**QA Tech Challenges**

Candidate Name: Pavan Pamireddy

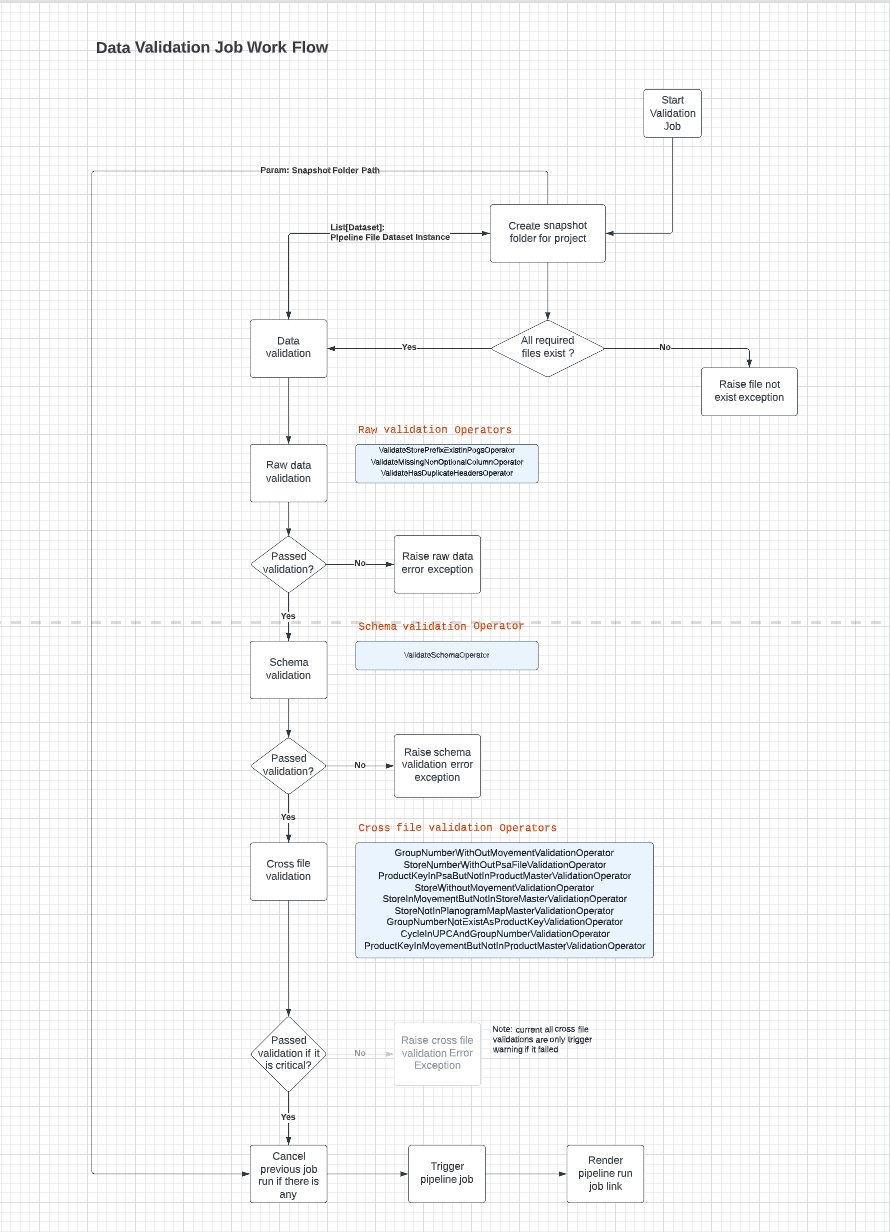
Date: 09/03/22

**Scenario Based Question 1 Data Validation:**

**Briefing:**

Hivery has a machine learning product **Curate** built for assortment and space awareness solution widely used by the retail industry for the category management. Customer data will go through ETL and data validation process using Databricks before the source files be taken to Curate for project installation, scenario creation, rules creation then optimization.

The data validation workflow follows below process to run Raw file validation, Schema validation and Cross File validation:

  
a. Based on above diagram, please design some high-level test scenarios to make sure all validation processes are covered.

[Pavan] -

1. Verify if the validation job is started/in queue/submitted
2. Verify snapshot folder path and project name
3. Verify if all data files exist in the dataset
4. Verify file not exist exception is reported in case data files are missing
5. Validate Raw data for each of Raw data operators conditions –storeprefixExistingpogs operator, Missingnonoptionalcoloumn operator,hasduplicateheader operator
6. Verify Rawdata error exception is reported in case any of the Rawdata operations failed
7. Verify validate schema operator condition
8. Verify schema validation error exception is reported in case any of the Rawdata operations failed
9. Verify cross file validation conditions for each of all 9 operator conditions
10. Verify if a trigged warning is reported when cross file validation is failed.
11. Verify of any previous job existed/running.
12. Verify if job is cancelled if any of the previous job existed and running.
13. Verify if pipeline job is triggered.
14. Verify rendered pipeline run link and check accessibility of the contents in the link.
15. Verify if the job output is stored in the snapshot folder path.

b. One of the Databricks job run failed with below Raw Data validation. The source file is saved at Azure Storage. Please describe the steps to fix the error and rerun the validation process.

Graphical user interface, text, application

Description automatically generated

[Pavan] There are 2 exceptions caught – One is the Rawdata validation error and Schema validation error. Below fix has to be applied for each type of exceptions raised –

Rawdata validation error – Add non option column store\_number. This should have been reported because the condition Missingnonoptionalcoloumn operator

Schema Validation error –

* The column “weekly\_revenue” data type is set to accept floating point numbers, but it contains characters. Fix the column values to have only floating point numbers.
* The column “weeks\_sold” accepts an int data type but have strings. Fix the column to have only integers.
* Lastly, UPC column is constrained by Not nullable value and it has a null value. Remove null value

Fix all above and re-run the job in the data and rerun the job.

**Scenario Based Question 2 UI Functional Testing:**

Curate has a web interface where a user can create a project based on uploading various of source files. By accessing a specific project, the user is directed to Project Page (screen shot shown below) where 2 foundational scenarios (Core & Base) have been initially generated as a starting point. The user can then create additional scenarios with the Core as a foundation by clicking the ‘+ New Scenario From Core’ button. Each scenario can be used to simulate how results are impacted when new rules are applied. They then can compare these results to the Base Scenario (which acts as a reference point to what occurred last year – as supplied via the source files).   
From each scenario (displayed as a card on this page), you can see some key scenario information available:

* Scenario name
* KPI indicators (i.e. PODs, Revenue/W)
* Last Modified
* Goal description
* Optimization Status

And some actions applied to the specific scenario card:

* Pin scenario as favourite (The star icon)
* Clone scenario (The copy icon)
* Archive scenario (The red box icon)

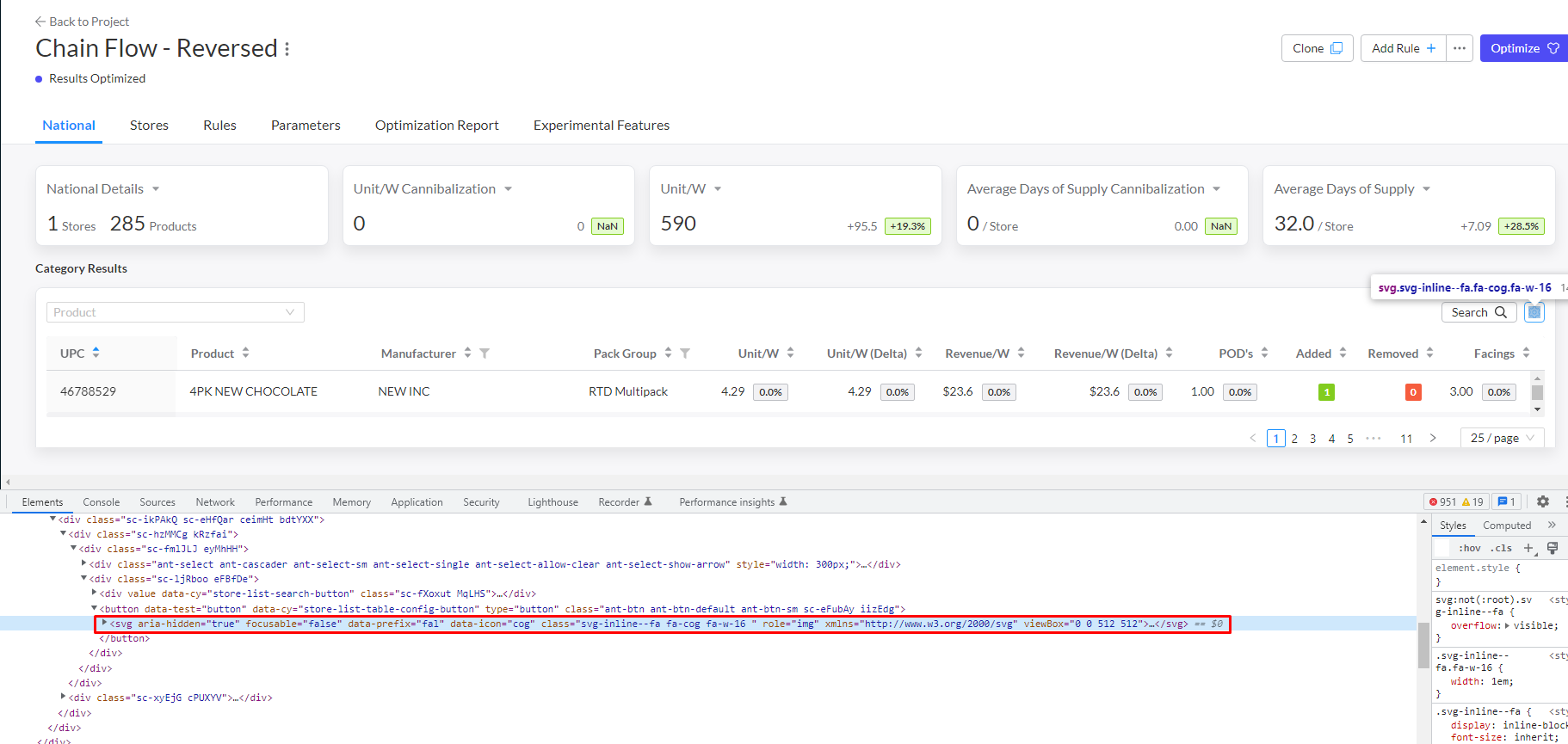
Based on the descriptions above and screen shot below, please draft some high level test cases to test the project page.

Graphical user interface, application, website

Description automatically generated

[Pavan] –

1. Verify importing a scenario from import scenario button
2. Verify search functionality
3. Verify if scenario is imported to right category based on scenario (Core/Base)
4. Verify adding core scenario using New scenario from core
5. Verify defining goals and it is updated on the page
6. Verify last modified data
7. Verify selecting a start to a scenario and it is highlighted
8. Verify copy and pasting scenario/scenario data using the copy button
9. Verify if only starred scenarios are displayed when favourites are selected.
10. Verify is scenarios are sorted in alphabetical order when selected.
11. Verify results optimized status colour indicator is appropriate as the status
12. Verify last update date/time
13. Verify display of number of products
14. Verify display of number of stores
15. Verify Archived scenarios using Archived button
16. Verify is scenario type is displayed for a scenario
17. Verify POD’s and Revenue/w numbers are displayed
18. Verify if %growth is displayed for POD’s and Revenue/w
19. Verify if Assortment and Space text is displayed
20. Verify Compare scenarios using compare scenarios button.
21. Verify if selected store is displayed

**Scenario Based Question 3 Test Automation:**  
  
**Question 1:** Get xpath of this tag as bellow image  
  


[Pavan] - //div[@class=’sc-ikPAkQ sc-eHfQar ceimHt bdtYXX’]//svg[@class=’svg-inline—fa fa-cog fa-w-16’ and @role=’img’]

Note that this optimum xpath could change based on number of similar elements on the DOM or if I could access the complete DOM.

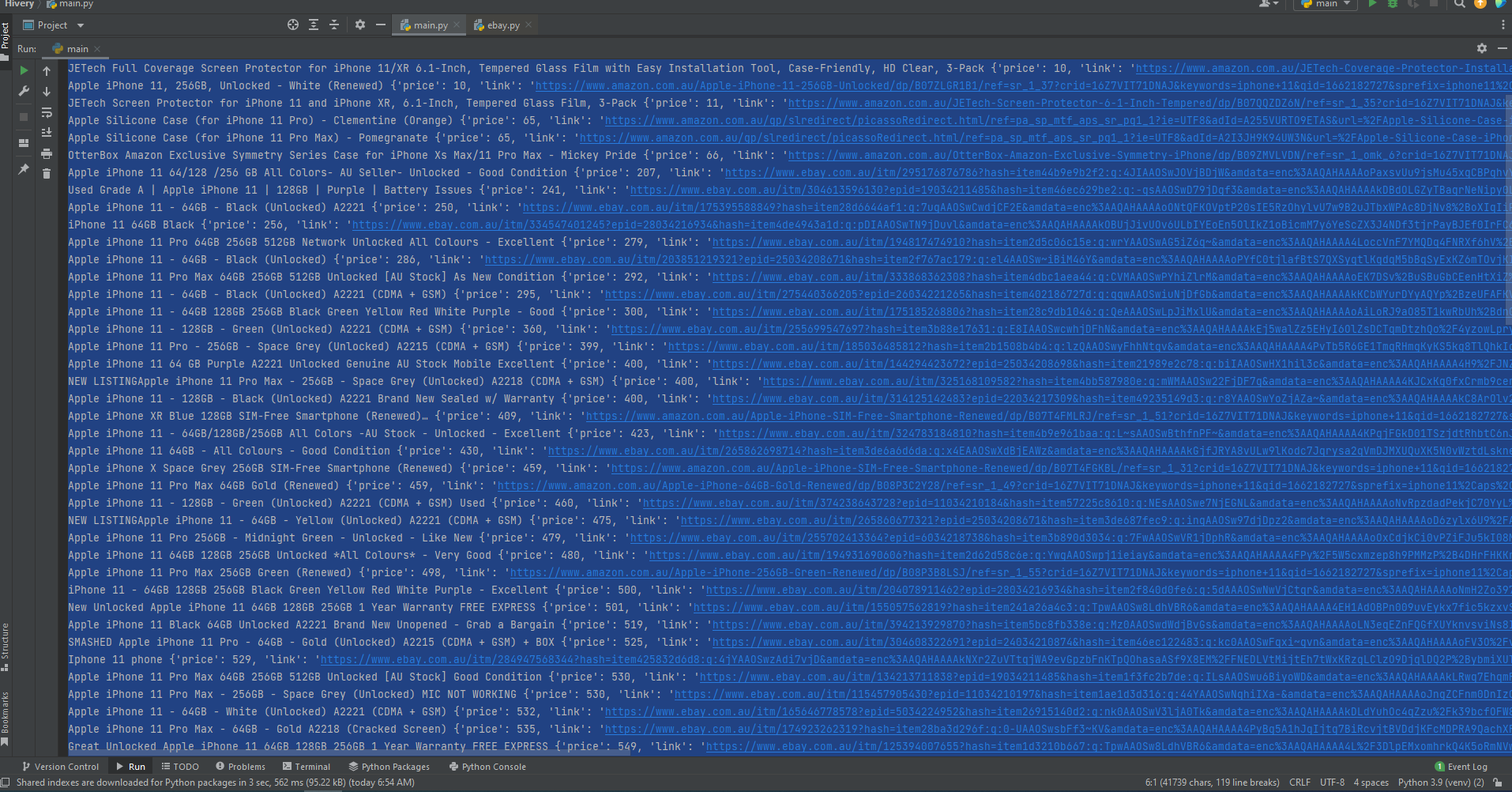
**Question 2:** Web automation task. Use selenium and any model to build project.  
  
1. Open two ecommerce websites say amazon.com and ebay.com

2. Search for "iPhone11" on the website and validate that the result is shown for the mentioned product

3. Now, combine the outputs of both the websites and display the result in ascending order of price.

4. The output should consist of the following info: Name of the Website, Name of the product, Price of the product and Link to the product

5. You can print the output in any reporter of your choice or even in the console.



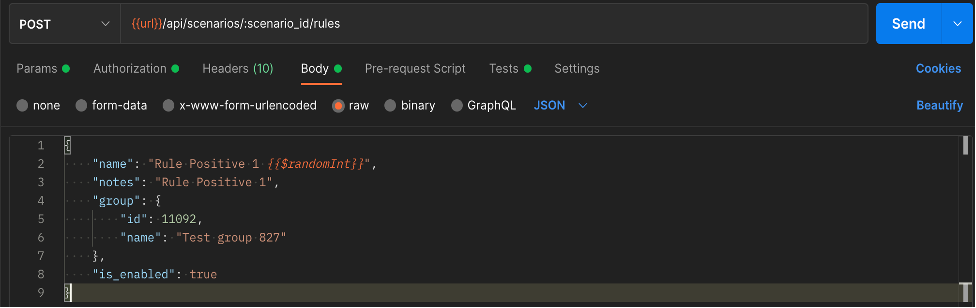
****

**Scenario Based Question 4 API Testing:**

**Question 1:** In the screenshot below, you can see the API to create a new rule in Hivery’s system. All the parameters are required field.   
Now we need to set up API Testing in POSTMAN to make sure the server returns validation error whenever any parameter is missing.  
Normally, since there are 5 parameters, we would set up 5 separate tests for the job.

Is it possible to achieve the same goal with just 1 single test? If yes, please provide your detail solution using POSTMAN.

Hint: utilize Pre-request script & Tests



[Pavan]

On the test tab, we are going to create a schema, of what is expected from a json response. We define those as required so that whenever a request is been sent out with missing parameters, the test would fail as the required parameters are missing. Secondly when there are multiple required parameters missing, the validation stops at first error. It is possible to find the list of all missing parameters using “tv4.validateMultiple” which would return an array of missing parameters.

const schema = {

"$schema": "https://json-schema.org/draft/2019-09/schema",

"$id": "http://example.com/example.json",

"type": "object",

"default": {},

"title": "Root Schema",

"required": [

"name",

"notes",

"group",

"is\_enabled"

],

"properties": {

"name": {

"type": "string",

"default": "",

"title": "The name Schema",

"examples": [

"Rule Positive 1 {{$randomInt}}"

]

},

"notes": {

"type": "string",

"default": "",

"title": "The notes Schema",

"examples": [

"Rule Positive 1"

]

},

"group": {

"type": "object",

"default": {},

"title": "The group Schema",

"required": [

"id",

"name"

],

"properties": {

"id": {

"type": "integer",

"default": 0,

"title": "The id Schema",

"examples": [

11092

]

},

"name": {

"type": "string",

"default": "",

"title": "The name Schema",

"examples": [

"Test group 827"

]

}

},

"examples": [{

"id": 11092,

"name": "Test group 827"

}]

},

"is\_enabled": {

"type": "boolean",

"default": false,

"title": "The is\_enabled Schema",

"examples": [

true

]

}

},

"examples": [{

"name": "Rule Positive 1 {{$randomInt}}",

"notes": "Rule Positive 1",

"group": {

"id": 11092,

"name": "Test group 827"

},

"is\_enabled": true

}]

}

Test to validate it –

pm.test(“Validating missing required inputs”, function() {

var result = tv4.validateMultiple(pm.response.json, schema)

pm.expect(result.valid).to.be.true;

}

**SQL Challenge:**

Graphical user interface, application, table, Excel

Description automatically generated

Given the schema above (with examples of data), please write queries to get the following data:

**Questions 1:** Number of active rules per customer

[Pavan]

Select cg.name, count(r.id) as "Active Rules"

From Customer groups cg

Join Projects p on cg.id=p.customer\_group\_id

Join Scenarios s on p.id = s.project\_id

Join Rules r on s.id = r.scenario\_id

where r.is\_deleted =’t’;

**Questions 2:** Customers that have rules with PODs count > 1000000 (1m)

[PAVAN]

Select cg.name

From Customer groups cg

Join Projects p on cg.id=p.customer\_group\_id

Join Scenarios s on p.id = s.project\_id

Join Rules r on s.id = r.scenario\_id

where r.is\_deleted =’t’

and r.pods\_count > 1000000;