ext_aeid4599-MedicalDisparities_v11

Analysis & Design Document

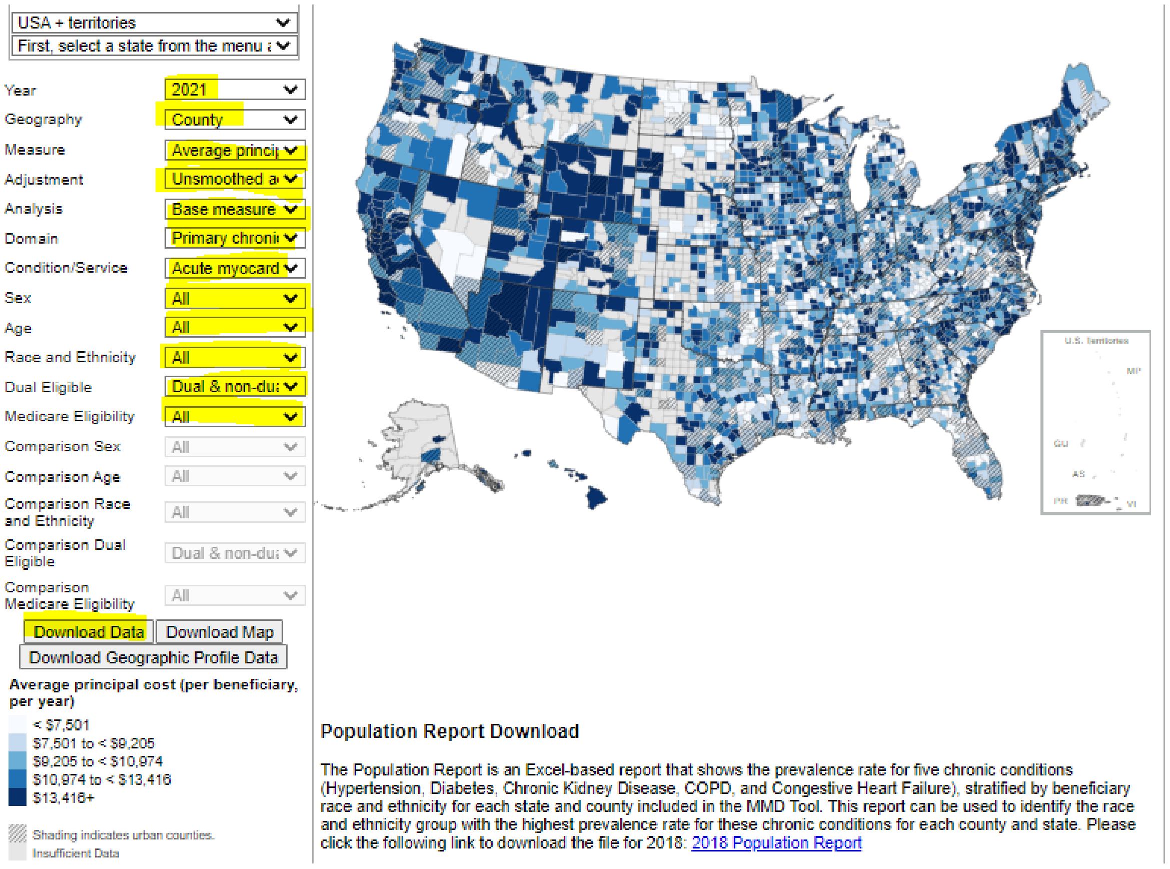
Date – 31 Oct 2022

Table of Contents

1. Scope of work

Scrap the below data from SITE: https://data.cms.gov/mapping-medicare-disparities

1. Each file has its own schema, want to come up with a common schema.
2. For each state, county and option in the dropdown menus, get the data.
3. Proceed to the set of combinations



USA + territories
First, select a state from the menu a

Year	2021
Geography	County
Measure	Average princi
Adjustment	Unsmoothed a
Analysis	Base measure
Domain	Primary chroni
Condition/Service	Acute myocard
Sex	All
Age	All
Race and Ethnicity	All
Dual Eligible	Dual & non-du
Medicare Eligibility	All
Comparison Sex	All
Comparison Age	All
Comparison Race and Ethnicity	All
Comparison Dual Eligible	Dual & non-du
Comparison Medicare Eligibility	All

Download Data | Download Map
Download Geographic Profile Data

Average principal cost (per beneficiary, per year)

⬜	< $7,501
🟦	$7,501 to < $9,205
🟦	$9,205 to < $10,974
🟦	$10,974 to < $13,416
🟦	$13,416+
▨	Shading indicates urban counties.
⬜	Insufficient Data

Population Report Download

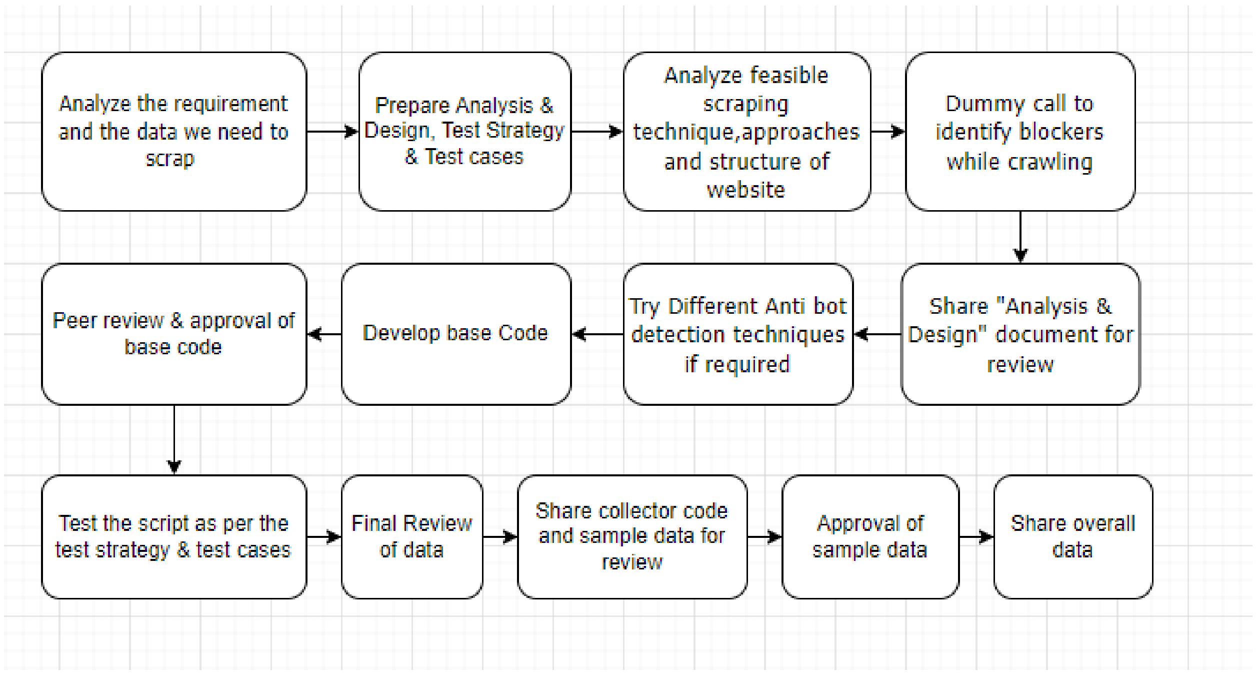
The Population Report is an Excel-based report that shows the prevalence rate for five chronic conditions (Hypertension, Diabetes, Chronic Kidney Disease, COPD, and Congestive Heart Failure), stratified by beneficiary race and ethnicity for each state and county included in the MMD Tool. This report can be used to identify the race and ethnicity group with the highest prevalence rate for these chronic conditions for each county and state. Please click the following link to download the file for 2018: 2018 Population Report

2. Solution Approach

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

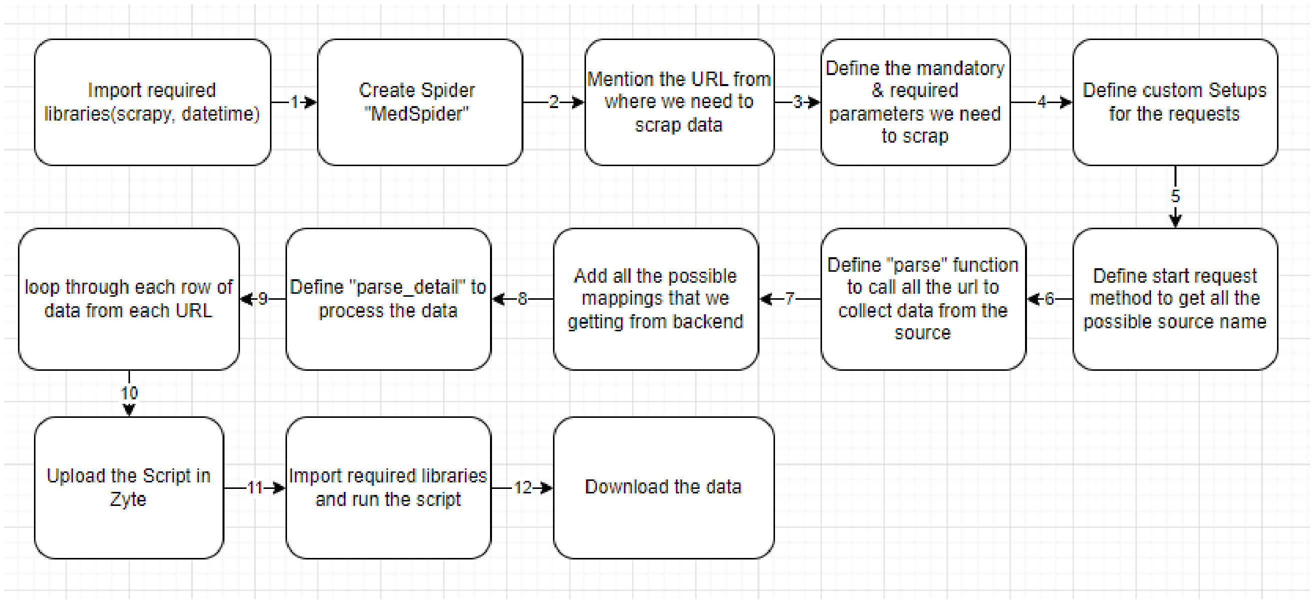
- The website has the various combinations of filters with which we need to scrap and merge the data together.
- We are creating URL's for each combination and mapping them with the data coming from the backend.
- The complete Dataset to be scrapped is vast which exceeds the free proxy limit, so we

are scraping a smaller subset of required data until we hit the max limit.
- We are getting blocked if we are going beyond the 40K approx request limit.
- There are 40 million possible combinations of logic to be incorporated in the code and iterate to fetch all the data.
- The collector code is built for all the combinations but we set a limit to scrap lesser data in-order to avoid getting blocked.
- We are getting blocked while running the collector code in Zyte.


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

3. Script Development Flow

Below steps are followed to create spider


Import required libraries(scrapy, datetime)
1
Create Spider "MedSpider"
2
Mention the URL from where we need to scrap data
3
Define the mandatory & required parameters we need to scrap
4
Define custom Setups for the requests
5
Define start request method to get all the possible source name
6
Define "parse" function to call all the url to collect data from the source
7
Add all the possible mappings that we getting from backend
8
Define "parse_detail" to process the data
9
loop through each row of data from each URL
10
Upload the Script in Zyte
11
Import required libraries and run the script
12
Download the data

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 - med.py

Here we are scraping the data as per the requirements

Step 1 - Importing required libraries

```
import scrapy
import json
import datetime
from os import environ
```

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

```
class MedSpider(scrapy.Spider):
    name = 'aeid4599_CMS'
    site = 'https://data.cms.gov/mapping-medicare-disparities'
    source_country = 'USA'
    context_identifier = ''
    file_create_dt = datetime.datetime.utcnow().strftime('%Y-%m-%d %T')[0:10]
    record_created_by = ""
    execution_id = ""  # This will be taken automatically from zyte, for now this is hardcoded
    feed_code = "aeid4599"
    t = 0
    row = 0
    e = []
    source_file = []
```

Step 3 - Here we are defining the custom details

```
custom_settings = {
        'ROBOTSTXT_OBEY': False,
        'RETRY_ENABLED': False,
        'CONCURRENT_REQUESTS': 256,
        'COOKIES_ENABLED': False,
        'COOKIES_DEBUG': False,
        "DOWNLOADER_MIDDLEWARES": {  # used for IP rotation
            'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware': None,
            'scrapy_user_agents.middlewares.RandomUserAgentMiddleware': 400,
        },
        'CONCURRENT_REQUESTS_PER_DOMAIN': 500,
        'DOWNLOAD_DELAY': 0,
        'AUTOTHROTTLE_ENABLED': True,
        'DOWNLOAD_TIMEOUT': 100,
        'DUPEFILTER_DEBUG': True,
    }
```

Step 4 - Starting request to get all the possible source names

```
    def start_requests(self):
        yield scrapy.Request(url='https://data.cms.gov/data-api/v1/mmd-tool/', callback=self.parse,
                             meta={'download_timeout': 100})
```

Step 5 - Function to call all the url to collect data from the source

```
def parse(self, response):
        source_li = json.loads(response.text)["_source"]
        print('source_li=========', type(source_li))
        source_ls = source_li[:]
```

Step 6 - Function to process data

```
    def parse_detail(self, response):
        item = MedicareDisparitiesItem()
```

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. **AEIDprojectId (Mandatory) -** We are capturing the hierarchy of product in a website
2. **row (Required) -** Adding indexing here from website**.**
3. **region_1 (Required) -** Capturing the state here from website.
4. **region_2 (Required) -** Capturing the county here from website.
5. **date_posted (Required)-** Capturing the year of the data.
6. **category (Required) -** Will capture this when Geography is county.
7. **category_2 (Required) -** This is hardcoded as "Population Health Measures ".
8. **category_3 (Required) -** Capturing the adjustment here from website.
9. **category_4 (Required) -** This is not capturing in data so excluded this from the template.
10. **category_5 (Required) -** This is not capturing in data so excluded this from the template.
11. **category_6 (Required) -** Capturing the condition service here from the website.
12. **gender (Required) -** Capturing the sex here from website.
13. **category_8 (Required) -** Capturing Race and Ethnicity here from the website.
14. **category_9 (Required) -** Capturing the dual Eligible here from website.
15. **category_10 (Required) -** Capturing age here from the website.
16. **metric (Required) -** Capturing the measure here from website.
17. **value (Required) -** Capturing the analysis value here.
18. **units (Required) -** Capturing the per beneficiary here from website.
19. **Record_create_by (Mandatory) -** This is hardcoded with spider name
20. **Record_create_dt (Mandatory) -** This is the timestamp for capturing the data.
21. **Site (Mandatory)-** This is hardcoded as "https://data.cms.gov/mapping-medicare-disparities".
22. **Source (Mandatory) -** This is hardcoded as "https://data.cms.gov/mapping-medicare-disparities".
23. **Source_country (Mandatory) -**This is hardcoded as "USA".

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.