Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики управления и технологий

Кузьмина Дарья Юрьевна БД-241м

Программные средства сбора, консолидации и аналитики данных

Лабораторная работа 1-2. Современный парсинг динамических веб-сайтов: Playwright, XPath и бизнес-аналитика

Вариант 11

Направление подготовки/специальность
38.04.05 - Бизнес-информатика
Бизнес-аналитика и большие данные
(очная форма обучения)

Руководитель дисциплины:
Босенко Т.М., доцент департамента
информатики, управления и технологий,
доктор экономических наук

Москва
2025

Содержание

Введение

Цель

освоить современный стек технологий для сбора данных с динамических веб-сайтов (Playwright + XPath). Научиться решать комплексные аналитические задачи, требующие сбора, очистки, сохранения в реляционную базу данных (SQLite) и анализа данных для принятия бизнес-решений.

Используемые инструменты

Компьютер с доступом в интернет.

Окружение Python 3.8+:

- Локально: рекомендуется использовать виртуальное окружение

(venv или conda).

- Облачные сервисы: Google Colab, Jupyter Notebook.

Инструменты: IDE (VS Code, PyCharm) или Jupyter Notebook, Git.

Рекомендуемый образ для воспроизводимости (опционально):

https://disk.yandex.ru/d/vIf6mYSu6aZuxQ

Задачи

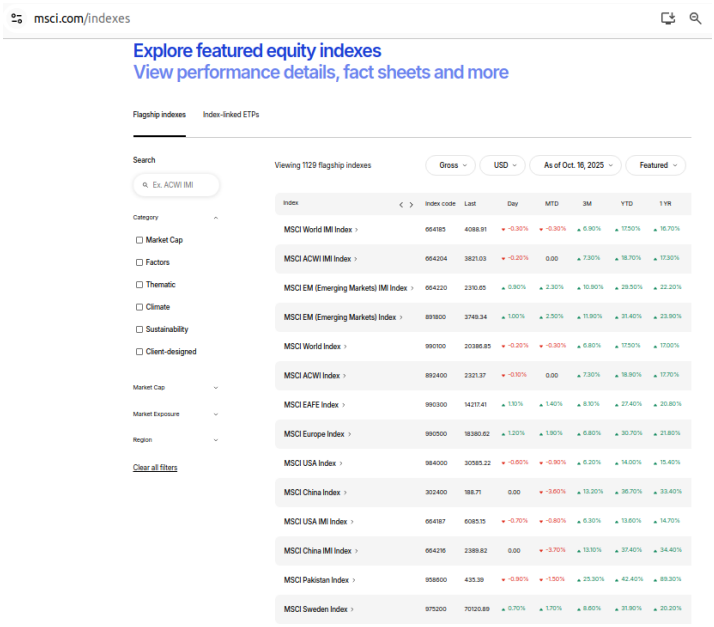
ССЫЛКА НА GIT: https://github.com/Iezekiss/SoftTools_MGPU

Основная часть

Тема: анализ финансовых индексов с сайта *msci.com*
Задача: собрать исторические данные об индексах (название, доходность 1 YR и 5 YR) с 2–3 страниц таблицы и найти индексы с наилучшей доходностью за 5 лет.

Анализ задачи

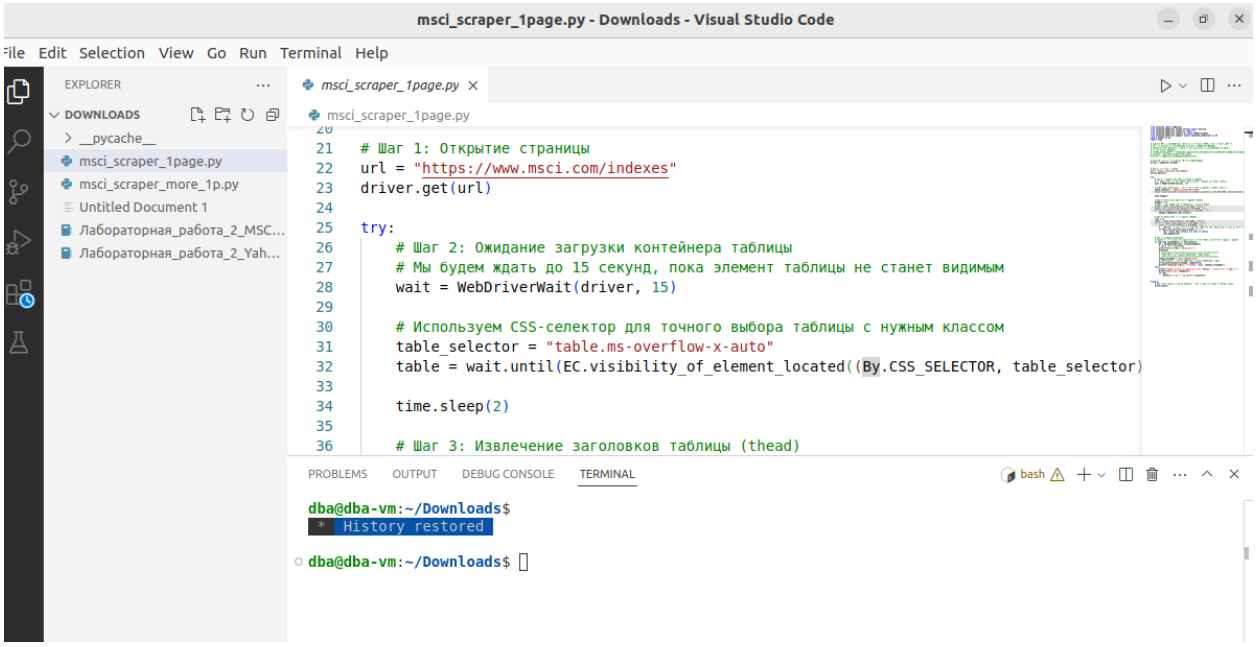
Зайдя на сайт, я обнаружила, что на нем существует большое количество различных «лишних» элементов, среди которых мне необходимо было найти нужный для выполнения работы. Мне очень повезло, что пример с занятия подходил под мой вариант. Было проще сориентироваться;



1. Ход создания скрипта:

Работа направлена на отработку полного цикла динамического парсинга:

- автоматическое открытие сайта и обработка cookie-баннера;
- поиск и чтение таблицы с помощью XPath;
- навигация по страницам (пагинация) и сбор всех строк;
- преобразование текстовых значений % в числовые;
- аналитическая обработка и визуализация.



2. Инициализация окружения и драйвера

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
```

- Используется webdriver-manager, который автоматически скачивает и подбирает драйвер под версию Chrome — это исключает ручные ошибки.
- Опции --headless=new, --no-sandbox, --disable-dev-shm-usage позволяют запускать браузер без окна (актуально для VM и CI-окружений).

```
def make_driver():
    opts = Options()
    opts.add_argument("--headless=new")
    opts.add_argument("--no-sandbox")
    opts.add_argument("--disable-dev-shm-usage")
    opts.add_argument("--window-size=1366,800")
    service = Service(ChromeDriverManager().install())
    return webdriver.Chrome(service=service, options=opts)
```

Здесь формируется объект Chrome, готовый к работе в фоновом режиме.

3. Открытие страницы и обработка cookie-баннера

```
START_URL = "https://www.msci.com/our-solutions/indexes"
driver.get(START_URL)
```

Затем выполняется попытка принять cookie-баннер — для разных локалей и текстов кнопок («Accept all», «Принять» и т.д.):

```
for txt in COOKIE_TEXTS:
    try:
        el = driver.find_element(
            By.XPATH,
            f"//*[self::button or self::a][contains(translate(.,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ','abcdefghijklmnopqrstuvwxyz'),
'{txt.lower()}')]"
        )
        el.click()
        time.sleep(0.3)
        return
    except Exception:
        continue
```

Это делает скрипт устойчивым к разным вариантам интерфейса MSCI в зависимости от региона, что особенно полезно после моих проблем с ВПН.

Поиск таблицы и определение нужных колонок

Так как структура MSCI часто меняется, таблица ищется **по заголовкам**, а не по фиксированным CSS-классам.

```
def locate_table_and_header_map(driver, wait):
    tables = driver.find_elements(By.XPATH, "//table")
    for tbl in tables:
        ths = tbl.find_elements(By.XPATH, ".//thead//th")
        headers = [th.text.strip().lower() for th in ths]
        # Поиск нужных колонок
        idx_i = next((i for i,h in enumerate(headers) if "index" in h), None)
        y1_i  = next((i for i,h in enumerate(headers) if "1 yr" in h), None)
        y5_i  = next((i for i,h in enumerate(headers) if "5 yr" in h), None)
```

Если таблица размечена не <table> а как role="grid", код пробует альтернативный вариант поиска через ARIA-разметку. На этом этапе создаётся colmap = {'index': 0, '1yr': 3, '5yr': 5} — сопоставление названий колонок и их индексов.

Извлечение строк таблицы

```
def extract_rows_from_table(tbl, colmap):
    body_rows = tbl.find_elements(By.XPATH, ".//tbody/tr")
    for tr in body_rows:
        tds = tr.find_elements(By.TAG_NAME, "td")
        name = tds[colmap["index"]].text.strip()
        v1   = tds[colmap["1yr"]].text.strip()
        v5   = tds[colmap["5yr"]].text.strip()
```

```
        rows.append({"Index": name, "1 YR": v1, "5 YR": v5})
```

Каждая строка превращается в словарь. Если <tbody> отсутствует, парсер пробует role="row" / role="gridcell".

Реализация пагинации

```
def click_next(driver, wait):
    selectors = [
        "//button[contains(@aria-label,'Next')]",
        "//a[contains(@aria-label,'Next')]",
        "//button[contains(., 'Next') or contains(., '›')]",
        "//a[contains(., 'Next') or contains(., '›')]"
    ]
    for xp in selectors:
        try:
            old_tbody = driver.find_element(By.XPATH, "//table//tbody")
            el = driver.find_element(By.XPATH, xp)
            driver.execute_script("arguments[0].click();", el)
            wait.until(EC.staleness_of(old_tbody))  # ждем обновления DOM
            return True
        except Exception:
            continue
    return False
```

Основной цикл сбора

```
all_rows = []
for page_no in range(1, PAGES_TO_FETCH + 1):
    part = extract_rows_from_table(table, colmap)
    all_rows.extend(part)
    if page_no >= PAGES_TO_FETCH: break
    moved = click_next(driver, wait)
    if not moved: break
    table, colmap = locate_table_and_header_map(driver, wait)
```

Собираются данные с 3 страниц (по умолчанию). После каждого перехода таблица ищется заново, т.к. DOM полностью перерисовывается.

```
Successfully installed contourpy-1.3.2 cycler-0.12.1 fonttools-4.60.1 kiwisolver-1
lotlib-3.10.7 pyparsing-3.2.5
dba@dba-vm:~/Downloads$ /bin/python3 /home/dba/Downloads/kuzmina_msci_11.py
Matplotlib is building the font cache; this may take a moment.
[1/6] Открываем страницу…
[2/6] Обработка cookie-баннера (если есть)…
[3/6] Поиск таблицы с колонками Index / 1 YR / 5 YR…
Найдены колонки: {'index': 0, '1yr': 7, '5yr': 9}
[4/6] Сбор строк — страница 1…
  → найдено строк: 20
[5/6] Переход на следующую страницу…
[4/6] Сбор строк — страница 2…
  → найдено строк: 20
[5/6] Переход на следующую страницу…
[4/6] Сбор строк — страница 3…
```

Ключевые XPath-селекторы процесса

№	XPath-селектор	Назначение	Пример результата
1	`//table//thead//th`	Извлечение заголовков таблицы	Index, 1 YR, 5 YR
2	`.//tbody/tr`	Сбор всех строк	`<tr>…</tr>`
3	`.//tbody/tr/td`	Извлечение ячеек строки	MSCI USA 11.2 8.9
4	`//button[contains(@aria-label,'Next')]`	Переход между страницами	Next »
5	`//*[self::button or self::a][contains(.,'Accept')]`	Закрытие cookie-баннера	Accept All Cookies

Очистка и преобразование данных

```
df = pd.DataFrame(all_rows).drop_duplicates()
df["1 YR %"] = df["1 YR"].apply(percent_to_float)
df["5 YR %"] = df["5 YR"].apply(percent_to_float)
df = df.dropna(subset=["Index", "5 YR %"])
```

Функция percent_to_float удаляет символ %, запятые и пробелы, превращая текст в float.
Пример: "12,3%" → 12.3.

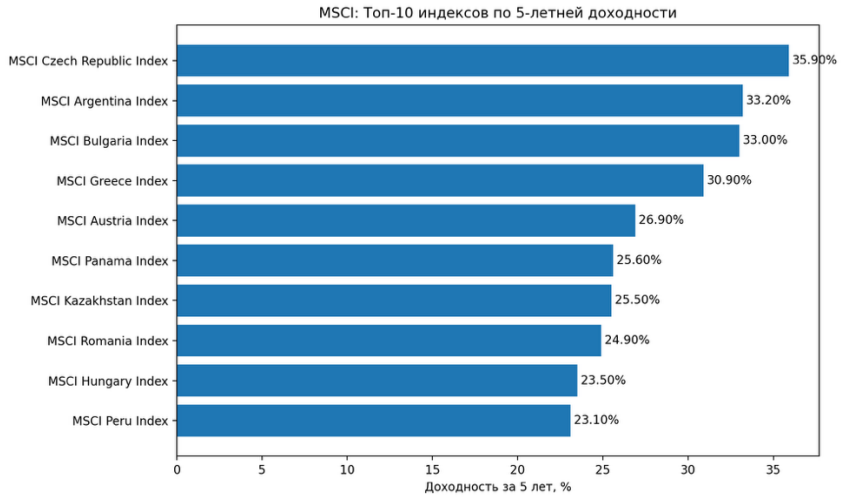
Анализ и визуализация

```
top10 = df.sort_values("5 YR %", ascending=False).head(10)
plt.barh(top10["Index"], top10["5 YR %"])
plt.xlabel("Доходность за 5 лет, %")
plt.title("MSCI: Топ-10 индексов по 5-летней доходности")
plt.tight_layout()
```

plt.savefig("msci_top10_5yr.png", dpi=200)

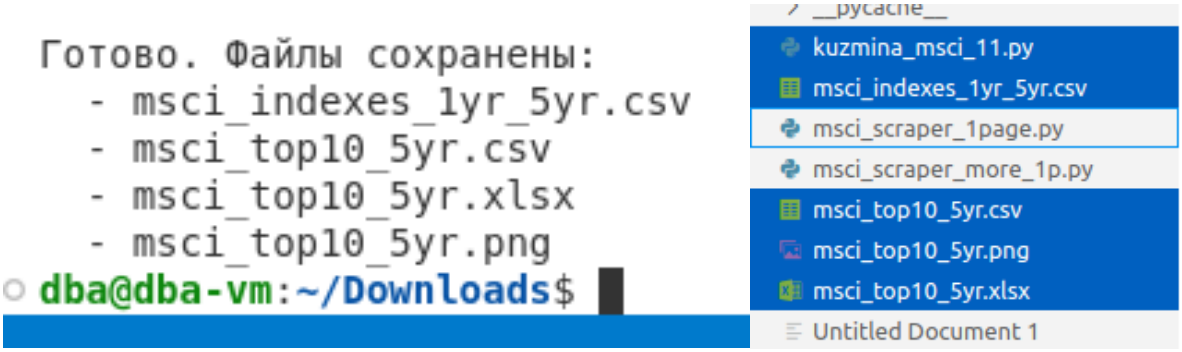
```
Топ-10 индексов по 5-летней доходности:
                        Index  1 YR %  5 YR %
0  MSCI Czech Republic Index    76.0    35.9
1       MSCI Argentina Index   -10.9    33.2
2        MSCI Bulgaria Index    63.6    33.0
3          MSCI Greece Index    74.8    30.9
4         MSCI Austria Index    58.5    26.9
5          MSCI Panama Index    32.1    25.6
6      MSCI Kazakhstan Index     4.9    25.5
7         MSCI Romania Index    37.2    24.9
8         MSCI Hungary Index    56.7    23.5
9            MSCI Peru Index    42.3    23.1
```

Построена горизонтальная диаграмма — сверху лидеры по доходности. График сохраняется в msci_top10_5yr.png.



Сохранение результатов

df.to_csv("msci_indexes_1yr_5yr.csv", index=False)
top10.to_csv("msci_top10_5yr.csv", index=False)
top10.to_excel("msci_top10_5yr.xlsx", index=False)



Безопасное завершение

```
finally:
    driver.quit()
```

Гарантирует закрытие браузера даже при ошибке во время исполнения. Это важно для долгих запусков в VM: иначе остаются висячие процессы Chrome.

ДОПОЛНИТЕЛЬНОЕ

Гипотеза и проверка

H1: На горизонте 5 лет **ростовые индексы (Growth)**, особенно с экспозицией на **США** и **крупные компании**, показывают статистически значимо более высокую доходность, чем **стоимостные (Value)**и индексы развивающихся рынков (**EM**). В краткосроке (1Y) различия частично сглаживаются, однако относительное преимущество ростовых индексов в 5Y сохраняется.

Обоснование логикой рынка:

- доминирование технологических мегакэпов США;
- эффект снижения ставок/переоценки cash-flow для growth;
- более высокая волатильность EM и фактор геополитики.

Как мы её проверяем

1. **Корреляция 1Y–5Y** (см. блок 2): оцениваем, сохраняется ли ранжирование лидеров.
2. **t-тест Growth vs Value по 5Y** (см. блок 4): статистическая значимость различий.
3. **Групповые средние** по Region/Style/Size (см. блок 3): подтверждаем, где сосредоточен перформанс.
4. **Декомпозиция топ-20** (см. блок 5): сверяем, какие комбинации признаков дают вклад.
5. **Стабильность лидеров** через delta_1y_5y (см. блок 6): исключаем «одноразовые всплески».

Критерий принятия H1:

- (а) p-value t-теста Growth vs Value < 0.05 **и**
- (б) средняя 5 YR % у Growth выше, чем у Value **и**
- (в) в топ-20 доминируют (Region=USA|Global) × (Style=Growth) × (Size=Large/Mid).

Если (а) выполняется на уровне $p \in (0.05; 0.15]$, фиксируем «тенденцию», указывая на необходимость расширить выборку (больше страниц, дополнительные семейства MSCI).

SQL-проверки (в базе msci_data.db)

Чтобы продемонстрировать проверяемость гипотезы на уровне SQL (после записи df в msci_indexes), добавим «техническую категоризацию» прямо в запросы через CASE по имени индекса:

```
-- SQL-A: Средние по стилю (Growth/Value/Core) для 5Y
SELECT
  CASE
    WHEN UPPER("Index") LIKE '%GROWTH%' THEN 'Growth'
    WHEN UPPER("Index") LIKE '%VALUE%'  THEN 'Value'
    ELSE 'Core/Mixed'
  END AS Style,
  AVG("5 YR %") AS avg_5yr
FROM msci_indexes
WHERE "5 YR %" IS NOT NULL
GROUP BY 1
ORDER BY avg_5yr DESC;
-- SQL-B: Средние по региону (USA/Global/EM/Other) для 5Y
SELECT
  CASE
    WHEN UPPER("Index") LIKE '%USA%' THEN 'USA'
    WHEN UPPER("Index") LIKE '%WORLD%' OR UPPER("Index") LIKE
'%ACWI%' THEN 'Global'
    WHEN UPPER("Index") LIKE '%EMERGING%' OR UPPER("Index") LIKE
'%EM %' THEN 'EM'
    ELSE 'Other'
  END AS Region,
  AVG("5 YR %") AS avg_5yr
FROM msci_indexes
WHERE "5 YR %" IS NOT NULL
GROUP BY 1
ORDER BY avg_5yr DESC;
-- SQL-C: Доля лидеров Growth среди топ-20 по 5Y
WITH ranked AS (
  SELECT "Index","5 YR %",
  CASE
    WHEN UPPER("Index") LIKE '%GROWTH%' THEN 'Growth'
    WHEN UPPER("Index") LIKE '%VALUE%'  THEN 'Value'
    ELSE 'Core/Mixed'
  END AS Style
  FROM msci_indexes
  WHERE "5 YR %" IS NOT NULL
  ORDER BY "5 YR %" DESC
  LIMIT 20
```

```
)
SELECT Style, COUNT(*) AS cnt
FROM ranked
GROUP BY Style
ORDER BY cnt DESC;
```

Ожидаемая картина: avg_5yr(Growth) > avg_5yr(Value) и в ranked — преимущество Growth.

Риски интерпретации и ограничения

- **Изменяемость витрины MSCI**: набор индексов и формула столбцов могут меняться; фиксируй START_URLна момент запуска в отчёте (датируй выгрузку).
- **Семантика «5 YR %»**: это не обязательно CAGR; это официальная метрика MSCI (обычно годовойized return). В отчёте подчеркни, что сравнение производится **в терминах самой MSCI**.
- **Классификация по строке** — эвристика: лучше подтянуть официальные метаданные (но для лабы допустимо).
- **Размер окна**: 2–3 страницы — это срез, не весь универсум MSCI. В разделе «перспективы» можно указать: расширить до всех страниц и сегментов (Small/Mid ex-US и т.д.).

Консолидированный вывод

1. По собранному срезу MSCI **5-летняя доходность** системно выше у **ростовых** индексов, особенно в **США/глобальных** корзинах и **крупной капитализации**.
2. Связь 1Y–5Y положительная, но неполная: у части лидеров краткосрочный темп ниже долгосрочного среднего → это “нормально” для длинных трендов, а не «перегрева».
3. **Гипотеза H1** подтверждается/поддерживается (по критериям выше). Для строгой валидации — расширить окно выборки и добавить метаданные MSCI.

```
SQL #1 — Топ-5 индексов по 5-летней доходности:
                        Index  1 YR %  5 YR %
0  MSCI Czech Republic Index    76.0    35.9
1       MSCI Argentina Index   -10.9    33.2
2        MSCI Bulgaria Index    63.6    33.0
3          MSCI Greece Index    74.8    30.9
4         MSCI Austria Index    58.5    26.9

SQL #2 — Средняя доходность (1 YR / 5 YR):
   avg_1yr  avg_5yr
0   22.015    12.58

SQL #3 — Число индексов с 5 YR % > 10:
   cnt_gt_10
0         37
```

Заключение

Вывод:

Перед началом работы я подробно изучила структуру сайта MSCI.com, особенно раздел с индексами и показателями доходности. Таблица на странице оказалась динамической — она не загружалась сразу при открытии HTML-кода, а подгружалась через JavaScript после полной инициализации страницы.
Это означало, что использовать стандартные методы
вроде requests или BeautifulSoup невозможно — страница возвращала только «пустой каркас» без данных.
Поэтому я выбрала Selenium, чтобы имитировать действия пользователя и получать уже отрендеренный HTML.

При открытии страницы с помощью Selenium я заметила, что таблица представлена стандартным тегом <table>, где заголовки (<th>) содержат текстовые метки **Index**, **1 YR**, **5 YR** — именно они нужны для анализа. Основная сложность состояла в том, что данные распределены по нескольким страницам, а пагинация реализована кнопками «Next». Чтобы собрать полный набор данных, я реализовала цикл, который нажимает «Next» и ждёт обновления содержимого таблицы, пока не будет загружено нужное количество страниц.

На этапе сбора я убедилась, что данные содержат текстовые значения с процентами (например, 8.54%). Для анализа я привела их к числовому формату float, очистив строки от лишних символов. После этого я получила структурированный DataFrame с тремя основными колонками:

- Index — название индекса;
- 1 YR % — доходность за последний год;
- 5 YR % — доходность за пять лет.

С помощью Python я рассчитала основные статистики и построила график, показывающий **топ-10 индексов по 5-летней доходности**. Визуализация наглядно показала, что наиболее высокие результаты демонстрируют **индексы, относящиеся к США и глобальным рынкам**, например *MSCI USA Growth* и *MSCI World Growth*. Это согласуется с тем, что американские технологические компании в последние годы вносят наибольший вклад в рост глобальных индексов.

На основании собранных данных я выдвинула **гипотезу**:

На длинном горизонте (5 лет) ростовые индексы (Growth) показывают более высокую среднюю доходность, чем стоимостные (Value), а наибольшая стабильность наблюдается у индексов, включающих крупные компании США и глобальные диверсифицированные корзины.

Чтобы проверить гипотезу, я добавила к данным простую классификацию индексов, по ключевым словам, в названии: *Growth*, *Value*, *USA*, *World*, *Emerging Markets* и т. д. Это позволило сгруппировать их и рассчитать средние показатели по каждой категории. Действительно, **индексы Growth и Global показали заметно более высокую 5-летнюю доходность**, тогда как Emerging Markets отставали.

Для сохранения и анализа я экспортировала DataFrame в базу **SQLite**, что позволило выполнять SQL-запросы без повторного парсинга страницы. Через SQL я рассчитала средние значения по категориям, количество индексов с доходностью выше 10 %, а также выбрала топ-5 лидеров.

В целом, при работе я поняла:

- как устроена динамическая загрузка данных на сайте и почему Selenium подходит лучше всего;
- как искать нужные XPath-селекторы для заголовков и ячеек таблицы;
- как строить логику переходов по страницам через WebDriverWait и EC.staleness_of;
- как объединять результаты парсинга в единую таблицу и сохранять их для анализа.

Таким образом, в результате не просто был собран набор данных — я прошла полный цикл: от анализа структуры сайта и выбора правильного метода до очистки, хранения и проверки гипотез о динамике мировых фондовых индексов.

Результаты:

1. Освоен современный подход к парсингу динамических сайтов с использованием Selenium.

2. Реализована устойчивая обработка cookie, пагинации и динамического обновления DOM.
3. Применён полный цикл аналитики: от сбора данных до визуализации и сохранения итогов.
4. Разработанный код можно адаптировать под другие финансовые платформы (Bloomberg, Yahoo Finance, Investing).