Hello and welcome to TC25!

This is your lab environment for Build, Automate, and Scale: A Tableau REST API Deep Dive

First we'll need to do a bit of setup.

To log into your VM, you will need to use the following credentials:

Username: **LabUser**

Password: **Pa$$w0rd**

The username and password applies only to the VM. In this HOT Session, we will be using Tableau Cloud. When signing into Tableau Cloud you will utilize the following credentials:

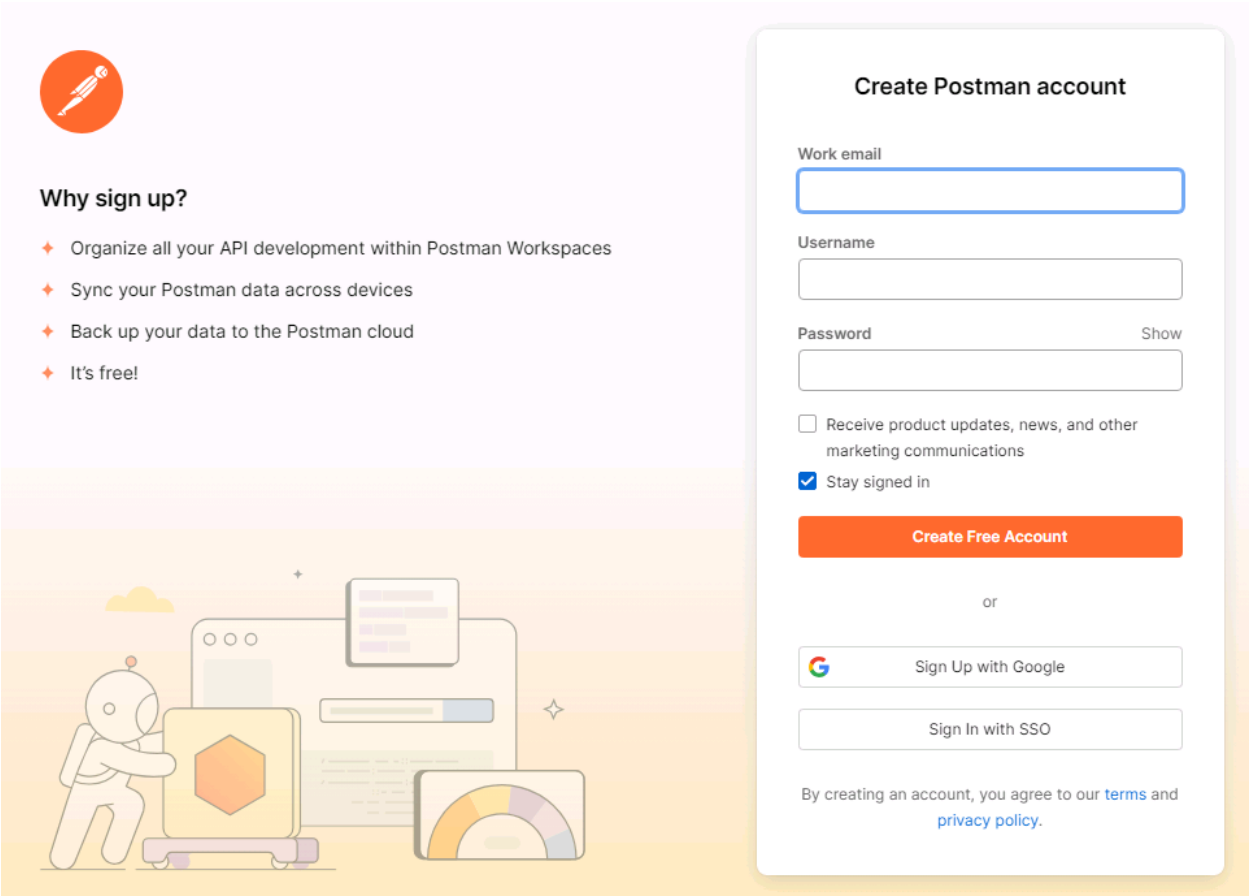
Username: @lab.CloudCredential(TC25HOT1942).Username

Password: @lab.CloudCredential(TC25HOT1942).Password

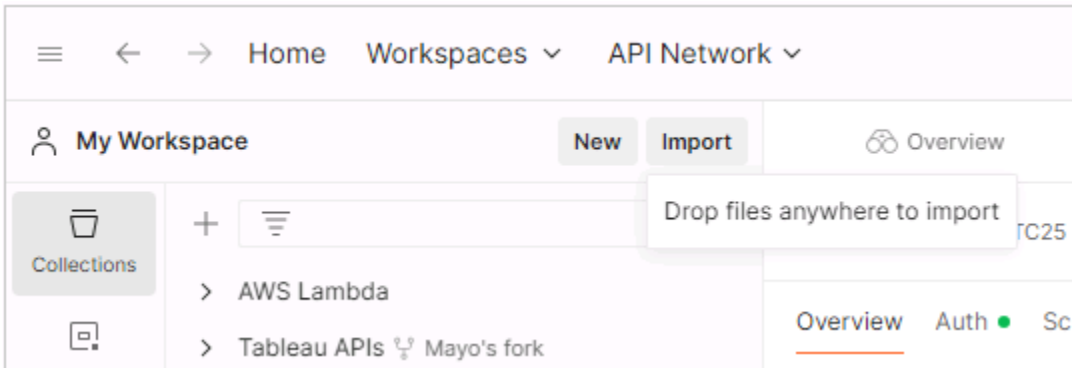
Let us know if you need assistance with developing your lab. The Room Assistants are here to help!

Exercise #1a Environment Setup & Authentication

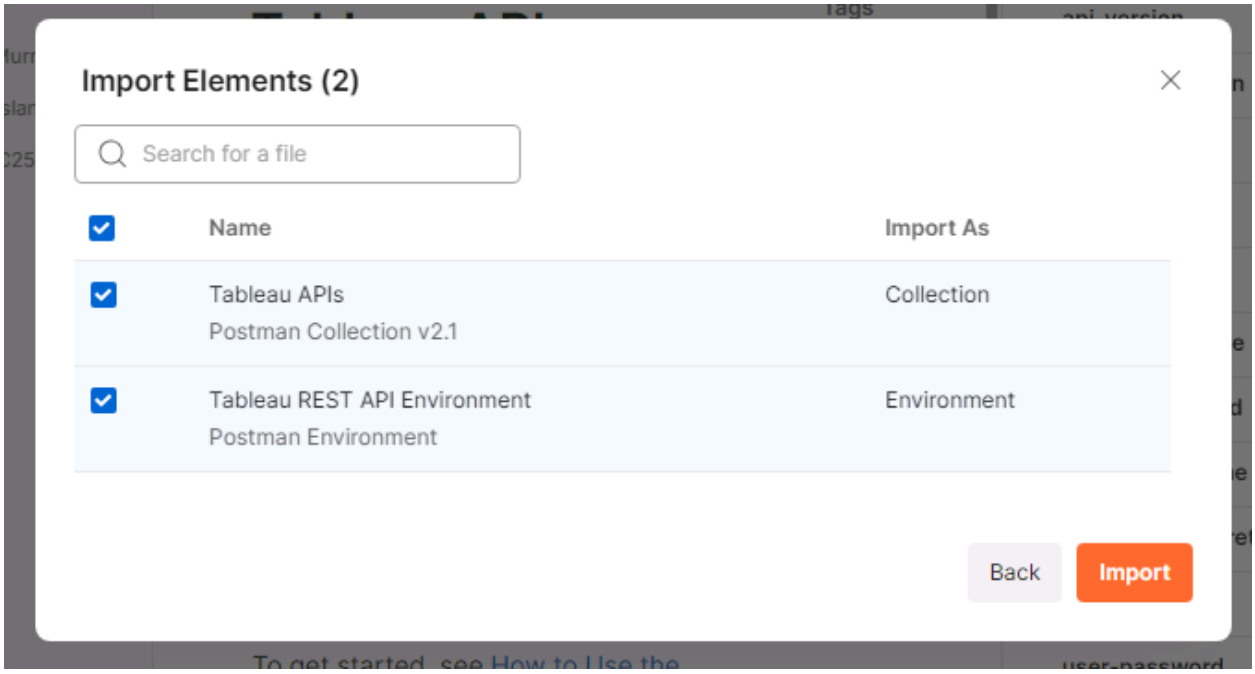
1. Sign into the VM with the credentials provided if not already.
2. Today we will be using Postman, an API platform for building, testing and using APIs. You will need a free account setup with Postman. If you already have an account, skip to Step 6.
3. Go ahead and open up the Postman application by double clicking the Postman icon (insert logo) on the desktop.
4. Click the Create Account button in the upper right. This will open up a browser window where you can create a quick account as well as sign in with Google if preferred.



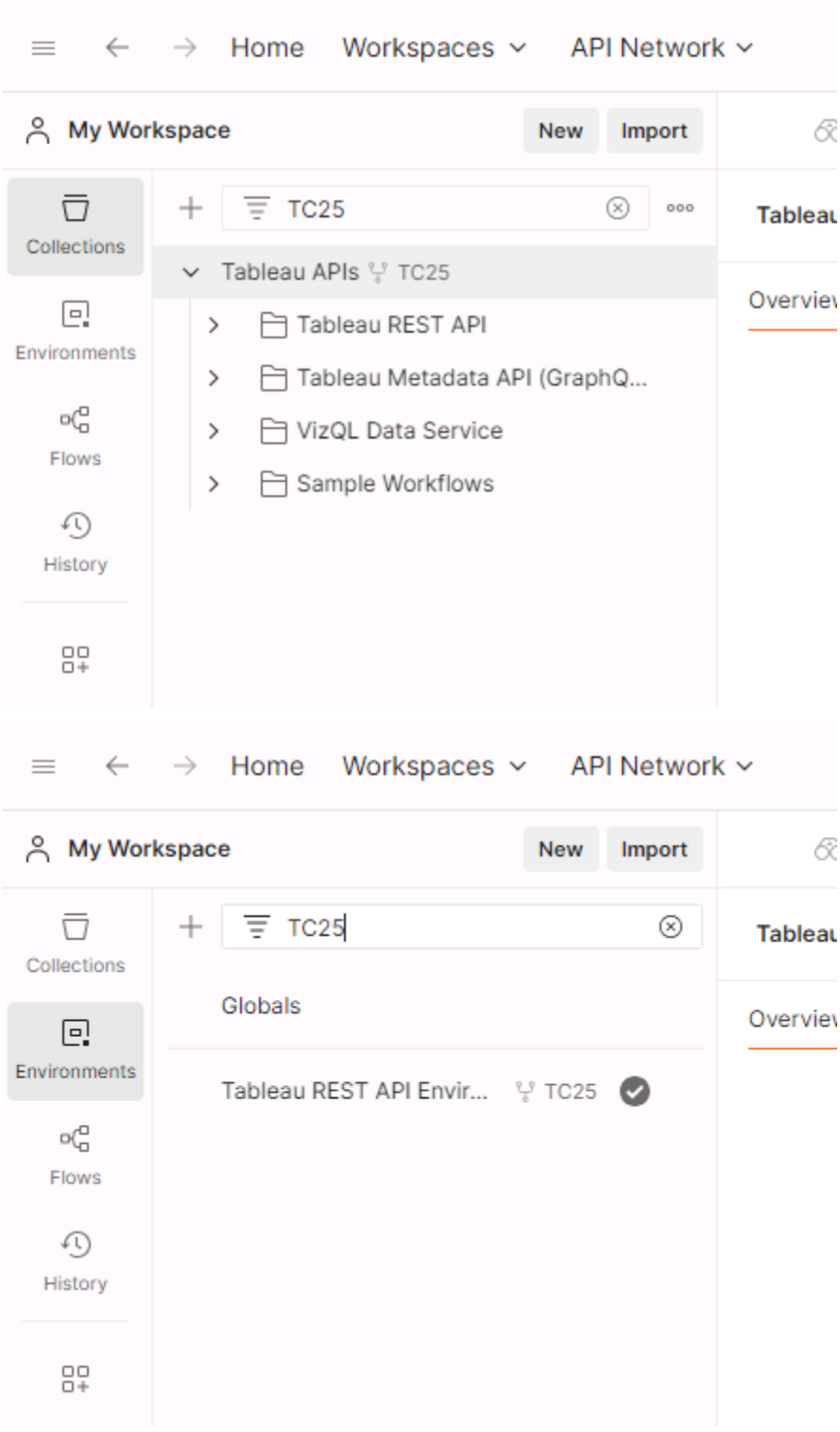
5. Once you've created your account, go back to the Postman application on the desktop and sign-in using your credentials. Strangely, this re-opens a browser tab to sign-in, only to route you back to the application. 🤦
6. Now that we're signed in, we can go get our Tableau REST API Collection. On the Desktop, you will see a folder called Lab Files. Go ahead and open it up. There should be 2 files.
    a. TC25_Tableau APIs.postman_collection.json
    b. TC25_Tableau REST API Environment.postman_environment.json
7. We first need to import both the collection and the environment file to Postman. Click the **Import** button next to **My Workspace**.



8. Drag both files from the **Lab Files** folder to the import window in Postman. **Click Import.**

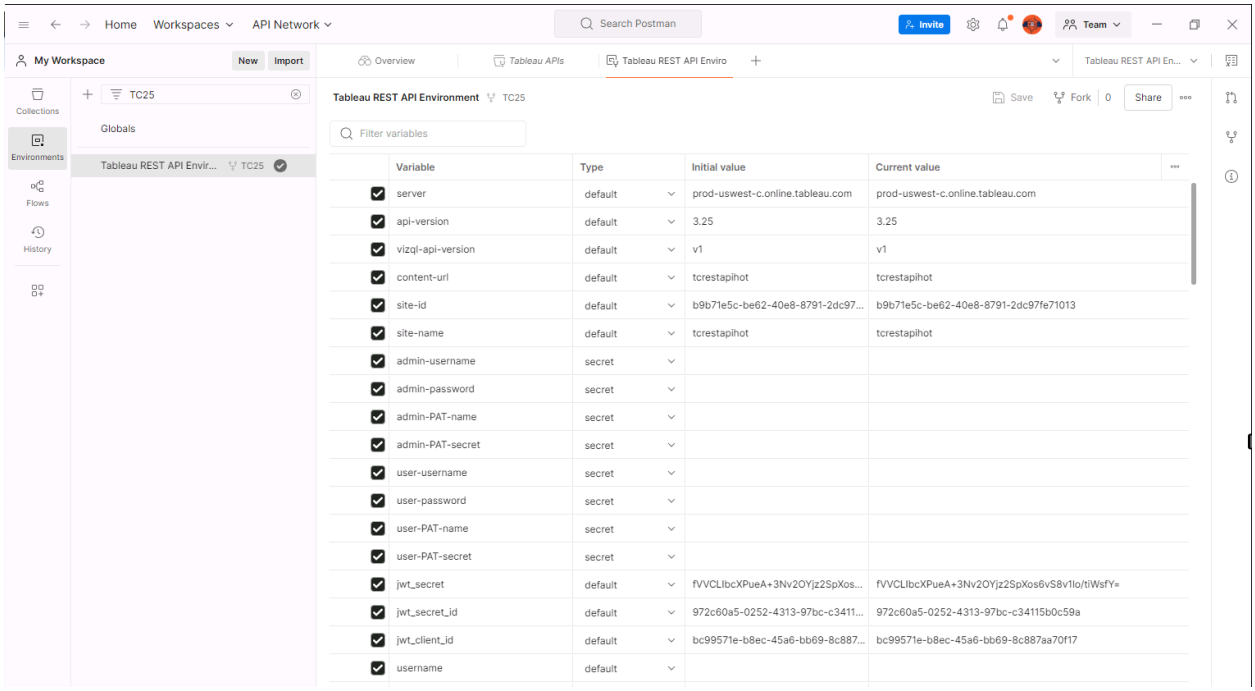


9. You should now have both the Collection and Environment in your Postman application.

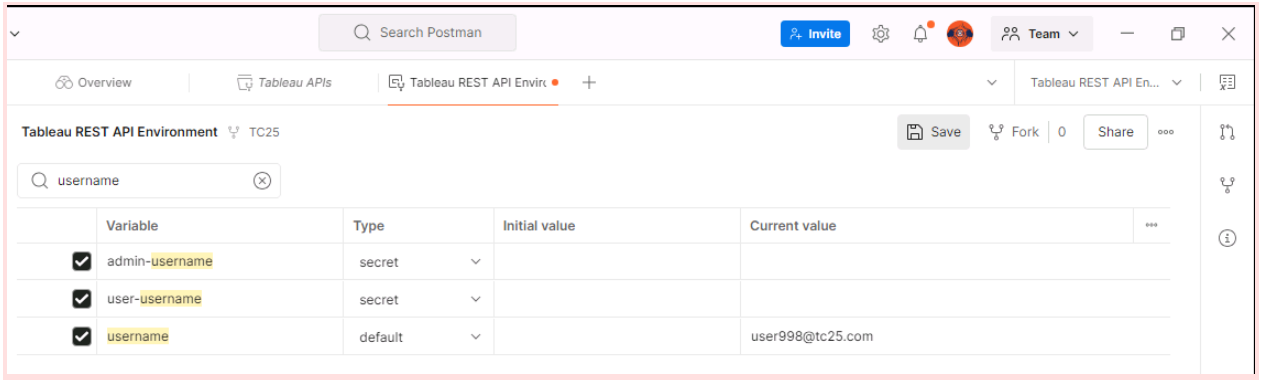


10. Go ahead and click on Environments and select the Tableau REST API Environment collection (if not already selected). There should be a number of variables already loaded into your Environment file. We only need to add one

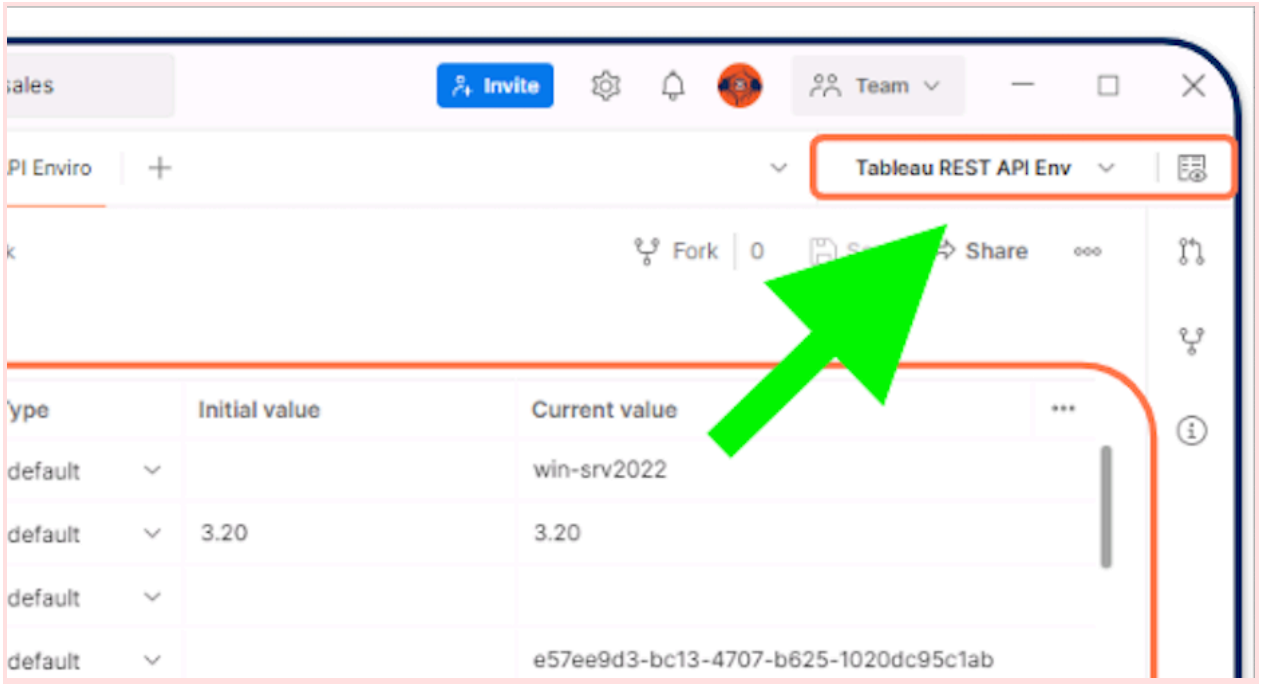
more… YOUR USERNAME.



11. Place your cursor in the cell for Current Value for the `username` variable and then from the top of the instruction click the Username: @lab.CloudCredential(TC25HOT1942).Username to paste your assigned user email address into the environment. Click the Save button to save your new variable entry.

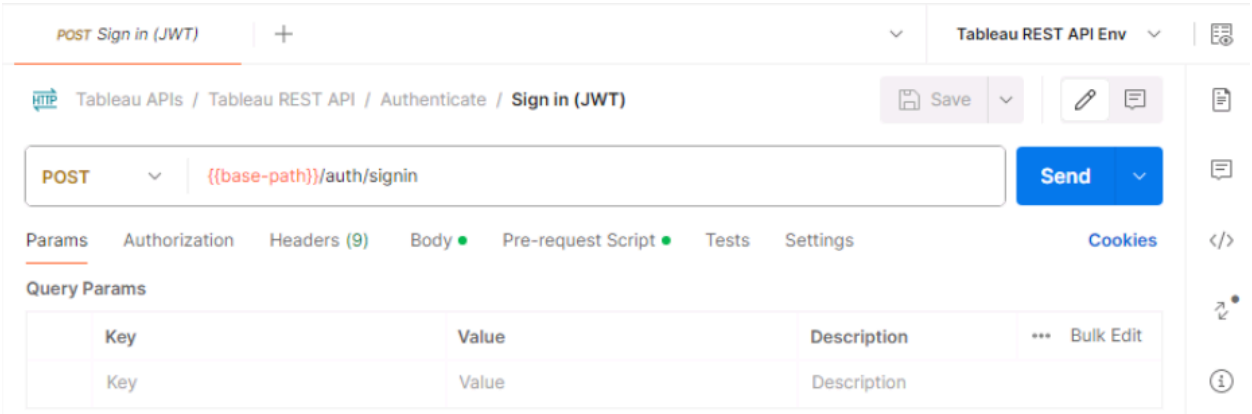


⚠️ IMPORTANT!! Don't forget to select your environment in the upper right corner. Failing to do so will cause unnecessary mental anguish.



***The Collection also has some variables to leverage when working with the Tableau REST API. We don't need to modify any of the variables from the Collection before we can get started. But if you're curious about what {{base-path}} is or your {{api-key}} is set you can click on **Collections->Tableau APIs** then select the **Variables** tab to view those.

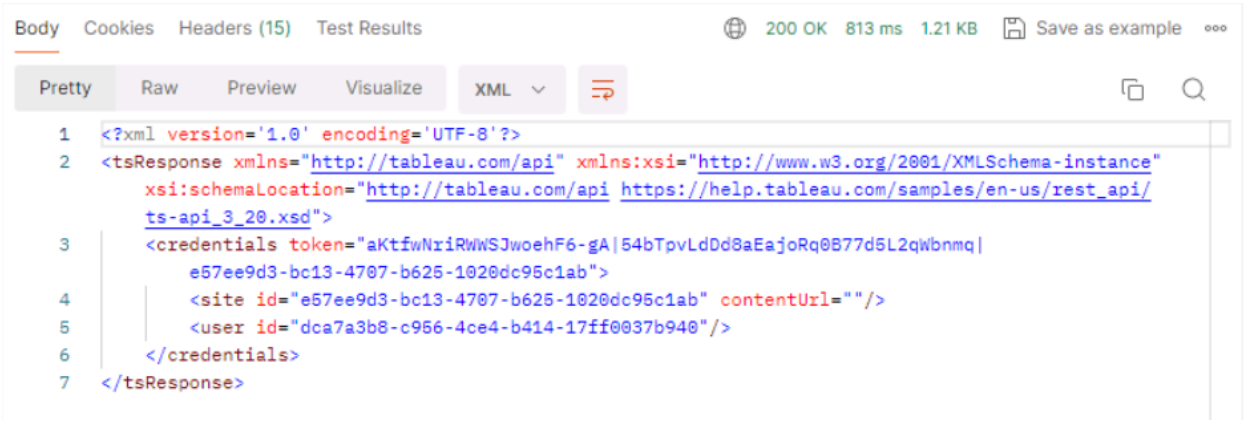
12. Now let's test some **Authentication** requests. Find the **Authentication** folder under the Tableau REST API section of the API collection. Click on the **Sign in (JWT)** API request. You'll notice the **Body** and **Pre-request Script** sections under the URI have a green dot 🟢. This indicates additional information that needs our attention.



13. Most of the API calls we make will have information in the **Body** to send with the request. Here we're sending a JSON Web Token `{{jwt}}` being generated in the **Pre-request Script** as our credentials. Additionally, we're sending the variable value for `{{content-url}}`, which is the equivalent of Site name.

```
<tsRequest>
    <credentials jwt="{{jwt}}">
        <site contentUrl="{{content-url}}" />
    </credentials>
</tsRequest>
```

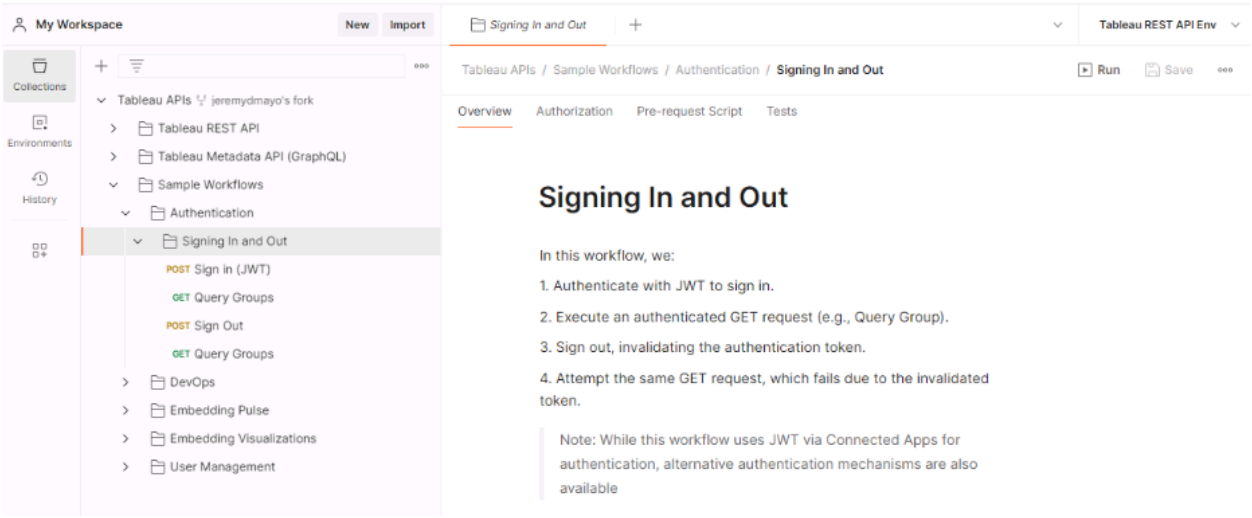
14. Click the Send button. You should receive a response that looks similar to the image below. Notice the credentials token. This token is your `{{api-key}}` to each subsequent call, until a **Sign-out** request is made or the token expires.

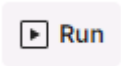


Exercise #1b - Authentication Workflow

1. Postman has a really great feature that allows you to run a complete test of APIs in a sequence. We call them **Workflows** and have included a few samples in our **Collection**. Find the **Sample Workflows** folder at the bottom of the **Tableau APIs** Collection. Under **Authentication**, there's a **Signing In and Out**

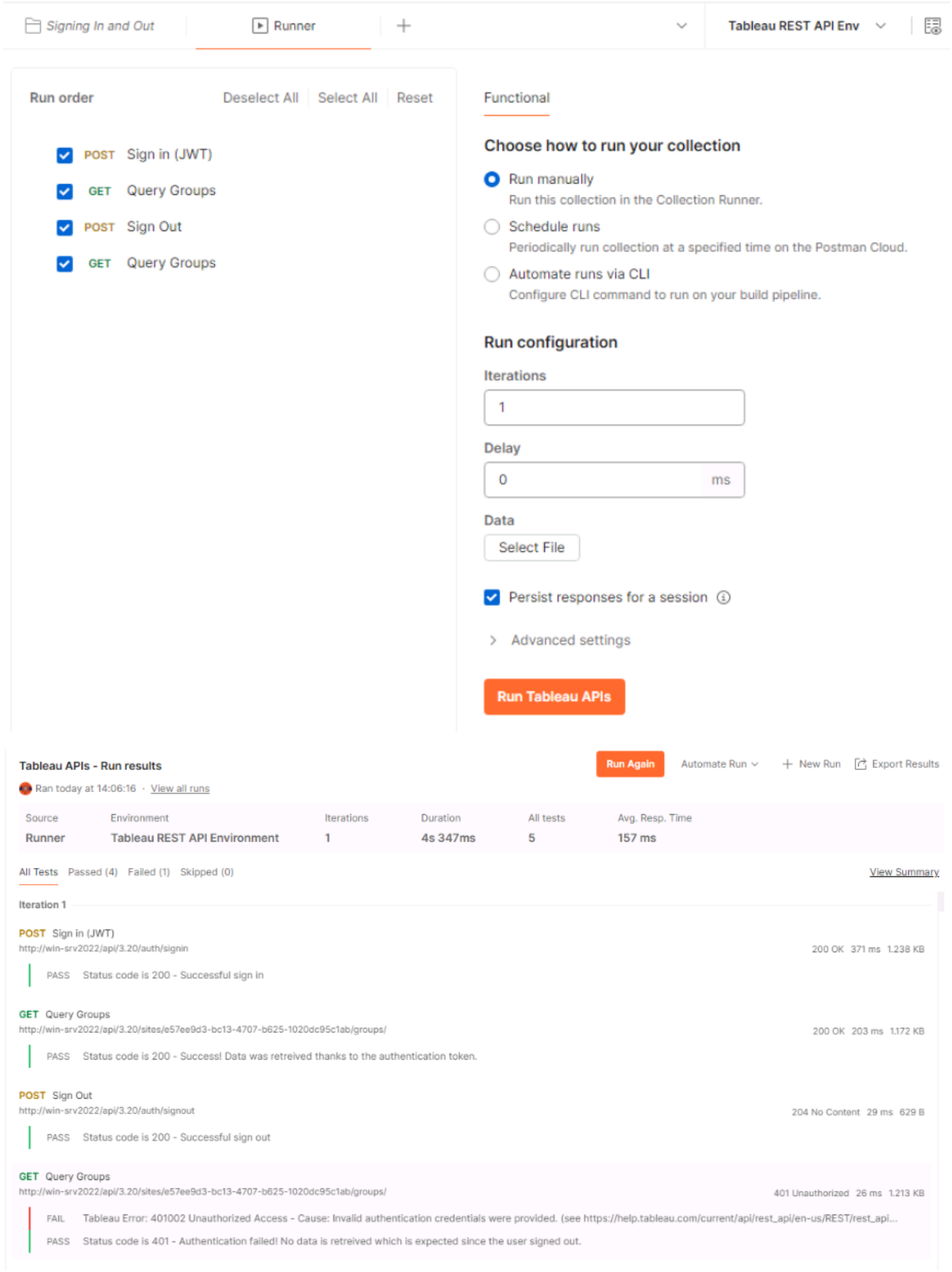
workflow. Select that and you will see all of the calls that make up the workflow.



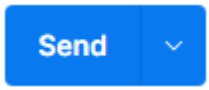
2. Click the 

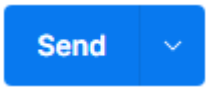
button to configure the workflow. Select the checkbox ☑️ to **Persist responses for a session** and leave the rest as default. Click 

and watch the magic happen. 🪄🎩

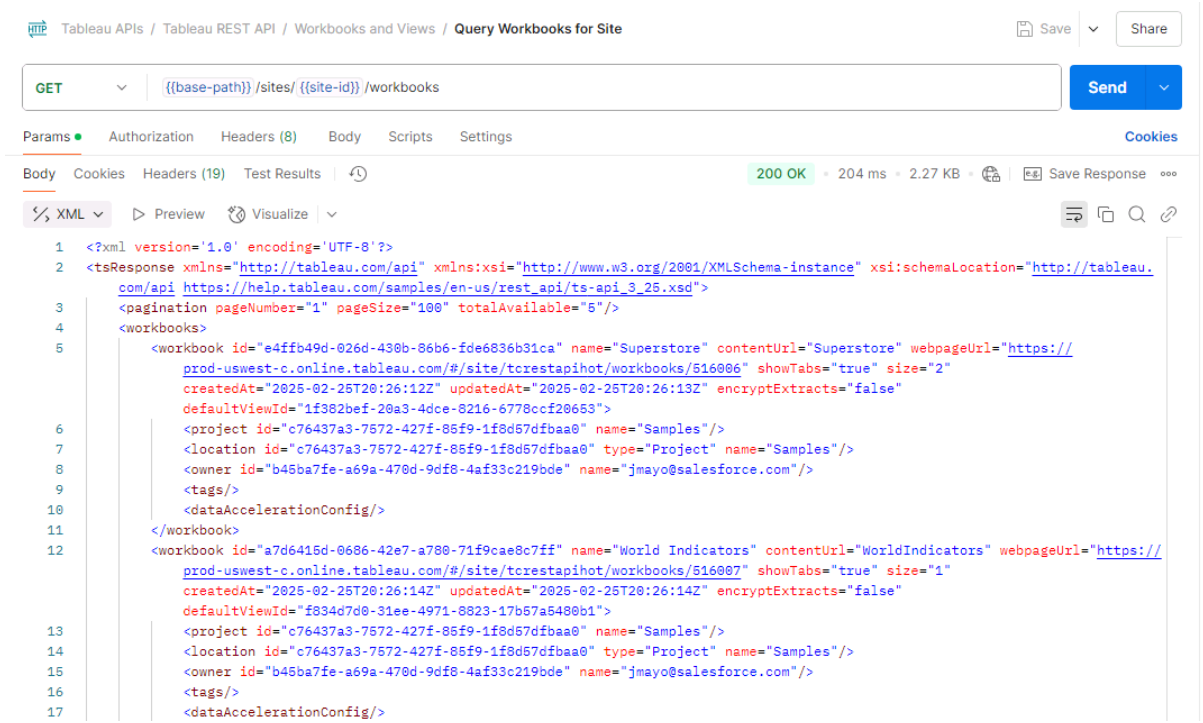

Signing In and Out
Runner
Tableau REST API Env
Run order
Deselect All
Select All
Reset
POST Sign in (JWT)
GET Query Groups
POST Sign Out
GET Query Groups
Functional
Choose how to run your collection
Run manually
Run this collection in the Collection Runner.
Schedule runs
Periodically run collection at a specified time on the Postman Cloud.
Automate runs via CLI
Configure CLI command to run on your build pipeline.
Run configuration
Iterations
1
Delay
0
ms
Data
Select File
Persist responses for a session
Advanced settings
Run Tableau APIs
Tableau APIs - Run results
Run Again
Automate Run
New Run
Export Results
Ran today at 14:06:16 · View all runs
Source
Runner
Environment
Tableau REST API Environment
Iterations
1
Duration
4s 347ms
All tests
5
Avg. Resp. Time
157 ms
All Tests
Passed (4)
Failed (1)
Skipped (0)
View Summary
Iteration 1
POST Sign in (JWT)
http://win-srv2022/api/3.20/auth/signin
200 OK 371 ms 1.238 KB
PASS Status code is 200 - Successful sign in
GET Query Groups
http://win-srv2022/api/3.20/sites/e57ee9d3-bc13-4707-b625-1020dc95c1ab/groups/
200 OK 203 ms 1.172 KB
PASS Status code is 200 - Success! Data was retreived thanks to the authentication token.
POST Sign Out
http://win-srv2022/api/3.20/auth/signout
204 No Content 29 ms 629 B
PASS Status code is 200 - Successful sign out
GET Query Groups
http://win-srv2022/api/3.20/sites/e57ee9d3-bc13-4707-b625-1020dc95c1ab/groups/
401 Unauthorized 26 ms 1.213 KB
FAIL Tableau Error: 401002 Unauthorized Access - Cause: Invalid authentication credentials were provided. (see https://help.tableau.com/current/api/rest_api/en-us/REST/rest_api...
PASS Status code is 401 - Authentication failed! No data is retreived which is expected since the user signed out.

Exercise #2a - Working with Variables

1. Let's **find a workbook** and see what variables are available. First we need to get a new **authentication** token.
2. Find and open the request to Sign Out and click 

. This clears the `{{api-key}}` token.
3. Now let's re-authenticate by hitting the request to **Sign In (JWT)** and get a new token 🔑.
4. Under the Tableau REST APIs folder, scroll to **Workbooks and Views**. Open the **Query Workbooks for Site** API request. Go ahead and click 

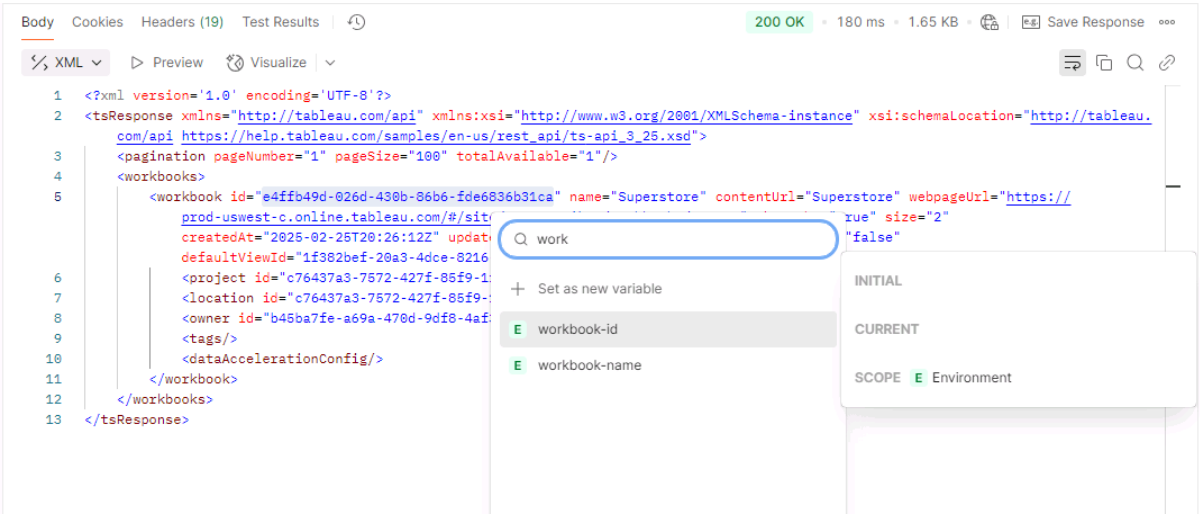
.
5. With a successful response ✅ you should also get all of the workbooks on the site.



6. This time let's do the same but narrow the search using a filter expression. Click on the **Params** tab and check the box next to `filter`. Under the Value (where it says filter-expression) let's add an expression to filter the search results to a workbook named **Superstore**, and owned by jmayo@salesforce.com. You can separate the expressions by a single comma.
`ownerEmail:eq:jmayo@salesforce.com,name:eq:Superstore`

7. This time your response should be limited to just the one workbook named **Superstore** owned by jmayo@salesforce.com.

8. Since we're going to need the workbook id in future calls, let's go ahead and grab that id now. Select the id (between the quotes) and right click 🖱. Select the option to **Set as Variable** and type workbook to locate `{{workbook-id}}`.

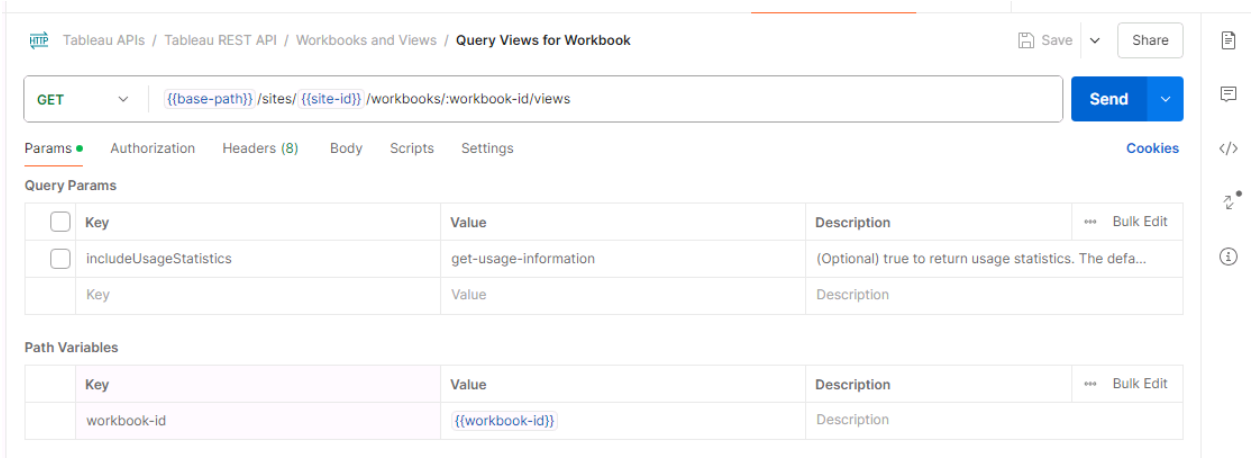


9. Now that we have our `{{workbook-id}}` we can go get some more info about our workbook in a later exercise.

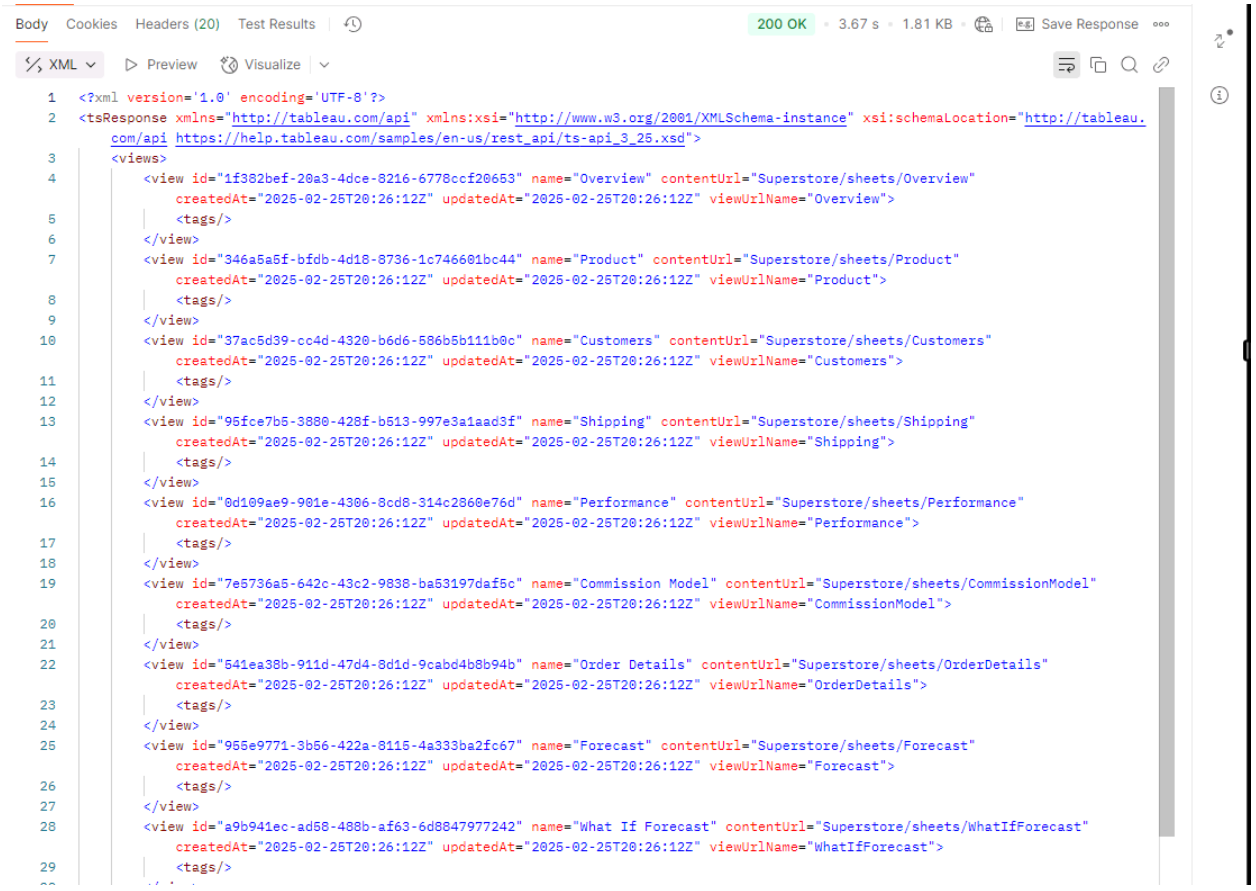
Exercise #3 - Requesting Images, PDFs & Data

1. Sticking with **Workbooks and Views**, open the **Query Views for Workbook** API request. On the **Params** tab, notice under **Path Variables** we're going to

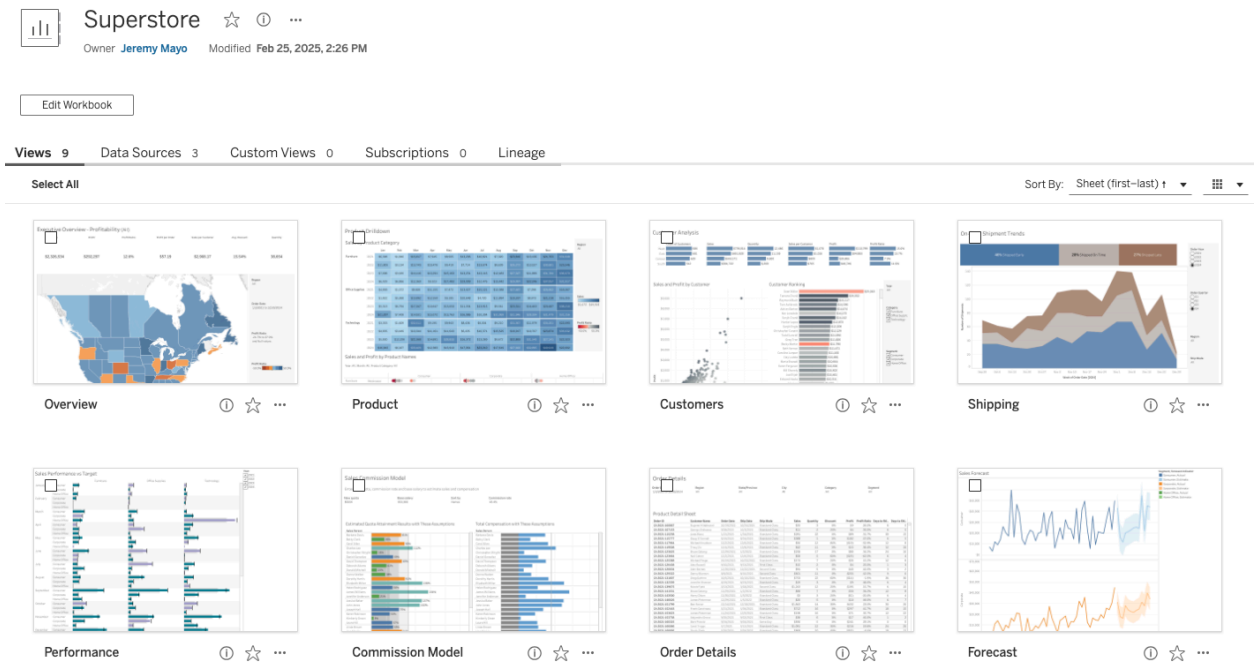
use the `{{workbook-id}}` we just grabbed, only this time we're getting a detailed list of the views that make up this workbook. Go ahead and click Send.



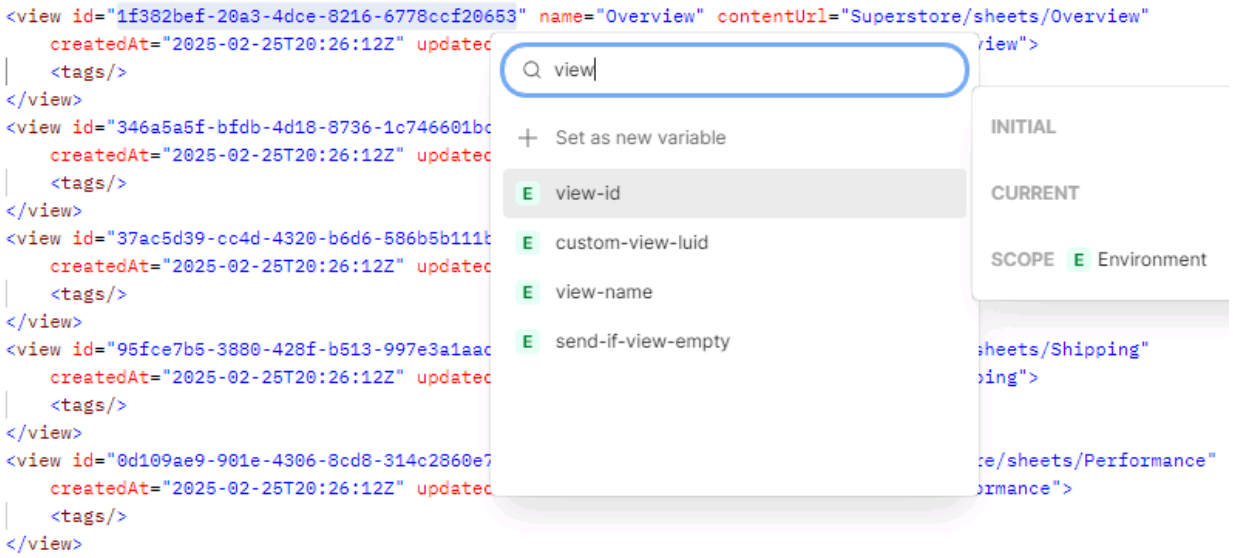
2. There we go! Now we have all the view ids and some other info for the views that are in this workbook.



3. Pick any of the views from the response and save it as your `{{view-id}}` variable. We're going to use that `{{view-id}}` to grab an image so for context below is a screenshot of what each view looks like.

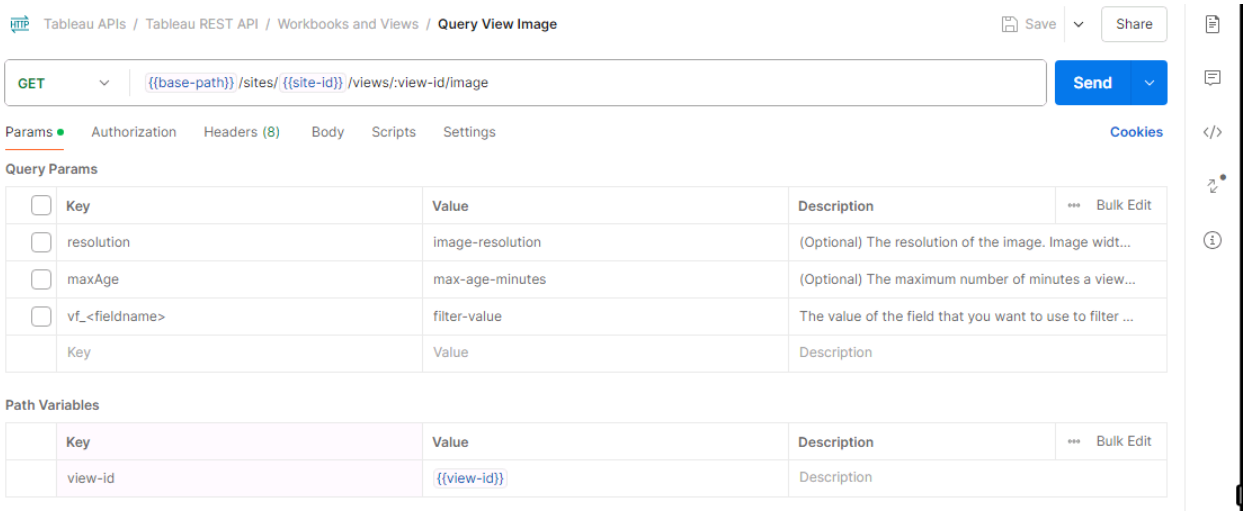


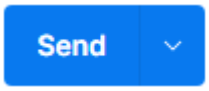
As a reminder, to save the `{{view-id}}`, select the **view id** of the view you want to query, then right click → **Set as variable** → Search for **view-id** → Click **view-id** to set your variable.



4. Under the **Workbook and Views** folder, locate the **Query View Image** API request and select it to open it up. Notice that the **Params** tab has some preset

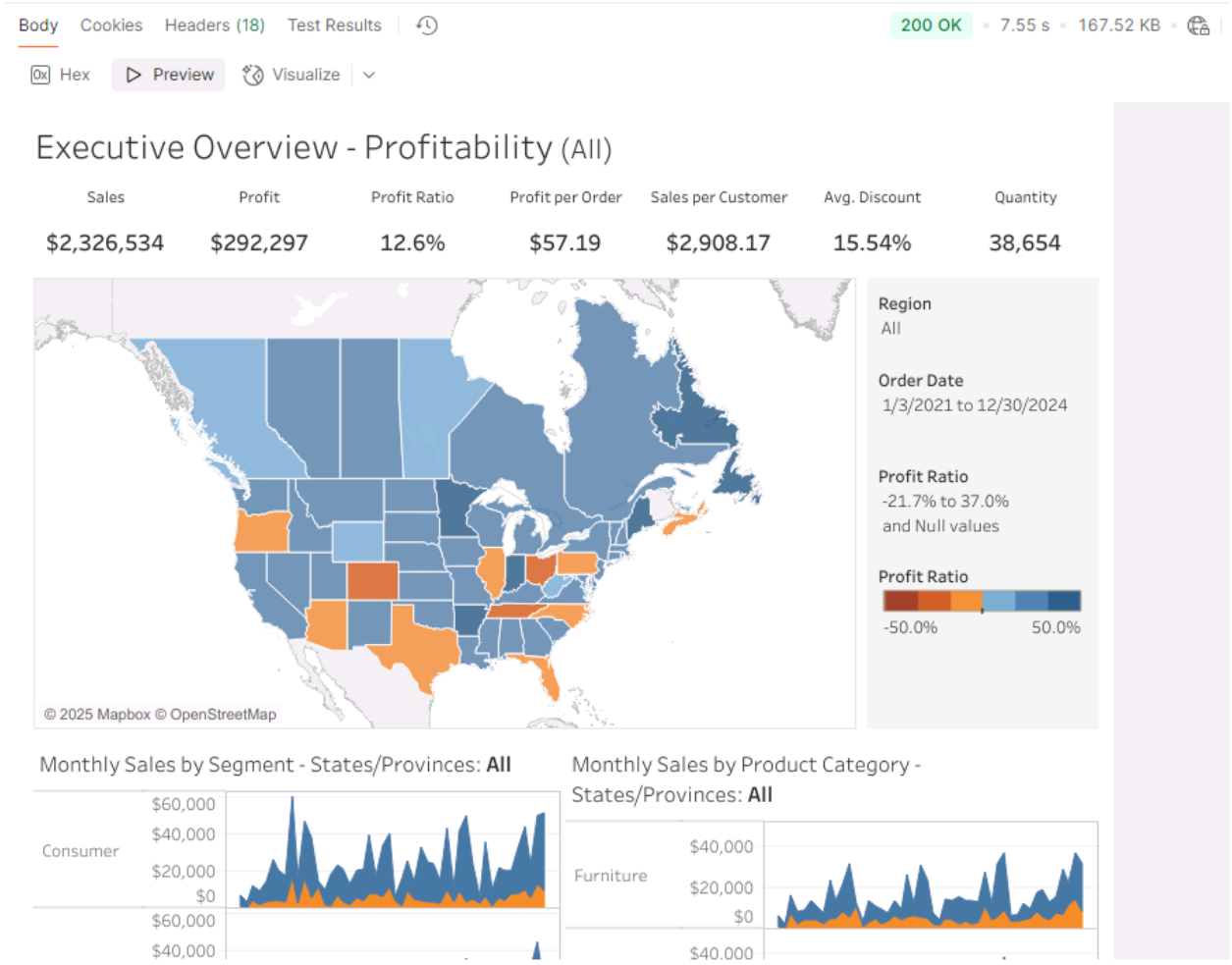
Path Variables using the `{{view-id}}` variable you just set.

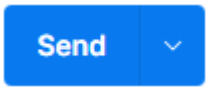


5. Let's give this one a go. Hit 

. You should get an image in the response body below, similar to the output here. Obviously, if you selected a

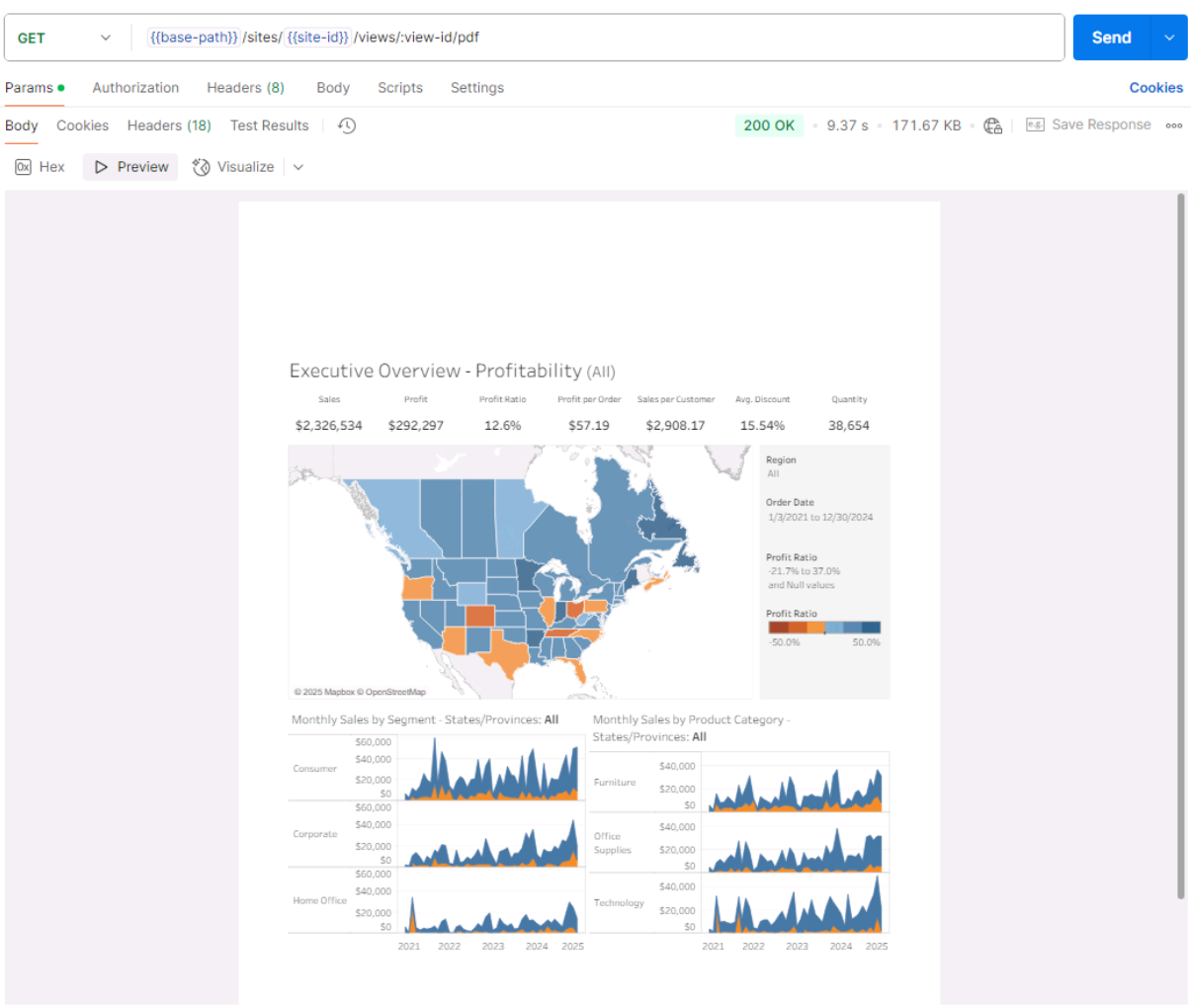
different `{{view-id}}` your image will be different.



6. Now let's do the same but get a PDF instead. Back under the **Workbook and Views** folder, locate the **Query View PDF** API request and select it to open it up. Notice that, similar to **Query View Image**, the **Params** tab has a preset **Path Variable** using the `{{view-id}}` variable. Click 

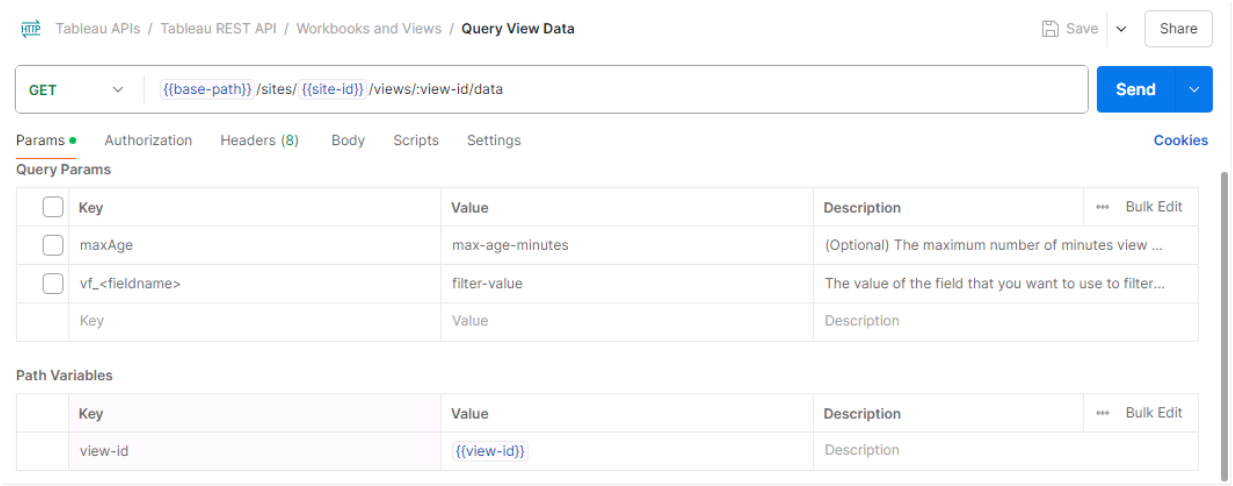
.
7. This time the output should be a formatted one page Portrait image of the same view. You can play around with some of the other **Query Params** to modify the

output.

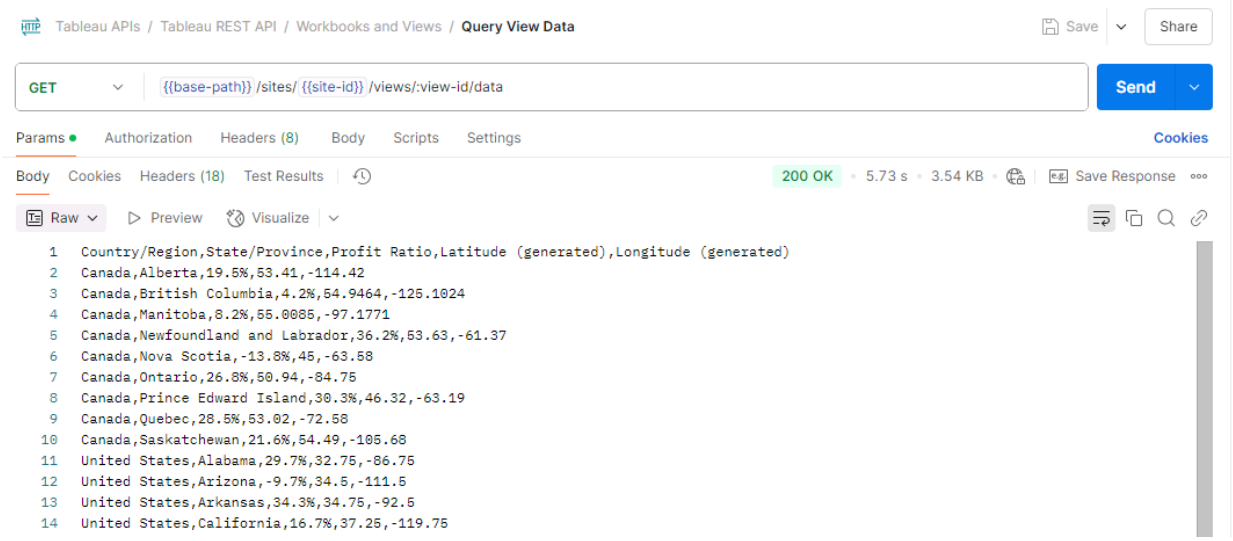


8. Finally, let’s grab data from the view. One more time under the **Workbook and Views** folder, locate the **Query View Data** API request and select it to open it up. Similar to the first 2 calls, the **Params** tab has a preset **Path Variable** using

the `{{view-id}}` variable. Hit Send.

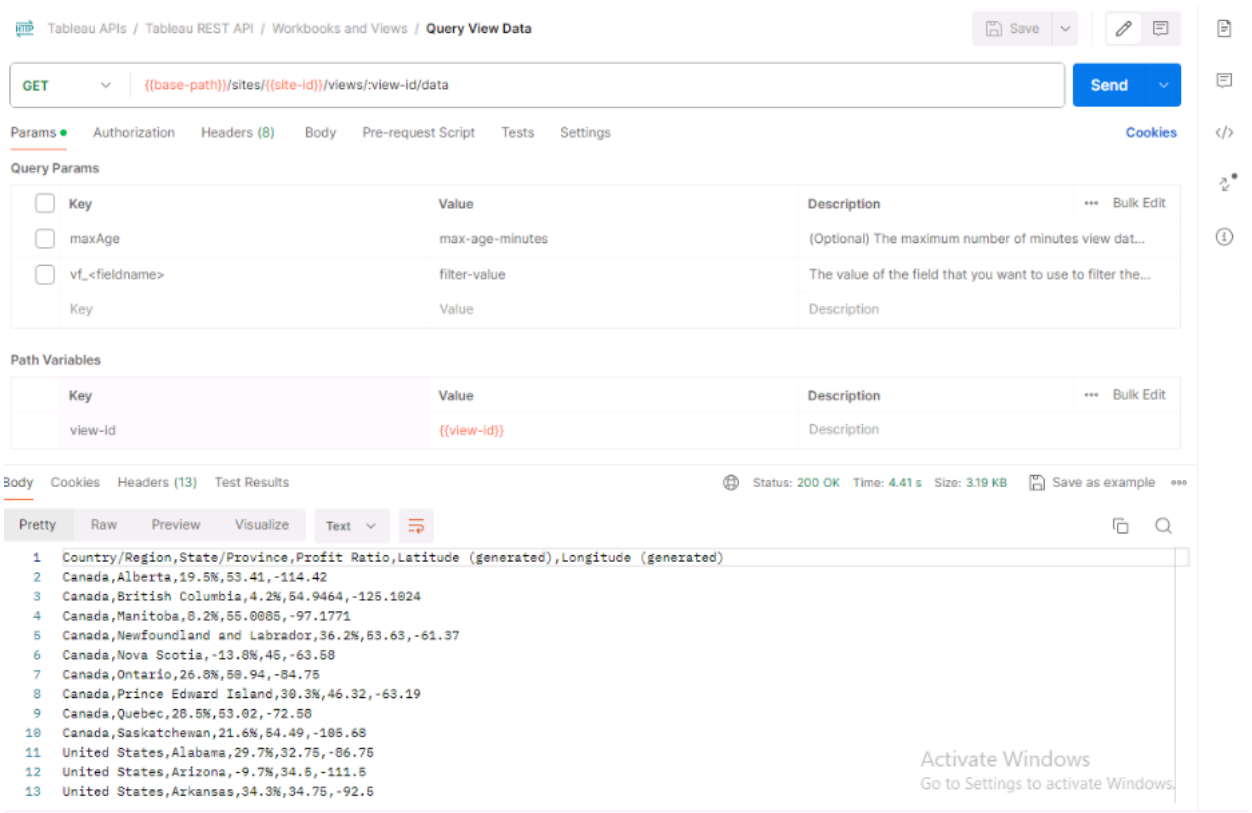


9. Depending on the view you chose, your results may vary.



Exercise #4 - Data & Filter Expressions

1. From the **Workbooks and Views** folder, open the **Query View Data** request. This requires a `{{view-id}}` variable and we still have ours from the last exercise (**Overview**). Go ahead and click Send.



3. This isn't quite what we expected. Why didn't it return the full data table for the **Overview** view? That's because when using **Query View Data** from a dashboard, only data from the first sheet 1 in the dashboard view gets returned.
4. Let's fix that. Open and run **Query Views for Workbook.** This takes a `{{workbook-id}}` as a path variable and you should already have that set from the previous exercise. From the response, scroll down until you see the view named **Order Details**. Grab the view id for **Order Details** and set this to your `{{view-id}}` variable.

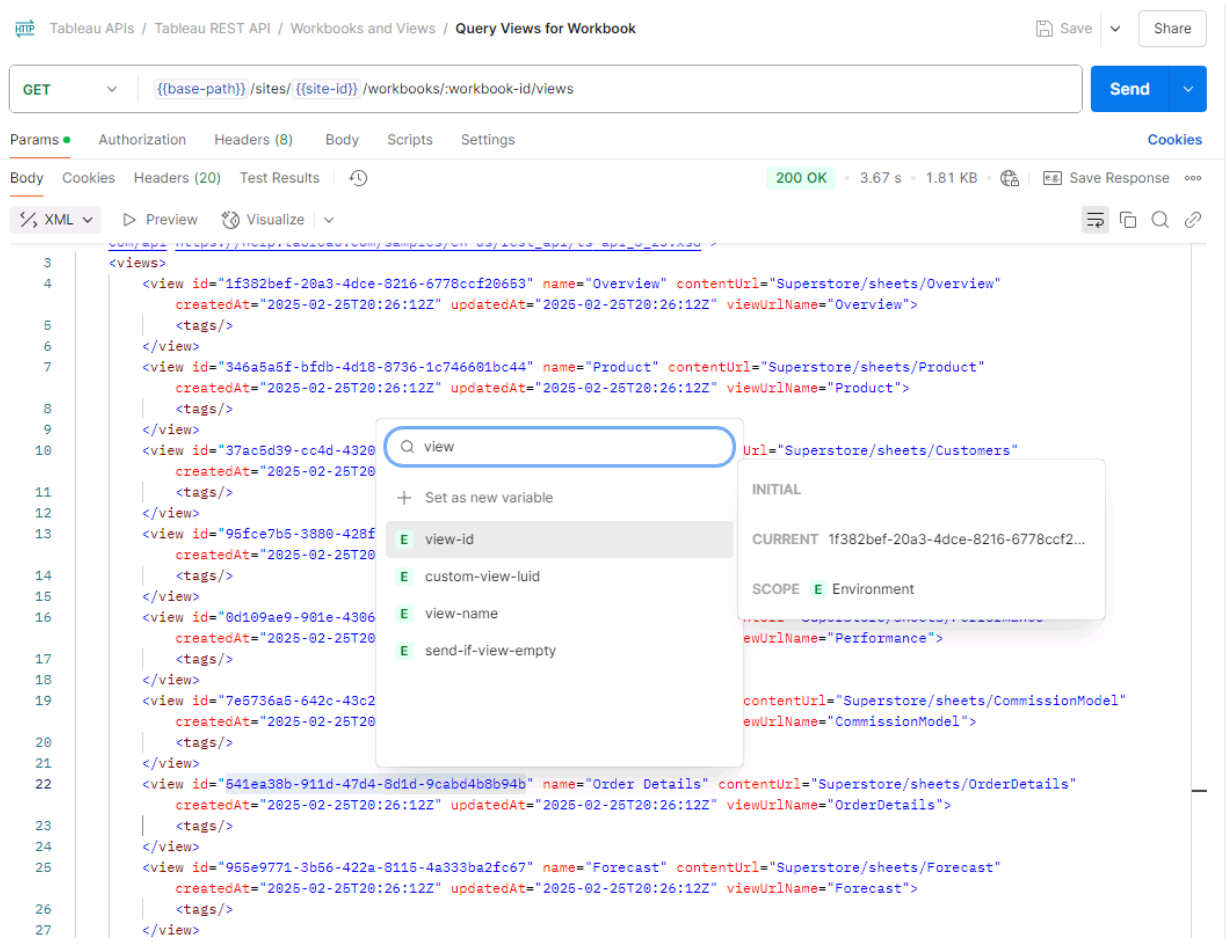
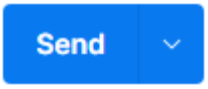
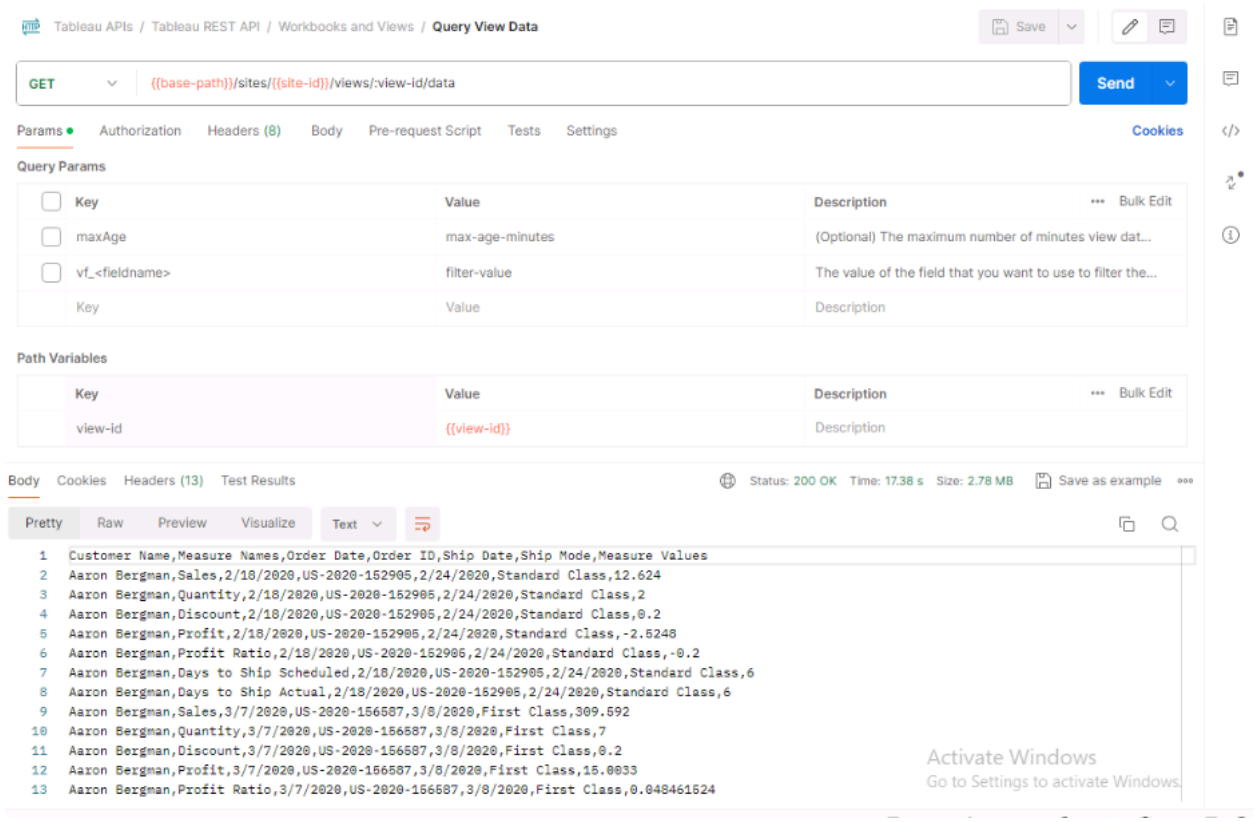


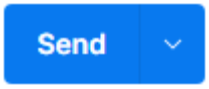
Tableau APIs / Tableau REST API / Workbooks and Views / Query Views for Workbook
Save
Share
GET
{{base-path}} /sites/ {{site-id}} /workbooks/:workbook-id/views
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
Body
Cookies
Headers (20)
Test Results
200 OK
3.67 s
1.81 KB
Save Response
XML
Preview
Visualize
<views>
<view id="1f382bef-20a3-4dce-8216-6778ccf20653" name="Overview" contentUrl="Superstore/sheets/Overview"
createdAt="2025-02-25T20:26:12Z" updatedAt="2025-02-25T20:26:12Z" viewUrlName="Overview">
<tags/>
</view>
<view id="346a5a5f-bfdb-4d18-8736-1c746601bc44" name="Product" contentUrl="Superstore/sheets/Product"
createdAt="2025-02-25T20:26:12Z" updatedAt="2025-02-25T20:26:12Z" viewUrlName="Product">
<tags/>
</view>
<view id="37ac5d39-cc4d-4320
Url="Superstore/sheets/Customers"
createdAt="2025-02-25T20
<tags/>
</view>
<view id="95fce7b5-3880-428f
createdAt="2025-02-25T20
<tags/>
</view>
<view id="0d109ae9-901e-4306
createdAt="2025-02-25T20
ewUrlName="Performance">
<tags/>
</view>
<view id="7e5736a5-642c-43c2
contentUrl="Superstore/sheets/CommissionModel"
createdAt="2025-02-25T20
ewUrlName="CommissionModel">
<tags/>
</view>
<view id="541ea38b-911d-47d4-8d1d-9cabd4b8b94b" name="Order Details" contentUrl="Superstore/sheets/OrderDetails"
createdAt="2025-02-25T20:26:12Z" updatedAt="2025-02-25T20:26:12Z" viewUrlName="OrderDetails">
<tags/>
</view>
<view id="965e9771-3b56-422a-8115-4a333ba2fc67" name="Forecast" contentUrl="Superstore/sheets/Forecast"
createdAt="2025-02-25T20:26:12Z" updatedAt="2025-02-25T20:26:12Z" viewUrlName="Forecast">
<tags/>
</view>
view
Set as new variable
E view-id
E custom-view-luid
E view-name
E send-if-view-empty
INITIAL
CURRENT 1f382bef-20a3-4dce-8216-6778ccf2...
SCOPE E Environment

5. Go back to the **Query View Data** request. Click 

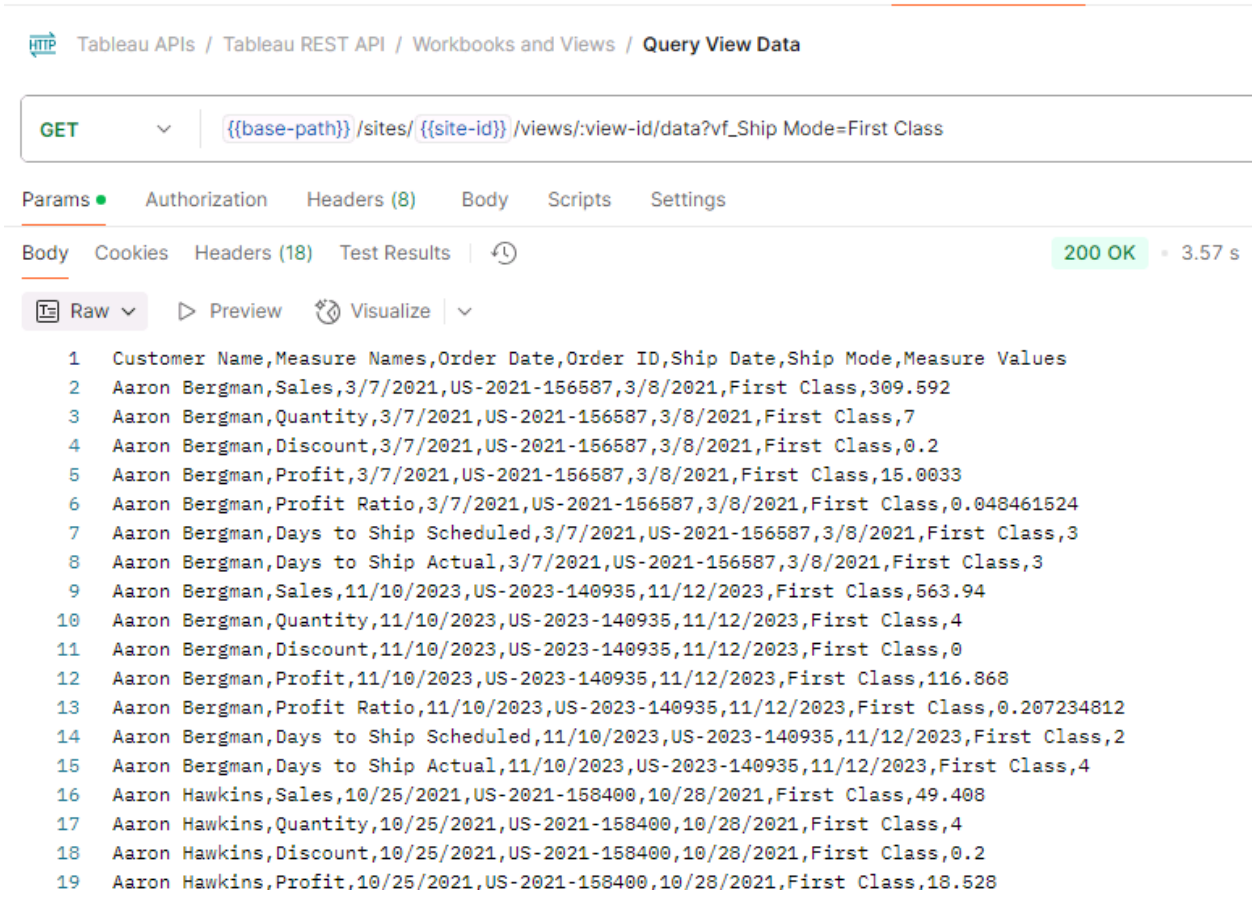
. That's more like it. This gives me more order specific detail from Superstore.



6. Let's filter this output so that we only get orders with a **Ship Mode of First Class.** Under the **Params** tab, check the box for vf_<fieldname>. Modify the vf_<fieldname> to read **vf_Ship Mode** and set the filter-value to **First Class**.

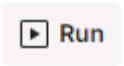
Click 

. You should now have a filtered data set result.



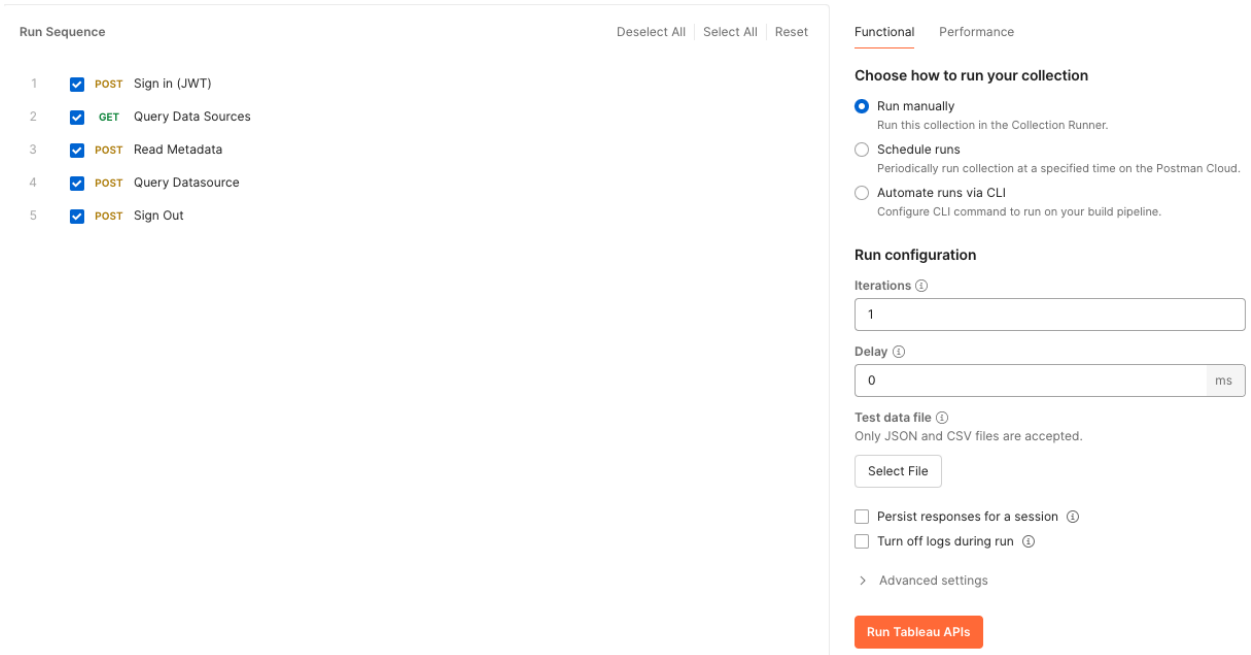
Exercise #5 - Headless BI Applications

1. Up to this point, we've focused exclusively on API calls made against workbooks or views that are published to Tableau Server or Cloud. But what if all you want is the data. Enter the VizQL Data Service API, also known as Headless BI API. Through the use of this newly introduced API, and its quick add to the Postman collection, we are able to query directly against a  Tableau Published Data Source.
2. In the interest of time, we're gonna speed up this exercise a bit. Postman has a really great feature that allows you to run a complete test of APIs in a sequence. We call them **Workflows** and have included a few samples in our **Collection**. Find the **Sample Workflows** folder at the bottom of the **Tableau APIs** Collection. Select the **VizQL Data Service Queries**, and read thru the Overview to get an idea of what is happening in this workflow.

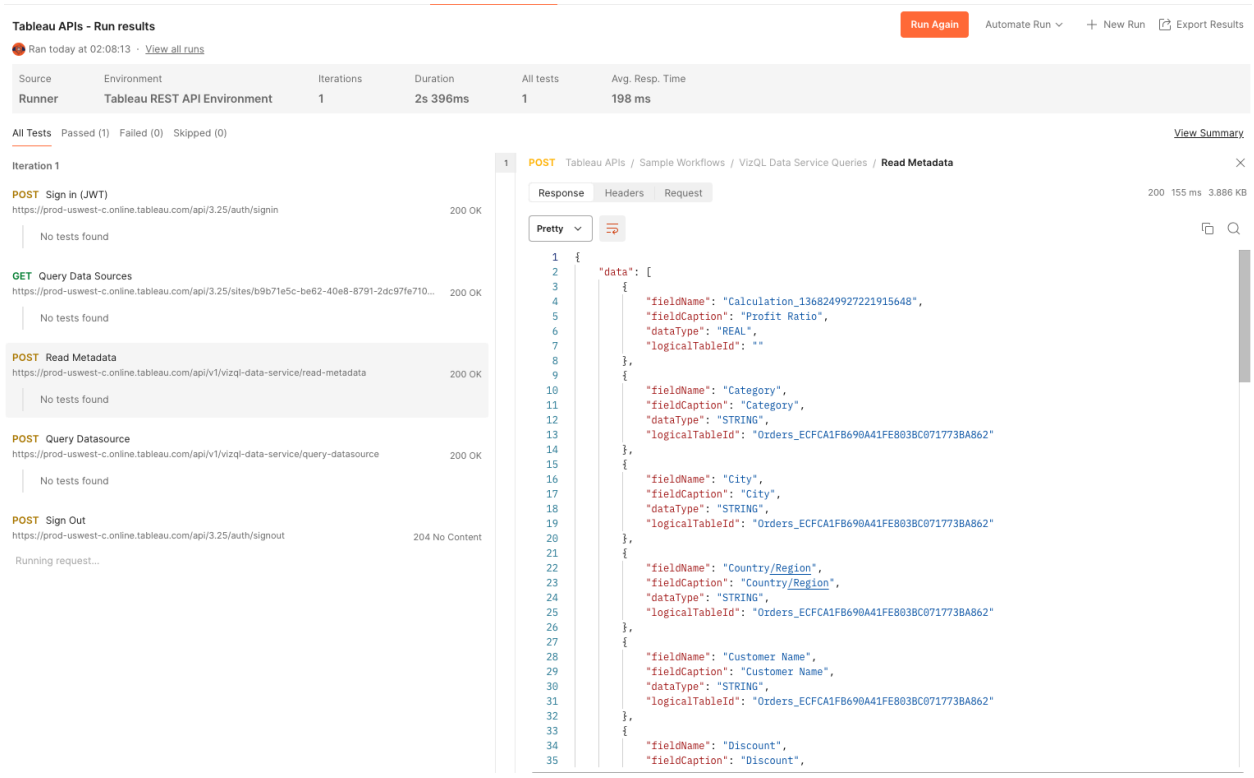
3. Click the 

button to configure the workflow. Select the checkbox ☑️ to **Persist responses for a session** and leave the rest as default. Click 

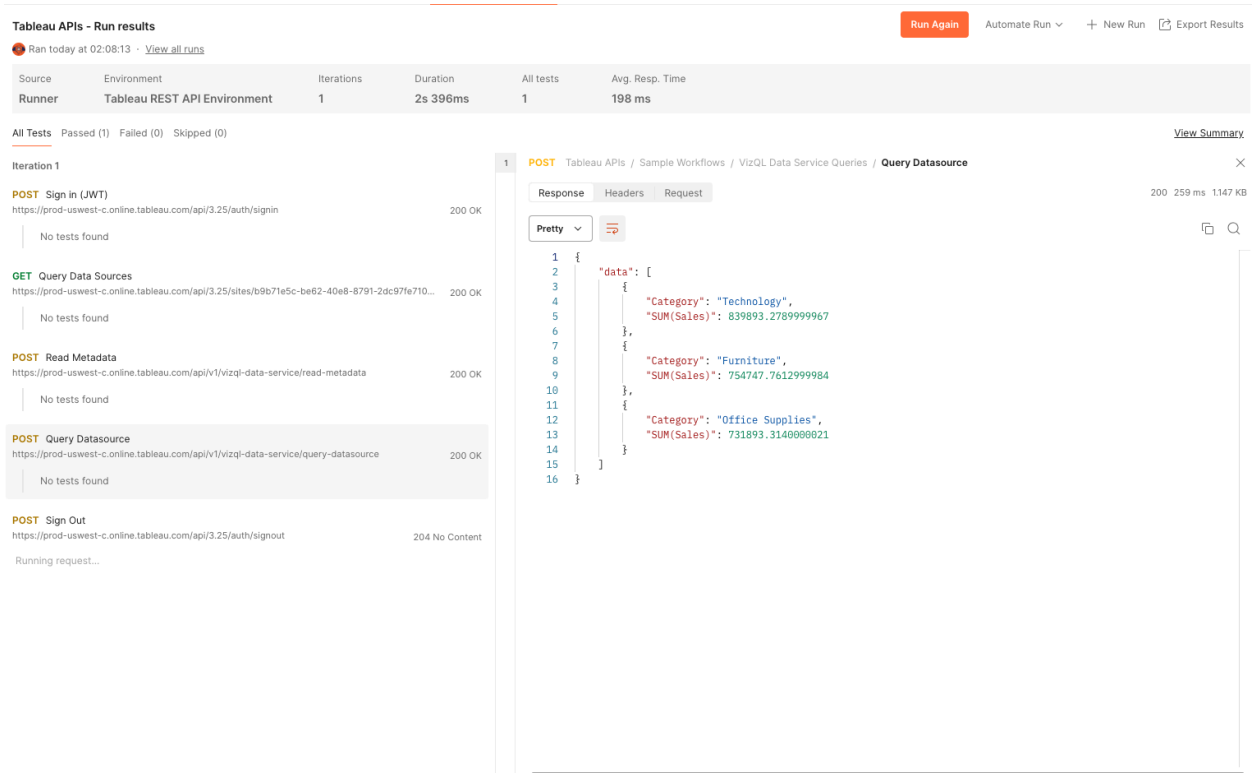
and watch the magic happen. 🪄🎩



4. Inspect the results set for the READ METADATA call



5. Inspect the results set for the Query Datasource call



Thanks so much for attending **Build, Automate, and Scale: A Tableau REST API Deep Dive Hands-on Training Session**!!

We hope this was valuable material to help you on your journey with Tableau. Please don't forget to leave feedback on the session. See ya next year!