Программные средства сбора, консолидации и аналитики данных

Смоляков Руслан Игоревич

Тема: «Лабораторная работа 1-2. Современный парсинг динамических веб-сайтов»

Цель работы: освоить современный стек технологий для сбора данных с динамических веб-сайтов.

Вариант - 21

Бизнес-кейс:

Анализ рынка авиаперевозок: исследование цен на билеты.

Источник данных для парсинга:

Aviasales.ru. Поиск билетов по популярному направлению (например, Москва - Сочи) на конкретные даты.

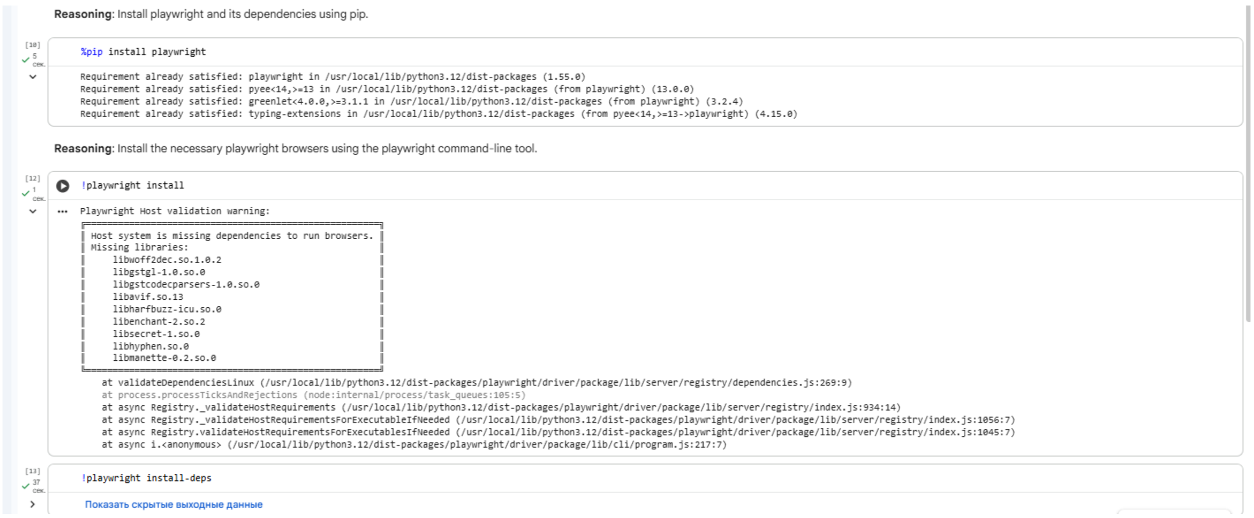
Аналитическая задача:

Собрать данные о предложениях: авиакомпания, цена, время в пути, наличие пересадок. Найти самую дешевую авиакомпанию для данного маршрута.

Ссылка на репозиторий:

https://github.com/Ruslanishka/soft-tools/tree/main/theme_1/lab_1

Подготовка:



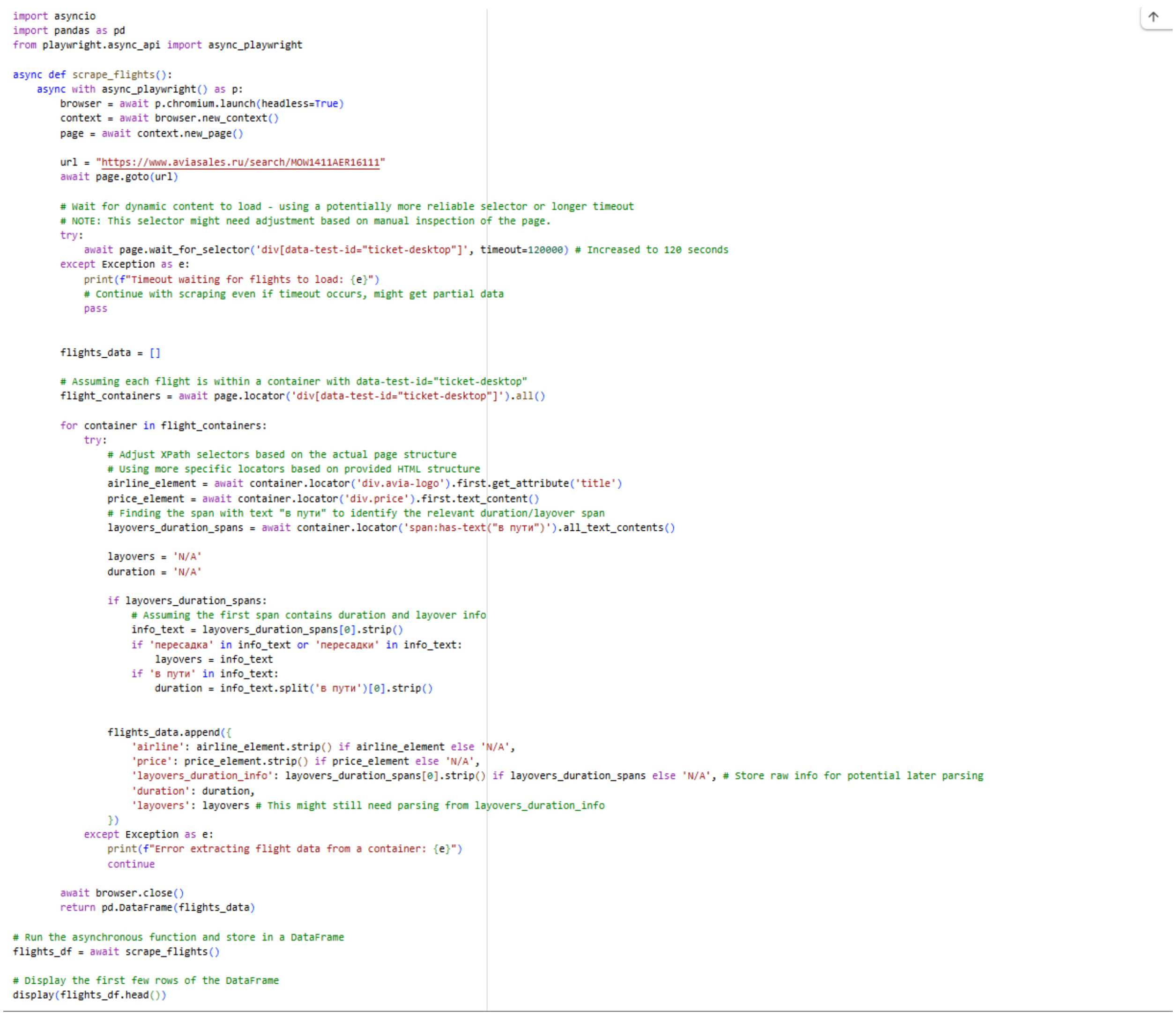
Reasoning: Install playwright and its dependencies using pip.

```
[10]
%pip install playwright

Requirement already satisfied: playwright in /usr/local/lib/python3.12/dist-packages (1.55.0)
Requirement already satisfied: pyee<14,>=13 in /usr/local/lib/python3.12/dist-packages (from playwright) (13.0.0)
Requirement already satisfied: greenlet<4.0.0,>=3.1.1 in /usr/local/lib/python3.12/dist-packages (from playwright) (3.2.4)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.12/dist-packages (from pyee<14,>=13->playwright) (4.15.0)
```

Reasoning: Install the necessary playwright browsers using the playwright command-line tool.

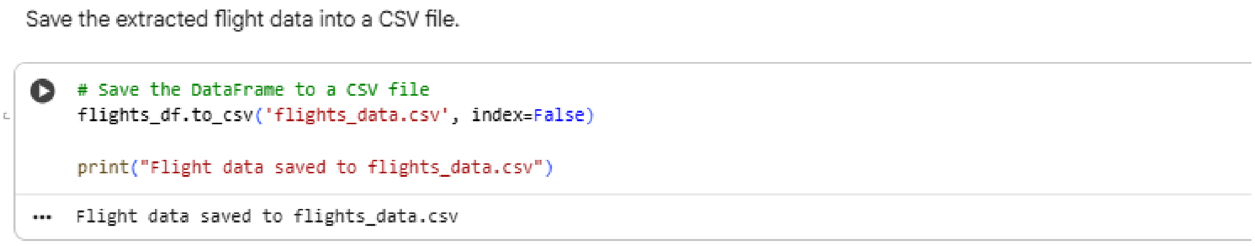
```
[12]
!playwright install

Playwright Host validation warning:
╔══════════════════════════════════════════════════════╗
║ Host system is missing dependencies to run browsers. ║
║ Missing libraries:                                   ║
║     libwoff2dec.so.1.0.2                             ║
║     libgstgl-1.0.so.0                                ║
║     libgstcodecparsers-1.0.so.0                      ║
║     libavif.so.13                                    ║
║     libharfbuzz-icu.so.0                             ║
║     libenchant-2.so.2                                ║
║     libsecret-1.so.0                                 ║
║     libhyphen.so.0                                   ║
║     libmanette-0.2.so.0                              ║
╚══════════════════════════════════════════════════════╝
    at validateDependenciesLinux (/usr/local/lib/python3.12/dist-packages/playwright/driver/package/lib/server/registry/dependencies.js:269:9)
    at process.processTicksAndRejections (node:internal/process/task_queues:105:5)
    at async Registry._validateHostRequirements (/usr/local/lib/python3.12/dist-packages/playwright/driver/package/lib/server/registry/index.js:934:14)
    at async Registry._validateHostRequirementsForExecutableIfNeeded (/usr/local/lib/python3.12/dist-packages/playwright/driver/package/lib/server/registry/index.js:1056:7)
    at async Registry.validateHostRequirementsForExecutablesIfNeeded (/usr/local/lib/python3.12/dist-packages/playwright/driver/package/lib/server/registry/index.js:1045:7)
    at async i.<anonymous> (/usr/local/lib/python3.12/dist-packages/playwright/driver/package/lib/cli/program.js:217:7)

[13]
!playwright install-deps

Показать скрытые выходные данные
```

«Скрапинг»



```python
import asyncio
import pandas as pd
from playwright.async_api import async_playwright

async def scrape_flights():
    async with async_playwright() as p:
        browser = await p.chromium.launch(headless=True)
        context = await browser.new_context()
        page = await context.new_page()

        url = "https://www.aviasales.ru/search/MOW1411AER16111"
        await page.goto(url)

        # Wait for dynamic content to load - using a potentially more reliable selector or longer timeout
        # NOTE: This selector might need adjustment based on manual inspection of the page.
        try:
            await page.wait_for_selector('div[data-test-id="ticket-desktop"]', timeout=120000) # Increased to 120 seconds
        except Exception as e:
            print(f"Timeout waiting for flights to load: {e}")
            # Continue with scraping even if timeout occurs, might get partial data
            pass


        flights_data = []

        # Assuming each flight is within a container with data-test-id="ticket-desktop"
        flight_containers = await page.locator('div[data-test-id="ticket-desktop"]').all()

        for container in flight_containers:
            try:
                # Adjust XPath selectors based on the actual page structure
                # Using more specific locators based on provided HTML structure
                airline_element = await container.locator('div.avia-logo').first.get_attribute('title')
                price_element = await container.locator('div.price').first.text_content()
                # Finding the span with text "в пути" to identify the relevant duration/layover span
                layovers_duration_spans = await container.locator('span:has-text("в пути")').all_text_contents()

                layovers = 'N/A'
                duration = 'N/A'

                if layovers_duration_spans:
                    # Assuming the first span contains duration and layover info
                    info_text = layovers_duration_spans[0].strip()
                    if 'пересадка' in info_text or 'пересадки' in info_text:
                        layovers = info_text
                    if 'в пути' in info_text:
                        duration = info_text.split('в пути')[0].strip()


                flights_data.append({
                    'airline': airline_element.strip() if airline_element else 'N/A',
                    'price': price_element.strip() if price_element else 'N/A',
                    'layovers_duration_info': layovers_duration_spans[0].strip() if layovers_duration_spans else 'N/A', # Store raw info for potential later parsing
                    'duration': duration,
                    'layovers': layovers # This might still need parsing from layovers_duration_info
                })
            except Exception as e:
                print(f"Error extracting flight data from a container: {e}")
                continue

        await browser.close()
        return pd.DataFrame(flights_data)

# Run the asynchronous function and store in a DataFrame
flights_df = await scrape_flights()

# Display the first few rows of the DataFrame
display(flights_df.head())
```

Сохраняем в csv



Save the extracted flight data into a CSV file.

```python
# Save the DataFrame to a CSV file
flights_df.to_csv('flights_data.csv', index=False)

print("Flight data saved to flights_data.csv")
```

```
Flight data saved to flights_data.csv
```