Импорт библиотек

In [2]:
```
import pandas as pd
import numpy as np
import investpy  # библиотека для получения финансовых данных
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
from itertools import combinations
import datetime
```

Получение данных с investing.com

В данном примере мы получим данные для нескольких акций. (Параметры функции investpy.get_stock_historical_data задаются согласно документации investpy.)

In [3]:
```
def get_moex_data(ticker, from_date, to_date):
    df = investpy.get_stock_historical_data(stock=ticker,
                                            country='russia',
                                            from_date=from_date,
                                            to_date=to_date)
    df = df[['Close']]  # берем только цены закрытия
    df.rename(columns={'Close': ticker}, inplace=True)
    return df

moex_tickers = [
    'GAZP',
    'SBER',
    'LKOH',
    'VTBR',
    'ROSN',
    'GMKN',
    'AFKS',
    'PLZL',
    'TATN',
    'NLMK',
    'MGNT',
```

```
    'MTSS',
    'PIKK',
    'IRAO',
    'AFLT',
    'CBOM',
    'RTKM',
    'HYDR',
    'BSPB',
    ]  # примерный набор

# Период за последние 6 месяцев
end_date = datetime.date.today()
start_date = end_date - datetime.timedelta(days=180)  # ~6 месяцев (180 дней)

# Формат дат для investpy: 'dd/mm/yyyy'
start_date_str = start_date.strftime('%d/%m/%Y')
end_date_str = end_date.strftime('%d/%m/%Y')

# Загружаем данные по всем тикерам
data_frames = []
for ticker in moex_tickers:
    try:
        df_temp = get_moex_data(ticker, start_date_str, end_date_str)
        data_frames.append(df_temp)
    except Exception as e:
        print(f"Ошибка загрузки {ticker}: {e}")


# Объединяем все DataFrame в один по датам
if not data_frames:
    raise ValueError("Не удалось загрузить данные ни по одному тикеру.")

# Объединяем данные по дате:
data = pd.concat(data_frames, axis=1)
data.dropna(inplace=True)
data
```

```
Ошибка загрузки SBER: ERR#0015: error 403, try again later.
Ошибка загрузки VTBR: ERR#0015: error 403, try again later.
Ошибка загрузки GMKN: ERR#0015: error 403, try again later.
Ошибка загрузки PLZL: ERR#0015: error 403, try again later.
Ошибка загрузки NLMK: ERR#0015: error 403, try again later.
Ошибка загрузки PIKK: ERR#0015: error 403, try again later.
Ошибка загрузки CBOM: ERR#0015: error 403, try again later.
Ошибка загрузки BSPB: ERR#0015: error 403, try again later.
```

Out[3]:

Date	GAZP	LKOH	ROSN	AFKS	TATN	MGNT	MTSS	IRAO	AFLT	RTKM	HYDR
2024-09-13	119.91	6624.5	498.00	16.54	607.3	4998.0	196.00	3.7050	48.01	75.65	0.5149
2024-09-16	121.73	6820.5	516.30	16.49	621.2	5359.5	207.50	3.7895	49.29	78.36	0.5397
2024-09-17	123.10	6803.0	517.95	16.74	635.4	5694.5	209.55	3.8075	51.78	79.33	0.5492
2024-09-18	122.18	6745.5	510.25	16.25	626.8	5597.5	204.90	3.8400	51.27	78.90	0.5385
2024-09-19	122.24	6811.0	512.55	16.23	624.2	5605.0	206.40	3.8195	52.19	78.06	0.5343
...	...	...	...	...	...	...	...	...	...	...	...
2025-03-06	172.93	7175.0	529.55	17.92	700.5	5000.0	243.15	3.7585	73.82	72.55	0.5439
2025-03-07	171.56	7225.0	528.80	17.98	693.0	4936.5	242.00	3.7330	75.10	71.04	0.5327
2025-03-10	171.09	7185.0	527.30	18.51	683.6	4841.5	239.75	3.7260	74.71	71.19	0.5200
2025-03-11	170.42	7248.5	527.35	18.43	694.7	4900.0	240.80	3.7365	74.26	69.78	0.5150
2025-03-12	168.49	7207.0	529.60	18.18	687.7	4846.5	240.20	3.7120	73.18	67.80	0.5105

126 rows × 11 columns

Переводим цены в логарифмы

In [4]:
```
log_data = np.log(data)
```

Тест Дики-Фуллера

In [5]:
```
def adf_stationary_test(series, signif=0.05):
    result = adfuller(series, autolag='AIC')
    p_value = result[1]
    return (p_value < signif)
```

Проверяем, какие логарифмы цен являются I(1).

In [6]:
```
i1_tickers = []
for ticker in log_data.columns:
    series = log_data[ticker].dropna()
    # Проверяем стационарность log(Price)
    if not adf_stationary_test(series):
        # Если log(Price) нестационарен, проверяем разности
        diff_series = series.diff().dropna()
        if adf_stationary_test(diff_series):
            i1_tickers.append(ticker)

print("Тикеры, у которых логарифм цены I(1):", i1_tickers)
```

```
Тикеры, у которых логарифм цены I(1): ['GAZP', 'LKOH', 'ROSN', 'AFKS', 'TATN', 'MGNT', 'MTSS', 'AFLT', 'RTKM', 'HYDR']
```

Оставляем в датафрейме только эти тикеры

In [7]:
```
log_data_i1 = log_data[i1_tickers].dropna()
```

Тест Энгла–Грэнджера

In [8]:
```
def engle_granger_test(y, x):
    x_const = sm.add_constant(x)
    model = sm.OLS(y, x_const).fit()
    residuals = model.resid
    # Тест на стационарность остатков
    return adf_stationary_test(residuals)
```

Проведем его для всех пар

In [9]:
```
coint_pairs = []
for t1, t2 in combinations(i1_tickers, 2):
    # Берём данные
    s1 = log_data_i1[t1]
    s2 = log_data_i1[t2]

    # Можно решить, что будем регрессировать "более дорогую" на "более дешёвую"
    # или просто попробовать оба порядка и посмотреть статистику.
    # Для упрощения — один порядок:
    if s1.mean() > s2.mean():
        y, x = s1, s2
        pair_name = f"{t1}-{t2}"
    else:
        y, x = s2, s1
        pair_name = f"{t2}-{t1}"

    # Проводим тест
    if engle_granger_test(y, x):
        coint_pairs.append(pair_name)

print("Найденные коинтегрированные пары:")
for pair in coint_pairs:
    print(pair)
```

```
Найденные коинтегрированные пары:
LKOH-GAZP
GAZP-AFKS
GAZP-AFLT
LKOH-AFKS
LKOH-TATN
LKOH-MTSS
LKOH-AFLT
LKOH-RTKM
LKOH-HYDR
AFLT-AFKS
```