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

Нургалеева Гузель Рустэмовна БД-241м

Программные средства сбора, консолидации и аналитики данных

Лабораторная работа №1-2. Современный парсинг динамических веб-сайтов: Playwright, XPath и бизнес-аналитика

Вариант задания: 16

Направление подготовки/специальность

38.04.05 - Бизнес-информатика

Бизнес-аналитика и большие данные

(очная форма обучения)

Москва

2025

Цель работы: На примере бизнес-кейса «Исследование рынка фриланса: анализ проектов» освоить современный стек технологий для сбора данных с динамических веб-сайтов (XPath). Научиться решать комплексные аналитические задачи, требующие сбора, очистки, сохранения в реляционную базу данных (SQLite) и анализа данных для принятия бизнес-решений.

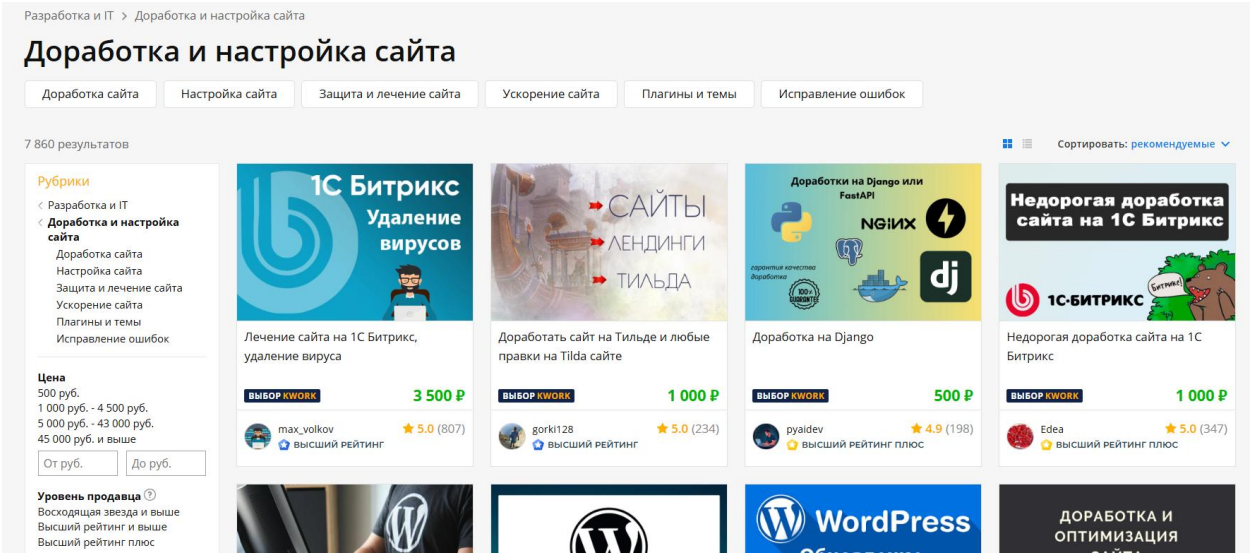
Ссылка на Git-репозиторий: https://github.com/GuzelN-4labs/XPath-Parsing.git

Описание бизнес-кейса и источника данных:

Источник данных: сайт **Kwork.ru**, раздел "Разработка и IT".
Задача: Обработать 2-3 страницы. Собрать название услуги, цену и количество выполненных заказов. Найти самые востребованные услуги.

Kwork – фриланс платформа, на которой можно приобрести услуги, связанные с цифровыми продуктами: веб-дизайн, разработка логотипа, разрабока и доработка сайтов, SEO, маркетинг, т.д. Фрилансеры оформляют свои услуги в виде кворков, которые можно купить в один клик... То есть работа исполнителей продается как товар. Также есть опция биржи фриланса с выбором специалиста по размещенной заявке по требуемым работам.

Рассматривались услуги раздела "Разработка и IT". Услуги размещены в виде карточек.



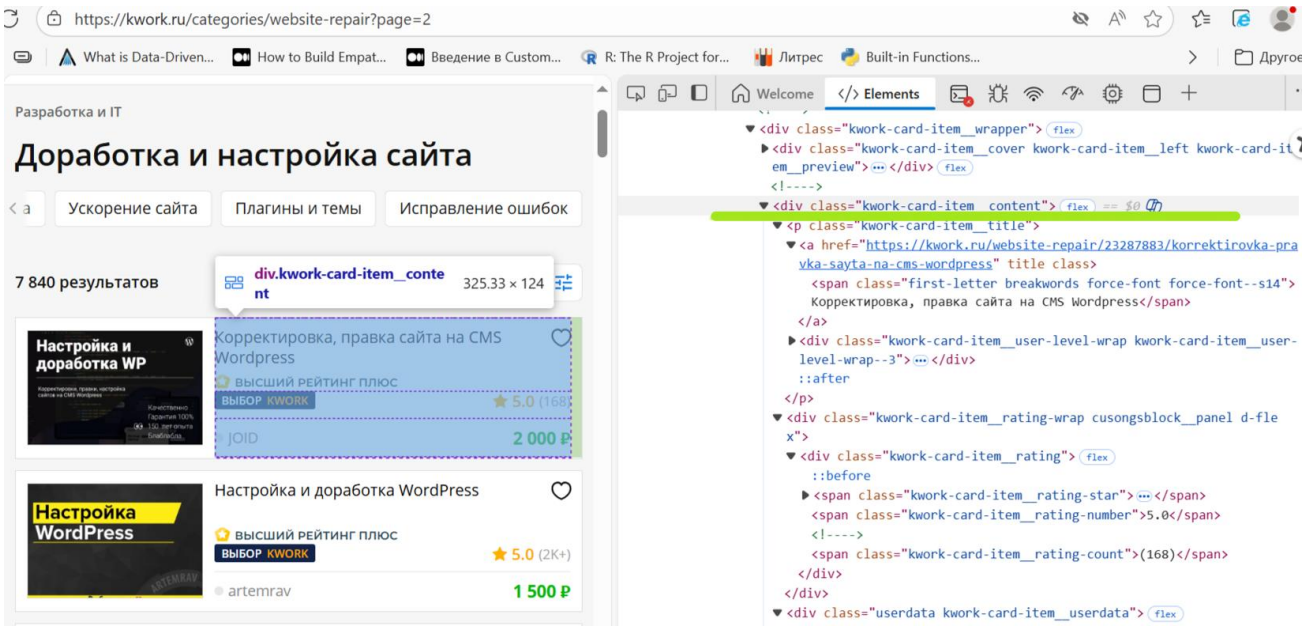
Ключевые XPath-селекторы:

Для парсинга были использованы следующие xPath-селекторы:

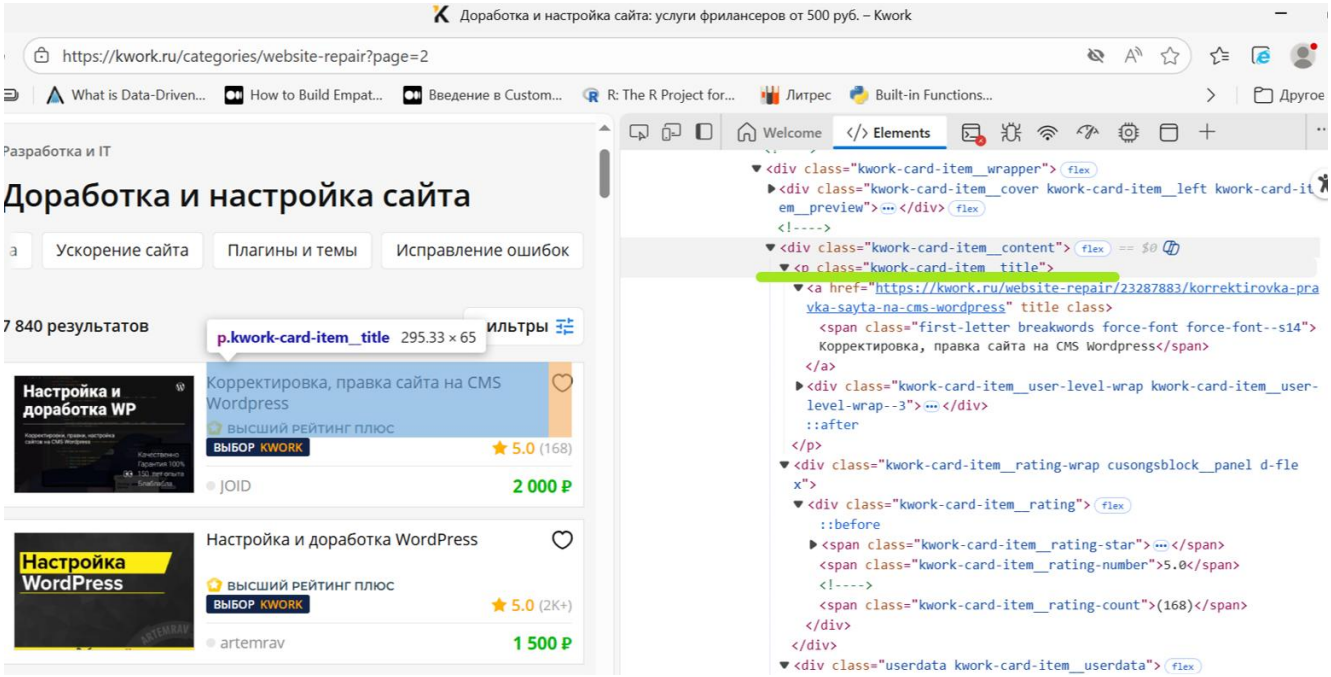
//	Выбирает узлы в документе, начиная с текущего, которые соответствуют выбору, где бы они ни находились.
@	Выбирает атрибуты
[…]	Предикат для фильтрации. Позволяет указать точные условия.

```
# Шаг 3: Извлечение данных
print("Извлечение данных...")
# Находим все элементы для каждой колонки отдельно. Используем XPath для карточек услуг.
# Название услуги
titles = [el.text for el in driver.find_elements(By.XPATH, '//p[@class="kwork-card-item__title"]')]
# Количество выполненных заказов
orders = [el.text for el in driver.find_elements(By.XPATH, '//span[@class="kwork-card-item__rating-count"]')]
# Цена
prices = [el.text for el in driver.find_elements(By.XPATH, '//span[@class="price-wrap__value force-font force-font--s15"]')]
```

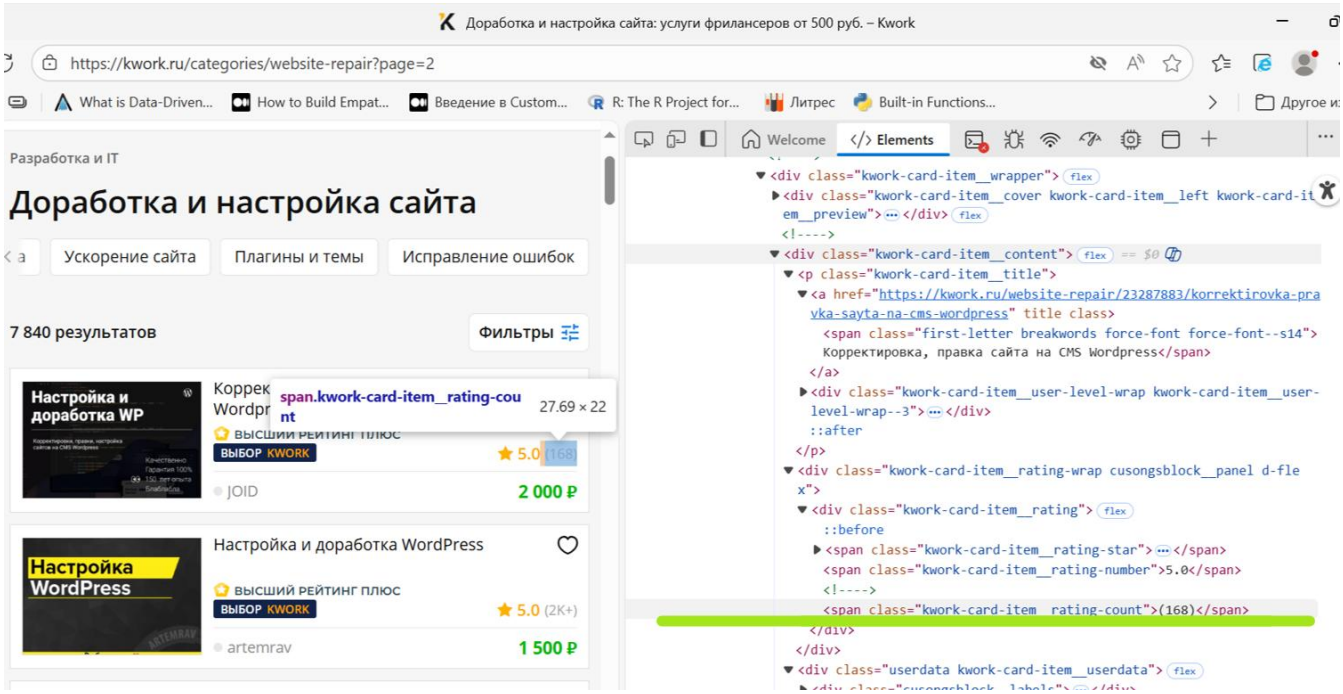
Часть карточки услуги, которая включает искомое описание



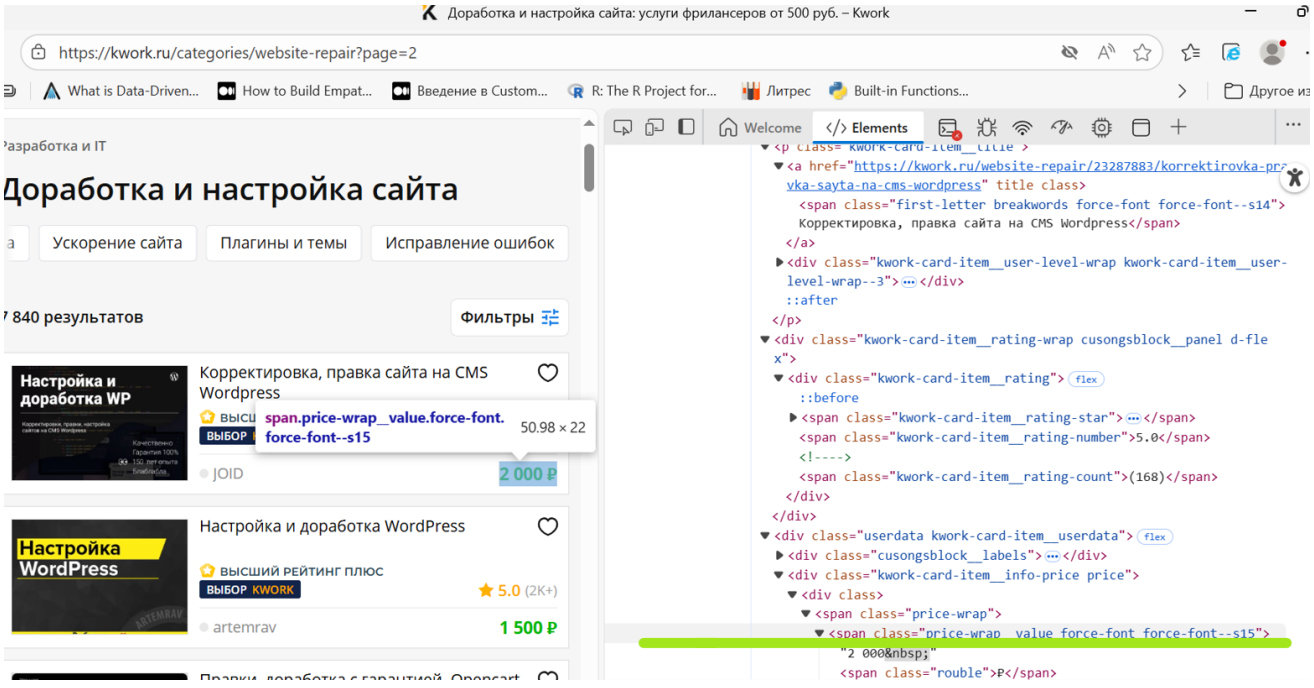
Название услуги



Количество выполненных работ



Стоимость услуги



Результаты анализа:

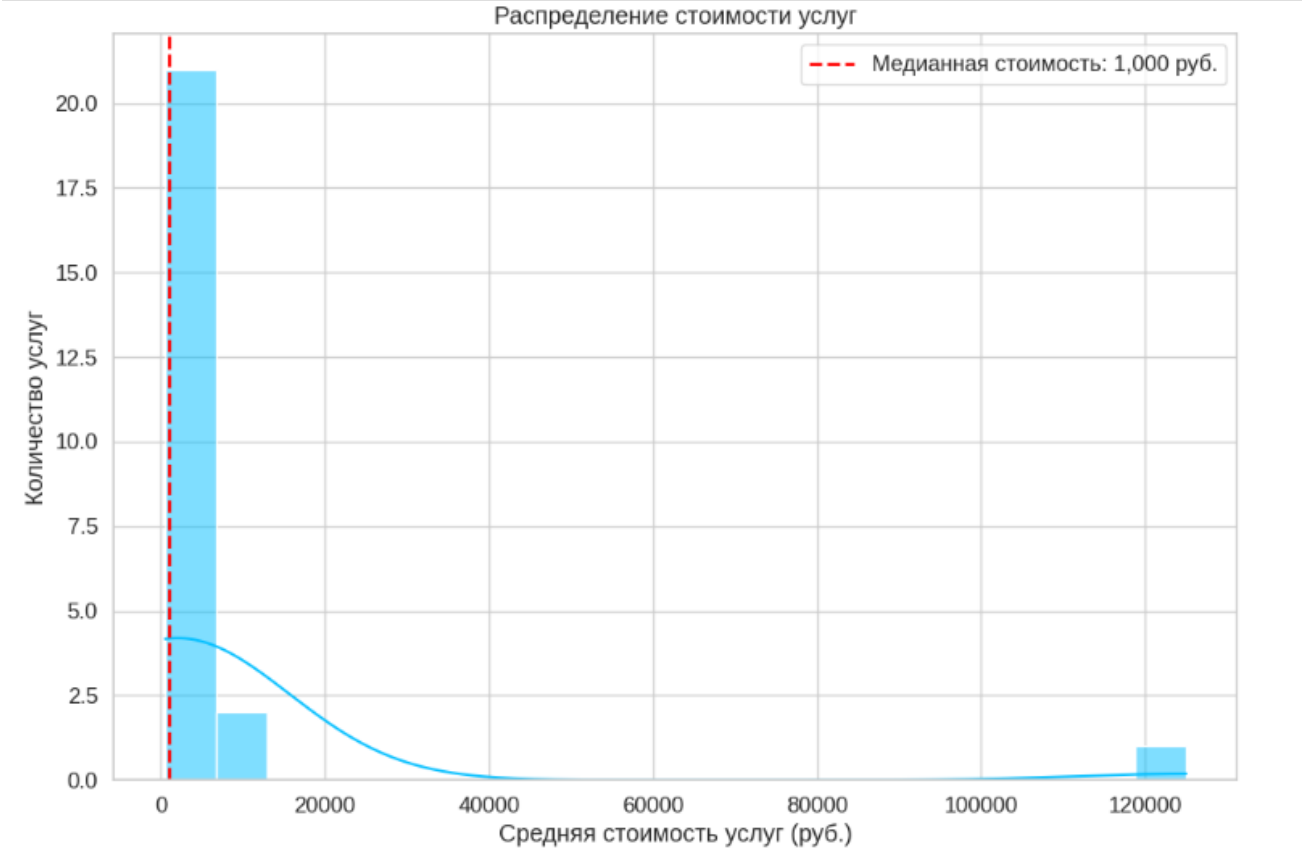
Ключевые фрагменты парсинга

```
driver = webdriver.Chrome(options=chrome_options)
driver.maximize_window()

url = 'https://kwork.ru/categories/website-repair'

try:
    print(f"Переход на страницу: {url}")
    driver.get(url)
    wait = WebDriverWait(driver, 30)

    # Шаг 1: Обработка всплывающего окна cookie (если оно появится)
    try:
        print("Поиск окна согласия на cookie...")
        # Используем более общий селектор, который подходит для разных текстов на кнопке
        agree_button = wait.until(EC.element_to_be_clickable((By.XPATH, "//button[contains(., 'Agree')] | //button[contains(., 'Accept all')]")))
        agree_button.click()
        print("Кнопка согласия нажата.")
    except TimeoutException:
        print("Окно согласия не найдено или уже принято. Продолжаем...")

    # шаг 2 удален, т.к. на сайте нет таблицы. Смотрим данные по карточкам, которые отображаются сразу

    # Шаг 3: Извлечение данных
    print("Извлечение данных...")
    # Находим все элементы для каждой колонки отдельно. Используем XPath для карточек услуг.
    # Название услуги
    titles = [el.text for el in driver.find_elements(By.XPATH, '//p[@class="kwork-card-item__title"]')]
    # Количество выполненных заказов
    orders = [el.text for el in driver.find_elements(By.XPATH, '//span[@class="kwork-card-item__rating-count"]')]
    # Цена
    prices = [el.text for el in driver.find_elements(By.XPATH, '//span[@class="price-wrap__value force-font force-font--s15"]')]
```

Преобразования и очистки данных

```
# 2. Очистка 'Completed Orders' и 'Price'
df_clean['Completed Orders'] = df_clean['Completed Orders'].astype(str).str.replace(r'[()+]', '', regex=True)

# Removing the Russian Ruble symbol and any non-digit characters except space (for thousands separator)
df_clean['Price'] = df_clean['Price'].astype(str).str.replace(r'[^\d\s]', '', regex=True).str.strip()

# Removing space as thousands separator
df_clean['Price'] = df_clean['Price'].str.replace(r'\s', '', regex=True)


# 3. Преобразование колонок в числовой формат
# Only 'Price' and 'Completed Orders' need conversion for Kwork data
cols_to_numeric = ['Price']
for col in cols_to_numeric:
    df_clean[col] = pd.to_numeric(df_clean[col], errors='coerce')

# Need to handle 'K' in 'Completed Orders' before converting to numeric
def convert_orders(orders_str):
    orders_str = str(orders_str).strip().upper()
    if orders_str.endswith('K'):
        return float(orders_str[:-1]) * 1000
    elif orders_str.endswith('+'): # Handle cases like '2K+' этот шаг не нужен, т.к. убрали знак + при помощи regex
         return float(orders_str[:-1]) * 1000
    return pd.to_numeric(orders_str, errors='coerce')

df_clean['Completed Orders'] = df_clean['Completed Orders'].apply(convert_orders)

# Удаляем строки, где могли возникнуть ошибки преобразования (стали NaN)
# Updated subset for Kwork data
df_clean.dropna(subset=['Price', 'Completed Orders'], inplace=True)
```

Визуализация



SQLite (соединение, сохранение в базе SQLite,запросы с обращением к базе)

```
import sqlite3

# df - ваш очищенный DataFrame после парсинга
# 1. Создание соединения с базой данных (файл будет создан,если не существует)

conn = sqlite3.connect('kwork_data.db')

# 2. Сохранение DataFrame в таблицу SQL.
# 'if_exists='replace'' перезапишет таблицу, если она уже существует.

table_name = 'parsed_data'
df_clean.rename({"Completed Orders": "Completed_Orders"}, axis =1).to_sql(table_name, conn, if_exists='replace', index=False)

# 3. Пример SQL-запроса: выбрать топ-5 услуг по цене,отсортированных по убыванию

query = "SELECT * FROM parsed_data ORDER BY Price DESC LIMIT 5"
result_df = pd.read_sql_query(query, conn)
print("Топ-5 самых дорогих услуг:")
display(result_df)

# 4. Закрытие соединения
conn.close()
```

```
Топ-5 самых дорогих услуг:
```

	Title	Completed_Orders	Price
0	AntiBot защита от накрутки поведенческих ботов...	662	125000
1	Правка или доработка сайта на Word Press	84	10000
2	Обновление WordPress, PHP и плагинов + скорост...	171	10000
3	Лечение сайта на 1С Битрикс, удаление вирусов,...	2000	6000
4	Правки, доработка с гарантией. Opencart, Опенк...	2000	2500

```
conn = sqlite3.connect('kwork_data.db')

query = "SELECT * FROM parsed_data WHERE Title like '%1С Битрикс%' ORDER BY 2 DESC"
result_df = pd.read_sql_query(query, conn)
print("Список услуг, касающихся доработки сайта на 1С Битрикс:")
display(result_df)

conn.close()
```

```
Список услуг, касающихся доработки сайта на 1С Битрикс:
```

	Title	Completed_Orders	Price
0	Лечение сайта на 1С Битрикс, удаление вирусов,...	2000	6000
1	Доработка сайта на 1С Битрикс	979	1000
2	Доработка сайта на 1С Битрикс	730	1000
3	Недорогая доработка сайта на 1С Битрикс	347	1000
4	Любая доработка и правка сайта 1С Битрикс	347	1000

```
conn = sqlite3.connect('kwork_data.db')

query = """
SELECT
SUM(Completed_Orders) as Orders_Completed,
SUM(Price*Completed_Orders) as Total_Value,
AVG(Price) as Avg_Price,
SUM(Price*Completed_Orders)/SUM(Completed_Orders) as Weighted_Avg_Price
FROM parsed_data
WHERE Title like '%1C Битрикс%'
"""
result_df = pd.read_sql_query(query, conn)
print("Саммери по работам, связанным с 1С Битрикс:")
display(result_df)

conn.close()
```

```
Саммери по работам, связанным с 1С Битрикс:
```

	Orders_Completed	Total_Value	Avg_Price	Weighted_Avg_Price
0	4403	14403000	2000.0	3271

```
conn = sqlite3.connect('kwork_data.db')

query = """
WITH CTE AS
(SELECT
SUM(Price*Completed_Orders) as Grand_Total,
SUM(CASE
    WHEN Title like '%1C Битрикс%' then Price*Completed_Orders
    ELSE null
    END) as Bitrix_orders_price
FROM parsed_data)
SELECT
Bitrix_orders_price,
Grand_Total,
Bitrix_orders_price*100/Grand_Total as 'Bitrix_share (%)'
FROM CTE
"""
result_df = pd.read_sql_query(query, conn)
print("На какую сумму выполнено работ по Bitrix, какая доля от общей суммы выполненных работ:")
display(result_df)

conn.close()
```

```
На какую сумму выполнено работ по Bitrix, какая доля от общей суммы выполненных работ:
```

	Bitrix_orders_price	Grand_Total	Bitrix_share (%)
0	14403000	118437500	12

Выводы

В ходе выполнения лабораторной работы мы успешно решили задачу парсинга данных с веб-сайта Kwork.ru, раздел "Разработка и IT".

1. **Применили знания XPath** для создания надежных селекторов
2. **Настроили и использовали Selenium** в Google Colab для автоматизации работы с браузером, включая обработку всплывающих окон и ожидание динамической загрузки контента.

3. **Извлекли данные** по самым востребованным работам и их стоимости и сохранили их в структурированном виде (Excel-файл).
4. **Провели комплексную предварительную обработку данных** с помощью Pandas, преобразовав текстовые значения количеств (например, '2K+') и цен с указанием валюты в результатах парсинга в числовые форматы, пригодные для анализа.
5. **Выполнили базовый анализ и визуализацию**, определив лидеров по объему выполняемых работ, распределению стоимости услуг, общей сстоимости выполненных услуг и доли в ней работ, связанных с доработкой 1С Битрикс.