ext_aeid-5012-OWARocksAsk_v11

Analysis & Design Document

Date – 28 Oct 2022

Table of Contents

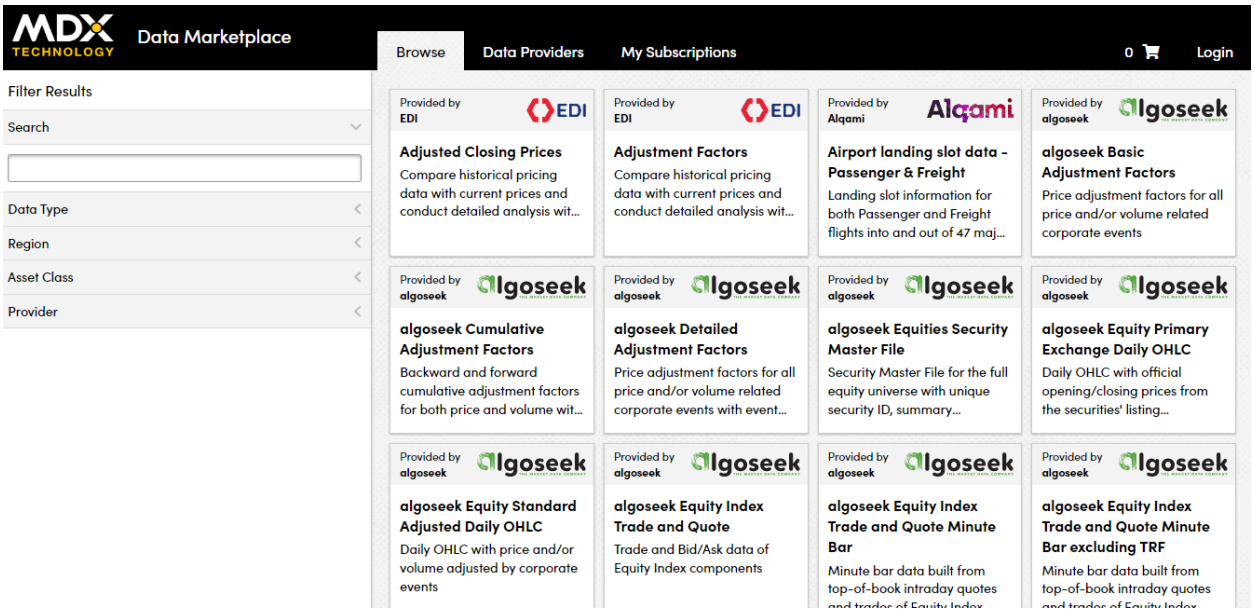
Topic **Page No.**

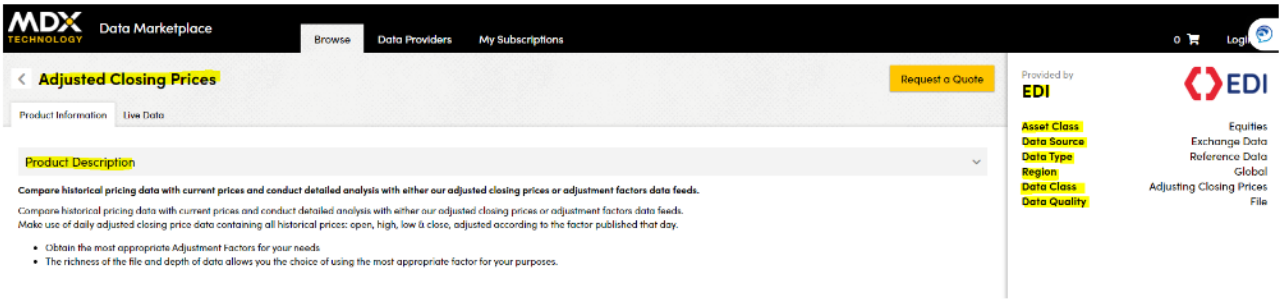
1. Scope of work

Scrap the below data from SITE: https://app.mdxtechnology.com/catalogue/products

For each card visible on the screen, click into the underlying information and scrap the below data

1. Provider
2. Asset Class
3. Data Source
4. Data Type
5. Region
6. Data Class
7. Data Quality





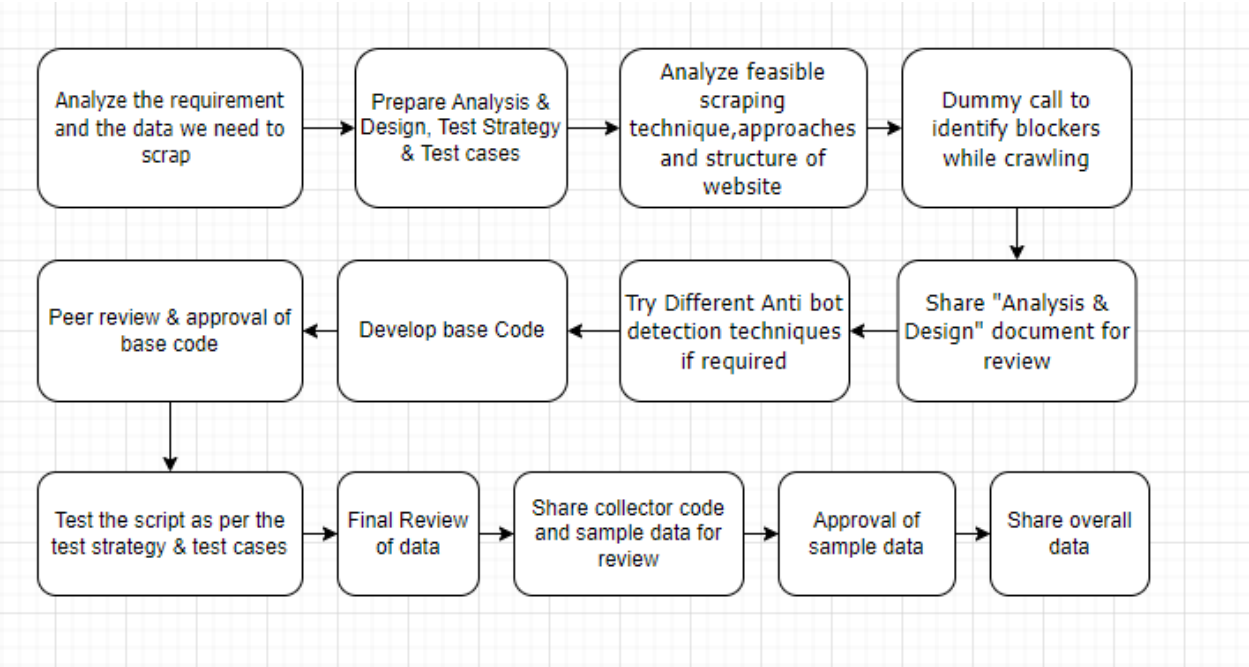
(The provider URL was sourced by hovering over "EDI" on the right side.)

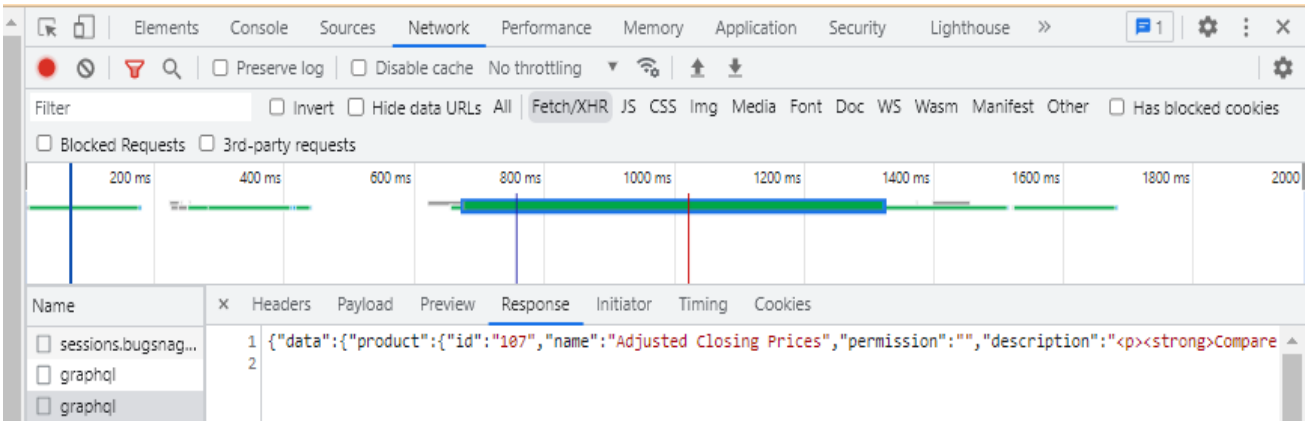
3. Melt the columns according to the schema described above. The keys are highlighted in yellow underneath "Description".
4. Push to BigQuery
5. Pivot columns in a view

2. Solution Approach

We are following the below steps to develop the script as per the requirement

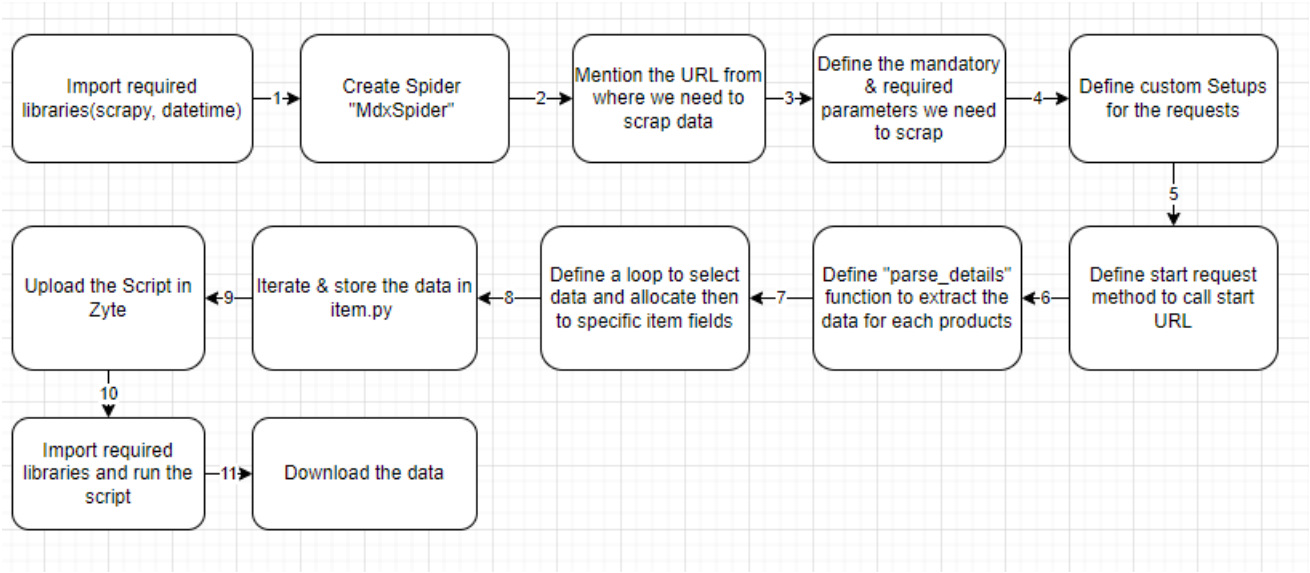
- The website is **global**, hence only one collector code is needed.
- We are fetching the required details for each product.
- Checked the javascript data (the data we get from AJAX calls) with the help of view page source.
- We are getting the data via graphql, when we are checking the fetch/XHR via Network.
- We are using request function to capture the data.


Analyze the requirement and the data we need to scrap
Prepare Analysis & Design, Test Strategy & Test cases
Analyze feasible scraping technique,approaches and structure of website
Dummy call to identify blockers while crawling
Share "Analysis & Design" document for review
Try Different Anti bot detection techniques if required
Develop base Code
Peer review & approval of base code
Test the script as per the test strategy & test cases
Final Review of data
Share collector code and sample data for review
Approval of sample data
Share overall data


Elements Console Sources Network Performance Memory Application Security Lighthouse
Preserve log Disable cache No throttling
Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies
Blocked Requests 3rd-party requests
200 ms 400 ms 600 ms 800 ms 1000 ms 1200 ms 1400 ms 1600 ms 1800 ms 2000
Name
Headers Payload Preview Response Initiator Timing Cookies
sessions.bugsnag...
graphql
graphql
{"data":{"product":{"id":"107","name":"Adjusted Closing Prices","permission":"","description":"<p>Compare

3. Script Development Flow

Below steps are followed to create spider



4. Technology Considerations

Custom signup - Not required

Programming Language - Python

Framework - Scrapy

Tool - Zyte

Functions & Libraries used - datetime, scrapy-user-agents

Storage (Database) - Zyte Cloud

Deployment Requirements

- Install all the required libraries in Zyte Cloud

Logging considerations

- No logging is required
- No CAPCTHA authentication required

Proxy Details

- We are using user agent to avoid getting blocked, this is present in settings.py file.

5. Base Collector Code

File name - mdx.py

Here we are scraping the data as per the requirements

Step 1 - Importing required libraries

```
import scrapy
import json
import datetime
from ..items import MdxMarketplaceItem
```

Step 2 - Here a spider named "MdxSpider" is created & allowed domain and start url of the website are defined that we are crawling

```
class MdxSpider(scrapy.Spider):
    name = 'AEID-4666_mdxtechnology'
    start_url = 'https://app.mdxtechnology.com/graphql'
```

Step 3 - Here we are defining the mandatory data

```
    # AEID_project_id = ''
    site = 'https://app.mdxtechnology.com/catalogue/products'
    source_country = 'Global'
    context_identifier = ''
    file_create_dt = datetime.datetime.utcnow().strftime('%Y-%m-%d %T')[0:10]
    record_created_by = ""
    execution_id = "622153" # This will be taken automatically from zyte, for now this is hardcoded
    feed_code = "AEID-4666"
    type = ""
    row = 0
```

Step 4 - Here we are defining the custom settings needed for Crawling

```
   custom_settings = {
        'ROBOTSTXT_OBEY': False,
        'CONCURRENT_REQUESTS': 20,
        'COOKIES_ENABLED': False,
        'COOKIES_DEBUG': False,
        'CONCURRENT_REQUESTS_PER_DOMAIN': 500,
        'DOWNLOAD_DELAY': 0,
```

```
        'AUTOTHROTTLE_ENABLED': False,
        'DOWNLOAD_TIMEOUT': 20,
        'DUPEFILTER_DEBUG': True,
    }
```

Step 5 - Defining the function to call start URL

```
    def start_requests(self):
        headers = {
            'accept': '* / *',
            'content-type': 'application/json',
            }
        payload = {"query": "query ($params: JSON) {\n  searchProducts(params: $params) {\n
products {\n    id\n    name\n    excerpt\n    is_hidden\n    has_sample\n    provider {\n
id\n    name\n    logo_url\n    is_hidden\n    __typename\n    }\n    __typename\n
}\n    buckets\n    __typename\n  }\n}\n",
                "variables": {"params": {}}}
        yield scrapy.Request(self.start_url, method="POST", headers = headers,
body=json.dumps(payload), callback=self.parse)
```

Step 6 - Here we are defining "parse_details". Inside this function we are writing code for crawling the data to process data for products. Here we are scraping all the required details

```
  def parse_details(self, response):
```

Step 7 - Here is a loop to select data and allocate then to specific item fields and storing it to items.py file

```
for i in data["data"]['product']['facet_values']:
            if i['facet']['name'] == 'Data Quality':
                data_quality = i['label']
            if i['facet']['name'] == 'Region':
                region.append(i['label'])
            if i['facet']['name'] == 'Data Class':
                cat_name = i['label']
            if i['facet']['name'] == 'Data Source':
                dat_source = i['label']
            if i['facet']['name'] == 'Asset Class':
                Asset_Class = i['label']
            if i['facet']['name'] == 'Data Type':
                Data_Type = i['label']
                data_id =  i['id']
```

6. Template Parameters & Description

The template contains the data that is scraped as per the ranking of newly listed products.

For the parameters where **mandatory** is mentioned, this is mandatory parameters as per the required template.

For the parameters where **Required** is mentioned, this is parameters needed as per the requirement document.

Below are the parameters that we are scraping and their description

1. **key -** Zyte by default add this as an identifier.
2. **row (Required) -** Adding indexing here.
3. **AEIDprojectId -** Harcoded the project id**.**
4. **datasetId (Required) -** We are getting this from website
5. **category_id (Required) -** We are getting this from website
6. **category_name (Required) -** We are getting this from website
7. **seller_id (Required) -**We are getting this from website
8. **seller_title (Required) -** We are getting this from website
9. **seller_url (Required) -** We are getting this from website
10. **product_name (Required) -** We are getting this from website
11. **format (Required) -** We are getting this from website
12. **delivery (Required) -** We are getting this from website
13. **frequency (Required) -** Kept as null s per the ASK document**.**
14. **description (Required) -** We are getting this from website
15. **region (Required) -** We are getting this from website
16. **history (Required) -** Kept as null s per the ASK document**.**
17. **price_raw (Required) -** Kept as null s per the ASK document**.**
18. **record_create_dt (Mandatory) -** Added the timestamp of scraping the data
19. **record_create_by (Mandatory) -** Hardcoded the spider name
20. **source_country (Mandatory) -** Hardcoded "Global" as this is the global site.
21. **Site (Mandatory) -** Hardcoded the website link
22. **Product_url (Mandatory) -** This is the individual product link.
23. **execution_id (Mandatory) -** This will be taken automatically from zyte.
24. **file_create_dt (Required) -** Added the current date
25. **Data_Type (Required) -** We are getting this from website
26. **Data_Quality (Required) -** We are getting this from website
27. **Data_Source (Required) -** We are getting this from website
28. **Data_class (Required) -** We are getting this from website
29. **Asset_Class (Required) -** We are getting this from website

7. Risks and Dependencies

Below are the identified risks and their possible solutions:

Risk	Mitigation
Risk of getting blacklisted/blocked/IP restrictions due to security/network policies on the web server.	we need to control the concurrency & use different proxy methods.
If the semantic code/markup of the website changes, the script will have a possibility of failure.	Identify the changes in the semantic code/markup of the website and modify the script accordingly.