Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та комп’ютерних технологій

Звіт

Про виконання лабораторної роботи №8

З курсу «Системи опрацювання даних**»**

«Реалізація ETL-процесу,

Автоматизоване збирання, трансформація та збереження даних»

Виконав:

Студент групи ФеС-21

Осадчук Дмитро

Львів-2025

Лабораторна робота 8: Реалізація ETL-процесу

Тема: Автоматизоване збирання, трансформація та збереження даних

Платформа: Google Colab (https://colab.research.google.com/) або локально в Python.
Бібліотеки: `pandas`, `requests`, `json`.

Дані: Дані про курси валют із Open Exchange Rates API (https://openexchangerates.org/) або ExchangeRate-API (https://www.exchangerate-api.com/) або API НБУ.

Завдання:

1. **Збір даних:**
   - Автоматичне отримання курсів валют на основі щоденного API-запиту.
   - Наприклад
   - 

   - Додавання поточного часу отримання даних.
2. **Трансформація:**
   Для 20 довільних валют провести
   - Підрахунок середнього курсу за період (наприклад, тиждень).
   - Підрахунок відсоткової зміни курсу порівняно з попереднім періодом.
   - Розрахунок стандартного відхилення курсу за період.
3. **Збереження:**
   - Збереження даних для всіх валют у .csv.
   - Створення окремої таблиці для агрегованих даних .
4. **Візуалізація:**
   - Побудова графіків зміни курсу основних валют за період (наприклад, тиждень).
   - Створення діаграми для відображення коливань курсу з використанням діапазонів помилок.

Очікувані результати:

- Створення ETL-сценарію, який автоматизує процес збору, обробки та збереження даних.

Хід роботи:

```python
import requests
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime
import os
import uuid

APP_ID = 'f7dc9c1130fa479eb413bf09df17ae1e'

selected_currencies = ['EUR', 'GBP', 'JPY', 'AUD', 'CAD', 'CHF', 'CNY', 'SEK', 'NZD',
                       'MXN', 'SGD', 'HKD', 'NOK', 'KRW', 'TRY', 'INR', 'BRL', 'ZAR', 'RUB']
```

Підключення основних бібліотек для роботи, визначення конфігурації API сервісу, а також визначення основних валют

```python
# Функція для отримання поточних курсів валют
def fetch_today_rates():
    url = f"https://openexchangerates.org/api/latest.json?app_id={APP_ID}"
    try:
        response = requests.get(url)
        if response.status_code != 200:
            raise Exception(f"API помилка: {response.status_code} - {response.text}")

        data = response.json()
        rates = data.get('rates', {})
        today = datetime.now().strftime('%Y-%m-%d')

        today_rates = {'date': today}

        # Перевірка наявності кожної валюти
        for currency in selected_currencies:
            if currency not in rates:
                print(f"Попередження: валюта {currency} відсутня у відповіді API")
                today_rates[currency] = None
            else:
                today_rates[currency] = rates[currency]

        # Перевірка на коректність даних
        for key, value in today_rates.items():
            if key != 'date' and value is not None and not isinstance(value, (int, float)):
                print(f"Попередження: нечислове значення для {key}: {value}")
                today_rates[key] = None

        return today_rates
    except Exception as e:
        print(f"Помилка при отриманні даних від API: {str(e)}")
        return None
```

Функція для отриманні поточних курсів валют з за допомогою API ключа з сервісу

```python
# Функція для збереження даних у CSV
def save_to_csv(today_rates, filename='currency_rates_raw.csv'):
    try:
        file_exists = os.path.exists(filename)
        df = pd.DataFrame([today_rates])

        # Перевірка на наявність ком у значеннях
        for key, value in today_rates.items():
            if isinstance(value, str) and ',' in value:
                print(f"Попередження: значення для {key} містить кому: {value}")
                df[key] = df[key].replace(',', '')

        # Перевірка наявності необхідних колонок
        missing_columns = [col for col in selected_currencies if col not in df.columns]
        for col in missing_columns:
            df[col] = None

        # Упорядковуємо стовпці
        df = df[['date'] + selected_currencies]

        # Логування вмісту перед збереженням
        print(f"Зберігаємо дані: {df.to_dict(orient='records')}")

        # Запис у CSV
        df.to_csv(filename, mode='a', header=not file_exists, index=False, encoding='utf-8')
        print(f"Дані успішно збережено у файл {filename}")
    except Exception as e:
        print(f"Помилка при збереженні у CSV: {str(e)}")
```

Збереження даних у CSV файл, де вже будуть знаходитись підготовлені дані для обробки.

```python
try:

    today_rates = fetch_today_rates()
    if today_rates is None:
        raise Exception("Не вдалося отримати дані від API")

    print(f"Отримано дані за {today_rates['date']}: {today_rates}")

    # Очищаємо CSV перед збереженням нових даних
    if os.path.exists('currency_rates_raw.csv'):
        clean_csv()

    # Зберігаємо дані в CSV
    save_to_csv(today_rates)

    # Збір даних за останні кілька днів
    try:
        df = pd.read_csv('currency_rates_raw.csv', encoding='utf-8', on_bad_lines='warn')
        print("Дані з CSV успішно зчитано")
    except Exception as e:
        print(f"Помилка при зчитуванні CSV: {str(e)}")
        raise Exception("Зупинка через помилку в CSV")

    # Перевірка, чи DataFrame не порожній
    if df.empty:
        raise Exception("CSV файл порожній або містить лише некоректні дані")

    # Трансформація даних
    if 'date' not in df.columns:
        raise Exception("Стовпець 'date' відсутній у CSV")

    df['date'] = pd.to_datetime(df['date'])
    df.set_index('date', inplace=True)
    mean_rates = df.mean()
    percent_change = ((df.iloc[-1] - df.iloc[0]) / df.iloc[0]) * 100
    std_dev = df.std()
```

Основний блок коду, де відбувається очищення CSV перед збереженням нових даних, а також збір самих даних та їх трансформація (процес зміни формату, структури або значень даних, щоб зробити їх зручнішими для аналізу чи подальшої обробки)

```python
# Агреговані дані
agg_data = pd.DataFrame({
    'Середній курс': mean_rates,
    'Зміна (%)': percent_change,
    'Коливання (±)': std_dev
})

# Збереження агрегованих даних
agg_data.to_csv('currency_rates_aggregated.csv', encoding='utf-8')
print("Агреговані дані збережено у файл currency_rates_aggregated.csv")
```

Агрегація даних(процес об’єднання або підсумовування даних для отримання узагальнених показників. Отримання середнього значення, змінна відсоткової зміни

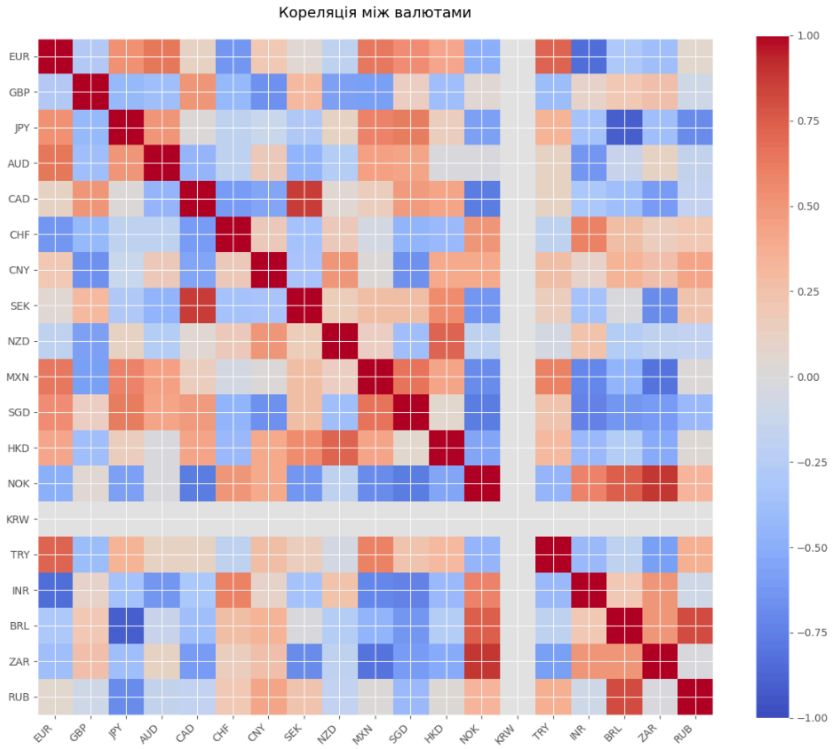
Побудова графіків:

```python
# Візуалізація
plt.style.use('ggplot')

# 1. Графік динаміки курсів
plt.figure(figsize=(14, 16))

plt.subplot(3, 1, 1)
for currency in selected_currencies[:8]:
    plt.plot(df.index, df[currency], marker='o', label=f'{currency}', linewidth=2)
plt.title('Динаміка обмінного курсу до USD', fontsize=14, pad=20)
plt.xlabel('Дата', fontsize=12)
plt.ylabel('Курс до USD', fontsize=12)
plt.xticks(rotation=45)
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left')
plt.grid(True, alpha=0.3)
```

```python
# 2. Середні курси
plt.subplot(3, 1, 2)
mean_rates.sort_values().plot(kind='barh', color='dodgerblue')
plt.title('Середній курс валют за період', fontsize=14, pad=20)
plt.xlabel('Середній курс до USD', fontsize=12)
plt.grid(True, axis='x', alpha=0.3)

# 3. Відсоткові зміни
plt.subplot(3, 1, 3)
colors = ['green' if x >= 0 else 'red' for x in percent_change]
percent_change.sort_values().plot(kind='barh', color=colors)
plt.title('Відсоткова зміна курсу за період', fontsize=14, pad=20)
plt.xlabel('Зміна курсу (%)', fontsize=12)
plt.grid(True, axis='x', alpha=0.3)

plt.tight_layout()
plt.savefig('currency_trends.png', bbox_inches='tight', dpi=100)
plt.close()
```

```python
# Теплокарта кореляції
plt.figure(figsize=(12, 10))
corr = df.corr()
plt.imshow(corr, cmap='coolwarm', vmin=-1, vmax=1)
plt.colorbar()
plt.xticks(range(len(selected_currencies)), selected_currencies, rotation=45, ha='right')
plt.yticks(range(len(selected_currencies)), selected_currencies)
plt.title('Кореляція між валютами', fontsize=14, pad=20)
plt.tight_layout()
plt.savefig('currency_correlation.png', dpi=100)
plt.close()

# Boxplot для розподілу курсів
plt.figure(figsize=(12, 6))
df[selected_currencies[:10]].boxplot(vert=False, patch_artist=True)
plt.title('Розподіл курсів валют', fontsize=14, pad=20)
plt.xlabel('Курс до USD', fontsize=12)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig('currency_distribution.png', dpi=100)
plt.close()
```

Отримані результати:

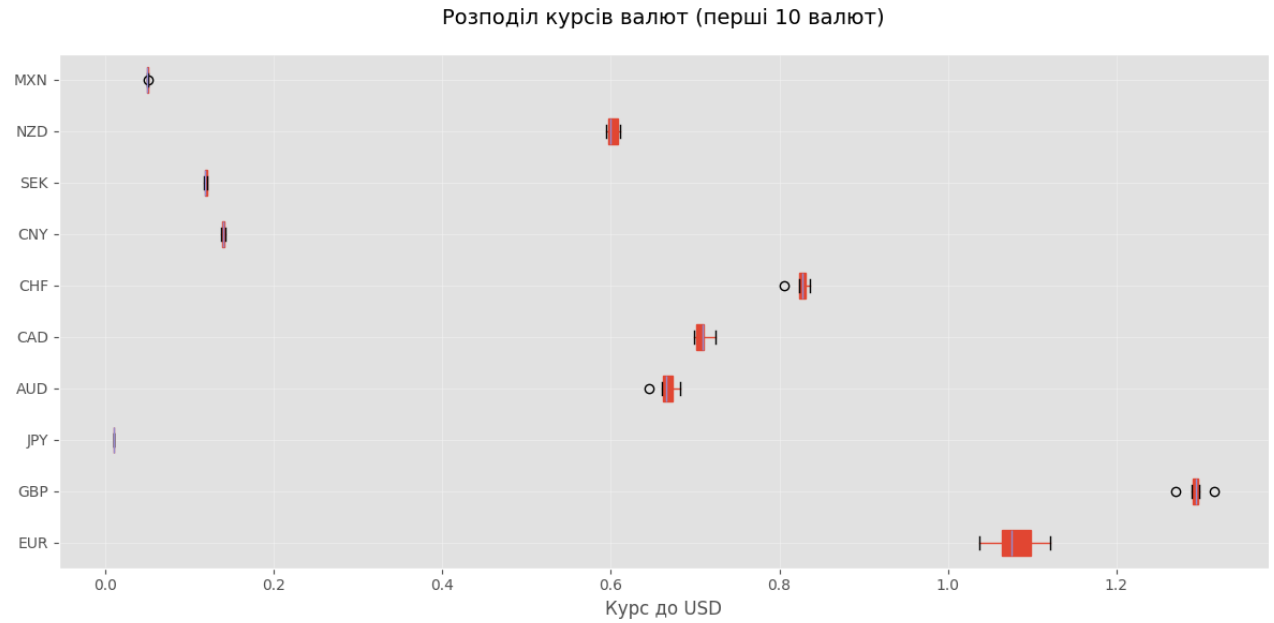


Негативна кореляція.

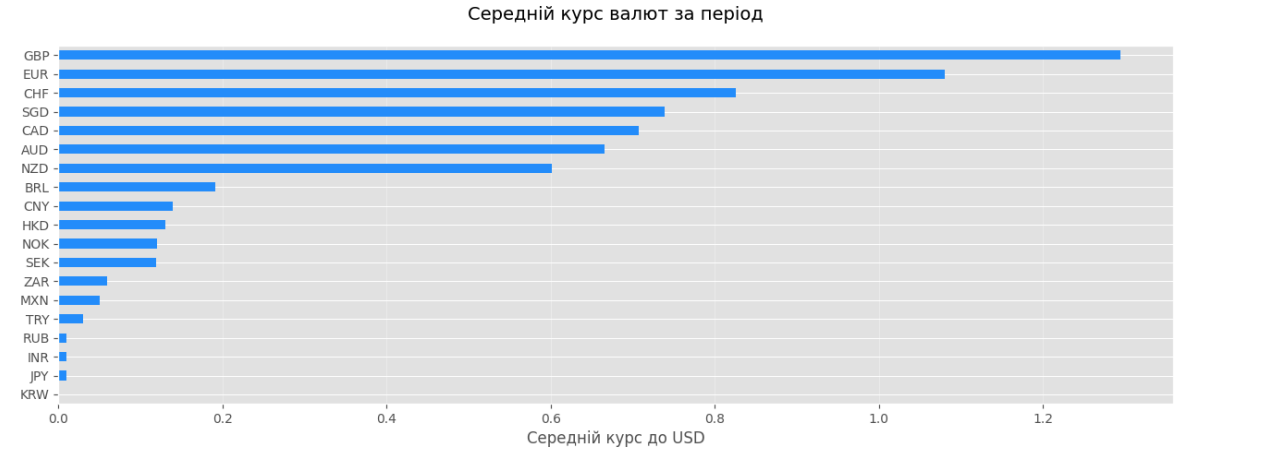
Між JPY (японська єна) та TRY (турецька ліра) є помітні сині клітинки (близько -0.5 до -0.75), що вказує на протилежний рух курсів, можливо різні економічні цикли.

Низька кореляція.

Наприклад, між RUB та SGD (сінгапурський долар) кореляція близька до 0, що означає, що їхні курси не пов’язані.



Валюти EUR, GBP, AUD (автрал. долар), CAD (канад. долар) мають курси в діапазоні 0.7–1.2 і відносно стабільний розподіл з невеликими коливаннями. NZD і CHF мають курси 0.6–1.2 із викидами, що вказують на аномалії в даних. MXN, SEK, CNY мають викиди на рівні 0.0–0.1, що свідчить про помилки в даних. Їхні основні значення курсів коливаються в межах 0.5–0.8. Загалом, валюти з викидами (MXN, SEK, CNY, CHF, NZD) вказують на проблеми зі збиранням або масштабуванням даних.



1) Більшість валют показують невеликі коливання протягом тижня, що свідчить про відносну стабільність на ринку.
2) **GBP, CAD, AUD, NZD**: Найвищі середні курси (≈0.8–1.0). Це валюти країн із сильними економіками (Велика Британія, Канада, Австралія, Нова Зеландія), які ближчі до паритету з USD.

   BRL, NOK, SEK, ZAR, MXN: Середні курси ≈0.4–0.6. Ці валюти слабші, але все ще мають помірний курс.

   RUB, INR, TRY, KRW: Найнижчі середні курси (≈0.0–0.2). Це валюти країн із менш стабільними економіками або з великою кількістю одиниць за 1 USD (наприклад, KRW ≈1380 за 1 USD).

3) Червоні смужки – ослаблення валюти (курс виріс, валюта стала дешевшою).
   Зелені смужки – зміцнення валюти (курс впав, валюта стала дорожчою).

Висновок: в ході виконання ЛР-8 я навчився створювати ETL сценарій, навчився працювати з API сервісами, такими як (https://openexchangerates.org). Отримав відповідні результати, які відповідають моїм очікуванням, проте отримав помилку з японською йенною, яка полягала в неправильному масштабуванні.