrame from the list of rows
kpi_df = pd.DataFrame(kpi_rows)

# Display the result
print("Here is an overview of the most important KPIs")
kpi_df
```

```
Here is an overview of the most important KPIs
```

Out[6]:

	Currency	Max	Max Date	Min	Min Date	Mean	Volatility
0	USD	1.5990	2008-07-15	0.9565	2022-09-28	1.2254	0.6425
1	CHF	1.6803	2007-10-12	0.9242	2025-04-15	1.2281	0.7561
2	CNY	11.1699	2008-03-17	6.5552	2015-04-13	8.3759	4.6147
3	AUD	2.0735	2008-12-18	1.1639	2012-08-09	1.5498	0.9096