Hardik Prabhu
241058019

1> Describe the Primary Purpose of User Agent header in web Scrapping.

→ * The User Agent header is an HTTP header used by a client (browser or bot) to identify itself to the server. When a server Recieve a request it can check the User-Agent String to determine what kind of device or software is making the Request. This helps to server tailor the content to different devices or browsers.

* In web scrapping setting a User-Agent header is essential to mimic a real browser, making your scraper look less likely to be blocked.

* So these are the primary purposes of User agent header,

2> What is the purpose of robots.txt file on a website.

→ The robots.txt file is a text file placed in the root directory of a website that provides guideline for web crawlers and bots about which parts of the site they are allowed to access. It uses the robots Exclusion Protocol & can include rules to restrict specific crawlers or path on the site.

Ex:-

User-agent: *

```
Disallow : /admin /
Disallow : /private -data /
```

In the above example, all bot (User-agent : *) are instructed not to access the /admin / and /private -data/ directories. While robots.txt is not a security factor

3) Scrape data contents from single web page using Beautiful Soup library.

→ Here the website used is http://quotes.toscrape.com/

```
from bs4 import BeautifulSoup
import requests
url = "http://quotes.tosrape.com/"
response = requests.get(url)
soup = BeautifulSoup (response.content ,"html.parser')
quotes = soup.find_all ("div", class_="quote")
for quote in quotes:
    text = quote.find ("span", class_="text").get_text()
    author = quote.find ("small", class_="author").get_text()
    print (f"Quote : {text}\nAuthor : {author}\n")
```

4) Scrape data contents navigating through multiple web pages using Beautiful Soup library.

→

```
from bs4 import BeautifulSoup
import Requests
base_url = 'http://quotes.tosrape.com/page/"
```

```
for page in range (1, 6):
    url = f"{base_url}{page}/"
    response = requests.get (url)
    soup = BeautifulSoup (response.content, "html.parser')
    quotes = soup.find_all ("div", class= "quote")
    for quote in quotes:
        text = quote.find ('span", class_="text").get_text()
        author = quote.find ("small", class_="author").get_text
        print (f"Page {page} - Quote : {text}\n Author : {author}\n"
```

5) Describe the Architecture of Scrappy:-

→ The Architecture of Scrapy has the following components

- *Scrappy Engine:-*

  The Engine is responsible for controlling the data flow between all components of the system. and triggering events when certain actions occur

- *Scheduler:-*

  The Scheduler receives requests from the engine and enqueues them for processing them later. when the Engine Requests them.

- *Downloader*

  The Downloader is Responsible for fetching web pages and feeding them to the Engine which in turn feeds them to the Spider.

* **Item pipeline:**

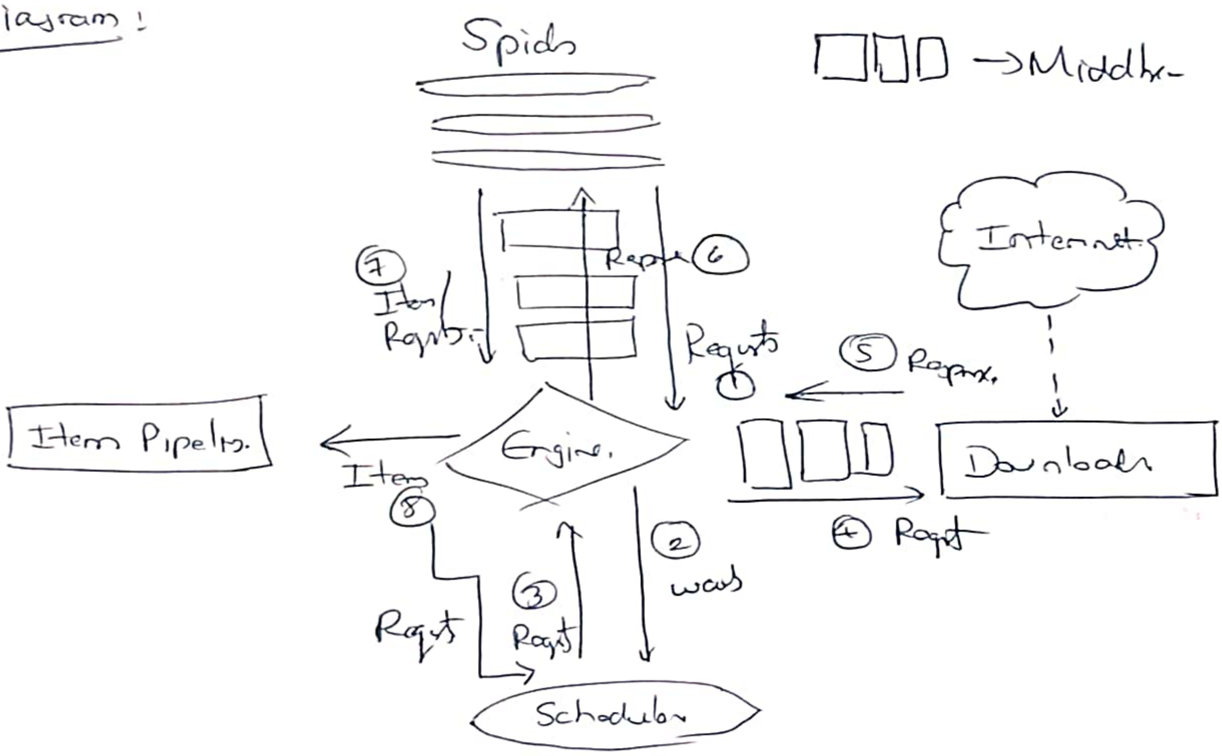
The item pipeline is responsible for process the items once they have been extracted by the spiders.

* **Middlewares :-**

- **Downloader middlewares :-** These are the specific hooks that are sitting between the Engine & Downloader.
- **Spider Middlewares :-**

These are the specific hooks that sit between Engine & the Spiders and are able to process Spider Input and output.

Diagram :



So this is the Architecture of the Scrapy framework used for web scraping.

6) Demonstrate Srapy framework for the provey
Example:-

```
import Srapy
class ProductSpider (Srapy.Spider):
    name : "product_spider"
    start_urls : ["http://www.spom_Tv.com"]
    def parse (self, response):
        for product in response.css ("product"):
            yield {
                "name" : product.css ("name ::text").get()
                "cost" : product.css ("cost ::text").get()
                "manufact" : product.css ("manufact ::text").get()
                "rating" : product.css ("rating ::text").get()
            }
```

Now we need to run the file by, `srapy runspider s/product_spider.py -o products.json.`

7) Create a pandas dataframe from Super market Invoice Data, & execute.

→ i) Loads

```
import pandas as pd.
df = pd.read_csv ('Supermarket-invoice-data.csv')
```

a) Count the number of customers per product category or department

```
customer_count = df.groupby ("Desription") ["Customer Code"].
                                                    nunique())
print (customer_count)
```

b) Calculate the average customer visit per product category

```
arg-visit = df. groupby ("Desription') ["Quantity"].mean()
```

c) Identify the product brought by the most custom

```
popular product = df.groupby ('Desrption' ]["Custom Code"].nunique.
                                                         idmax()
print (poplrproduct)
```

d) Sort Custom based on age, product Purchased, or product cart.

```
sorted_custm = df. sort_values (by = ["Sales", "Quantity"],
                         ascending = [False, False])
print (sorted_customers)
```

7) Explain Category of Exploratory visualization based on the Relationship between the necessary Players.

- Univariates
  - It shows the distribution of single variable
  - Visual examples
    - Histogram
    - Bar plots.
- Bivariate:-
  - Shows the Relationships between two variable
  - Visual Examples

- Scatter plots, box plots.

* Multivariate:
  - Shows relationships among multiple variables.
  - visb (e.g heatmaps, pair plots)

2) Define DAX, Demonstrate 10 DAX formula with context.

→ DAX which could be called as Data Analytics Expression is a formula language for data calculation in Power BI, SSAS & Excel Power Pivot

- Sum.

  TotalSales = SUM (Sales [Amount])

- Average.

  Avg = AVERAGE (Sales [Amount])

- Count

  Count = COUNT (Sales [CustomerID])

- Count Rows.

  CountRows => Total Rows = COUNTROWS (Sales)

- Min

  Min Sales = MIN (Sales [Amount])

- Max

  Max Sales = MAX (Sales [Amount])

- RELATED

  Related Category = RELATED (Products [Category])

- IF

  Sales Status = IF (Sales [Amount] > 500, "High", "Low")

- Calculate

  High Sales = CALCULATE (SUM (Sales [Amt), Sales [Amount]) > 100)

- FILTER

Filtered tbl = FILTER (Sls , Sls [Amt]>500)

3) Differns between Calculated Meas & Calculated Column.

Calculated Measure	Calculated Column.
* Used for Aggregation & Dynamic Calculations that chg based on filts & report Context	* Used to create additional columns with static vals in a table, calculs row by row
- Computed dynamically based on the reports filts cont	* Calculd once durm data fresh, vals do not chg with filts
* Stord as a formula in the model & calculates on demand	- Stord as actl vals in the data model.
* Cannot be placed directly	* Can be added directly
- Calculated at run time	* Pre calculated.
* Typically Faster	* Remains Constant until the next refresh.
* Calculated based on the Entire Dataset.	* Limited to row level Context
* Uses for total Sls avrs metrics etc.	* Used for Creating categoes and more
* Uses functions like SUM() AVERAGE() etc.	* Often use IF(), CONCATENATE() etc
* Can be used in Measure, cards KPI's et	* Can be used in tables, matrics etc.