Methane Monitoring from Space Project Team-47 Test Report

Name	Position	Email
Zachary Qi Jie Teng	Team Leader / Supervisor Liaison / Usability 2nd Lead	102416353@student.swin.edu.au
Md Radif Rafayet Chowdhury	Documentation Lead / Quality Lead / Client Liaison	103539316@student.swin.edu.au
Ho Man Lai	Documentation 2nd Lead / Usability Lead	103495104@student.swin.edu.au
Saborni Barua	Git Lead / Trello Lead / Developer 2nd Lead	103512168@student.swin.edu.au
Uddhav Grover	Research Lead / Planning Lead	103513802@student.swin.edu.au
Ang Fu	Developer Lead / Research 2nd Lead	103001255@student.swin.edu.au

Document Sign-Off:

Name	Position	Signature	Date
Zachary Qi Jie Teng	Team Leader / Supervisor Liaison / Usability 2nd Lead	Z.T.	18/10/2023
Md Radif Rafayet Chowdhury	Documentation Lead / Quality Lead / Client Liaison	Radif	18/10/2023
Ho Man Lai	Documentation 2nd Lead / Usability Lead	Ho Man Lai	18/10/2023
Saborni Barua	Git Lead / Trello Lead / Developer 2nd Lead	Saborni Barua	18/10/2023
Uddhav Grover	Research Lead / Planning Lead		
Ang Fu	Developer Lead / Research 2nd Lead		

Overview:

The purpose of this test report is to document the functional testing activities carried out for the Methane Monitoring from Space project. The report provides a comprehensive analysis of the system's capabilities, focusing on two key Python functions: **search_catalog** and **convert_format_date**. These functions are integral to the system's ability to query satellite data based on geographical and temporal parameters and to handle date conversions, respectively. The testing is sectionalized into unit testing and integration testing.

Testing Scope:

In Scope:

Functional Testing:

- Unit testing for the **search_catalog** function to validate bounding box and date period parameters.
- Unit testing for the **convert_format_date** function to validate various date formats.

Out of Scope:

End-to-End Testing: Comprehensive testing that evaluates the system's overall behavior from start to finish is not covered in this report.

Regression Testing: Ensuring that new changes have not adversely affected the existing functionalities of the system is not within the scope of this report.

Test Environment & Tools:

Device	Hardware Configuration	Software Configuration & Tools
Unit Testing		
PC 1	• X64 64-bit CPU • AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz • 32 GB RAM	• IDE: Visual Studio Code • Python ver: 3.14 • Unittest library
PC 2	• 11th Gen Intel(R) Core i7-1165G7 2.80 GHz 1.69 GHz • X64 64-bit CPU • 8 GB RAM	• IDE: Visual Studio Code • Python ver: 3.10.9 • Unittest library
Integration Testing		
PC 3	• X64 64-bit CPU • Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz • 8.00 GB (7.75 GB usable)	• Microsoft Planetary Hub
PC 4		

Functional testing:

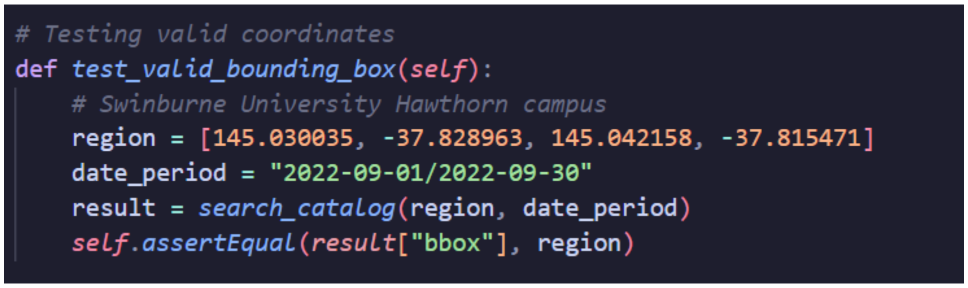
1. Unit Testing

1.1. Individual Components

Test Component(s)	Description	Expected Results	Actual Results	Passed/Failed
search_catalog Function	Validate bounding box and date period	Correct search parameters for valid inputs	Same as expected result	Passed
convert_format_date Function	Validate date formats	Date in "YYYY-MM-DD" format for valid inputs	Same as expected result	Passed

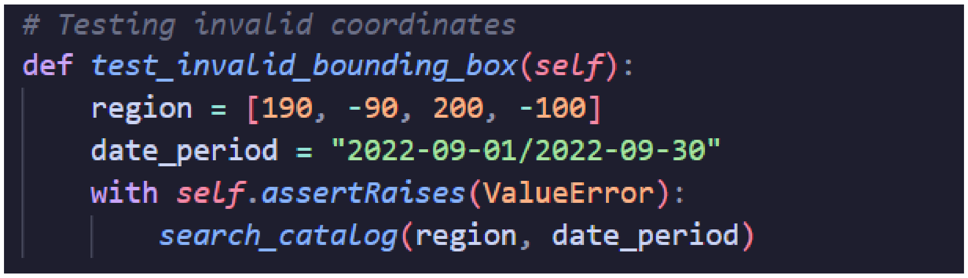
a. Unit Testing for ‘search_catalog’

Test Case 1: Valid Bounding Box (search_catalog)

Input	Output	Was it Expected?
‘region = [145.030035, -37.828963, 145.042158, -37.815471]’ **‘date_period = "2022-09-01/2022-09-30"’**	‘{"bbox": [145.030035, -37.828963, 145.042158, -37.815471]}’	**yes**
		

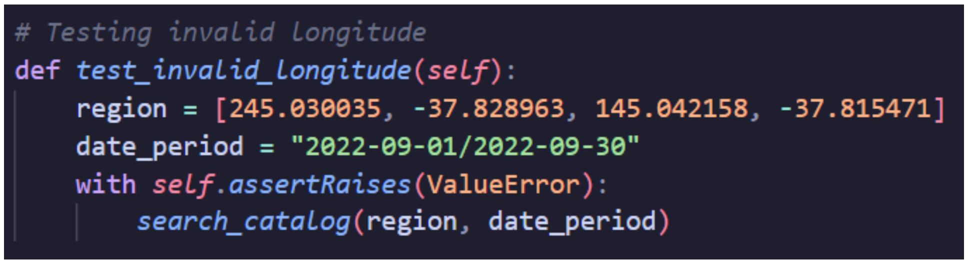
```python
# Testing valid coordinates
def test_valid_bounding_box(self):
    # Swinburne University Hawthorn campus
    region = [145.030035, -37.828963, 145.042158, -37.815471]
    date_period = "2022-09-01/2022-09-30"
    result = search_catalog(region, date_period)
    self.assertEqual(result["bbox"], region)
```

Test Case 2: Invalid Bounding Box (search_catalog)

Input	Output	Was it Expected?
‘region = [190, -90, 200, -100]’ **‘date_period = "2022-09-01/2022-09-30"’**	Raises ‘ ValueError: "Invalid coordinate in bbox" ‘	**Yes**
		

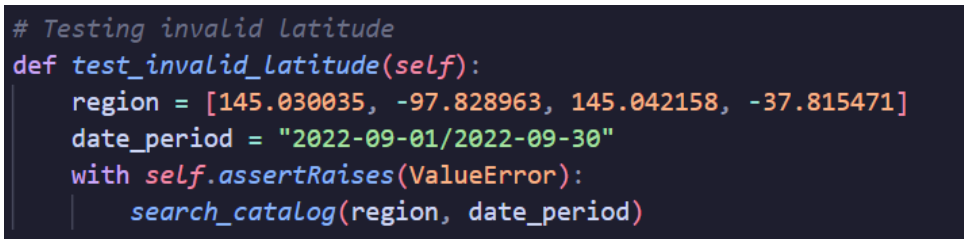
```python
# Testing invalid coordinates
def test_invalid_bounding_box(self):
    region = [190, -90, 200, -100]
    date_period = "2022-09-01/2022-09-30"
    with self.assertRaises(ValueError):
        search_catalog(region, date_period)
```

Test Case 3: Invalid longitude (search_catalog)

Input	**Output**	**Was it Expected?**
'region = [245.030035, -37.828963, 145.042158, -37.815471]' 'date_period = "2022-09-01/2022-09-30"'	Raises ' ValueError: "Invalid longitudes in bbox" '	**Yes**
		

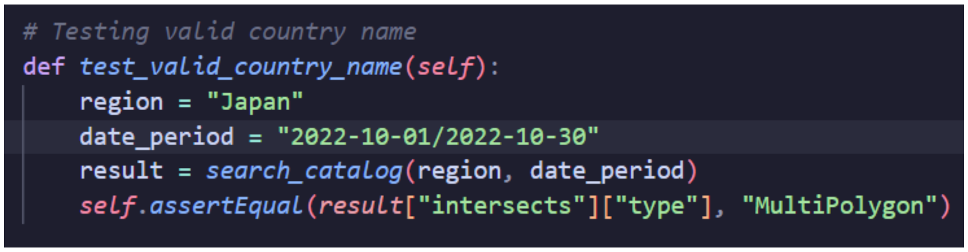
```python
# Testing invalid longitude
def test_invalid_longitude(self):
    region = [245.030035, -37.828963, 145.042158, -37.815471]
    date_period = "2022-09-01/2022-09-30"
    with self.assertRaises(ValueError):
        search_catalog(region, date_period)
```

Test Case 4: Invalid latitude (search_catalog)

Input	**Output**	**Was it Expected?**
'region = [145.030035, -97.828963, 145.042158, -37.815471]' 'date_period = "2022-09-01/2022-09-30"'	Raises ' ValueError: "Invalid latitude in bbox" '	**Yes**
		

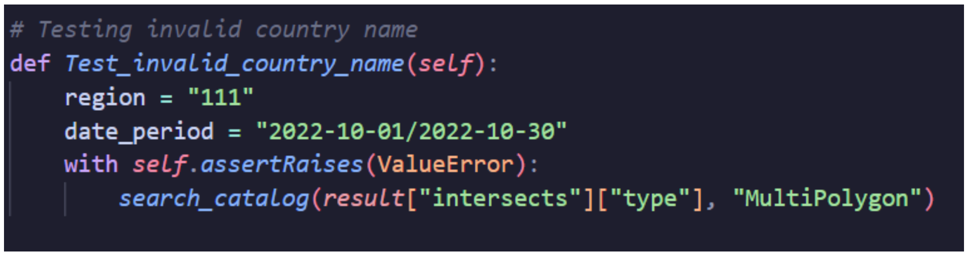
```python
# Testing invalid latitude
def test_invalid_latitude(self):
    region = [145.030035, -97.828963, 145.042158, -37.815471]
    date_period = "2022-09-01/2022-09-30"
    with self.assertRaises(ValueError):
        search_catalog(region, date_period)
```

Test Case 5: Valid country name (search_catalog)

Input	Output	Was it Expected?
‘region = “Japan”’ ‘date_period = "2022-10-01/2022-10-30"’	‘Number of items for input: 62’	**Yes**
		

```
# Testing valid country name
def test_valid_country_name(self):
    region = "Japan"
    date_period = "2022-10-01/2022-10-30"
    result = search_catalog(region, date_period)
    self.assertEqual(result["intersects"]["type"], "MultiPolygon")
```

Test Case 6: Invalid country name (search_catalog)

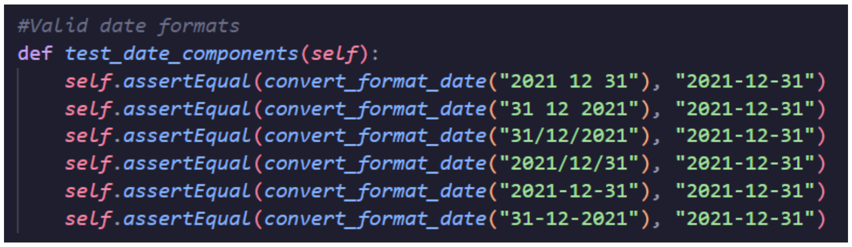
Input	Output	Was it Expected?
‘region = “111”’ ‘date_period = "2022-10-01/2022-10-30"’	Raises ‘IndexError: “list index out of range”’	**Yes**
		

```
# Testing invalid country name
def Test_invalid_country_name(self):
    region = "111"
    date_period = "2022-10-01/2022-10-30"
    with self.assertRaises(ValueError):
        search_catalog(result["intersects"]["type"], "MultiPolygon")
```

b. Unit Testing for ‘convert_format_date’

Test Case 1: Valid Date Format (convert_format_date)

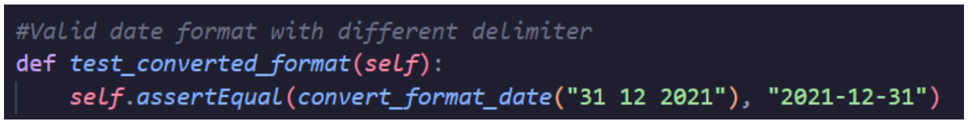
Input	Output	Was it Expected?
‘input_date = "2021 12 31"’	‘"2021-12-31"’	**Yes**



```
#Valid date formats
def test_date_components(self):
    self.assertEqual(convert_format_date("2021 12 31"), "2021-12-31")
    self.assertEqual(convert_format_date("31 12 2021"), "2021-12-31")
    self.assertEqual(convert_format_date("31/12/2021"), "2021-12-31")
    self.assertEqual(convert_format_date("2021/12/31"), "2021-12-31")
    self.assertEqual(convert_format_date("2021-12-31"), "2021-12-31")
    self.assertEqual(convert_format_date("31-12-2021"), "2021-12-31")
```

Test Case 2: Valid Date Format with Different Delimiter (convert_format_date)

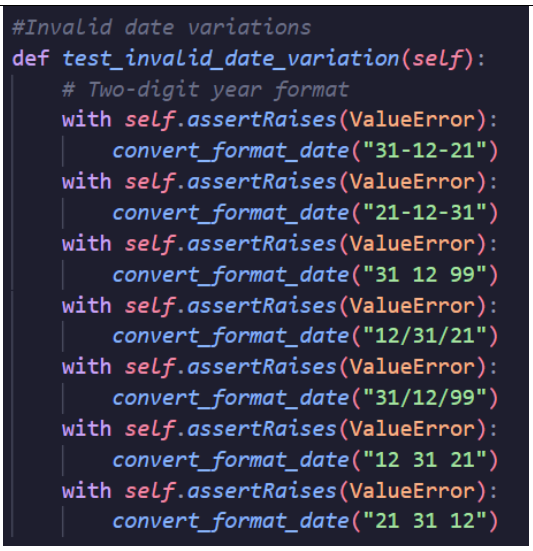
Input	Output	Was it Expected?
‘input_date = "31 12 2021"’	Raises ‘"2021-12-31"’	**Yes**



```
#Valid date format with different delimiter
def test_converted_format(self):
    self.assertEqual(convert_format_date("31 12 2021"), "2021-12-31")
```

Test Case 3: Invalid Date Variation or Format (convert_format_date)

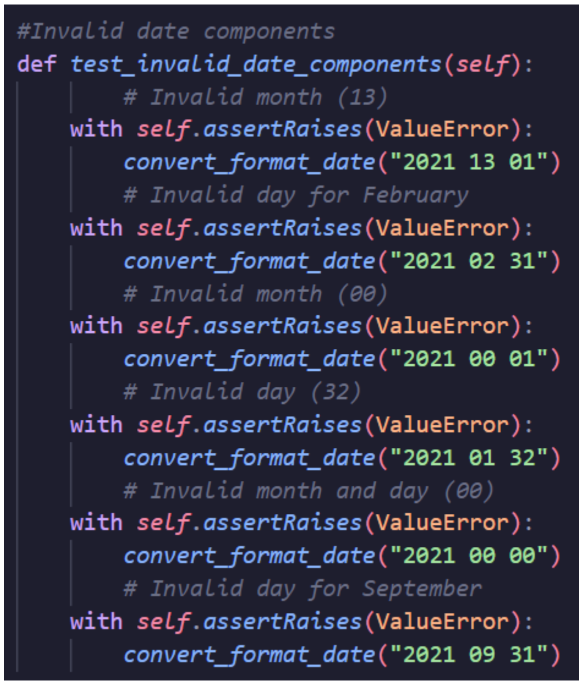
Input	Output	Was it Expected?
‘input_date = "31 12 21"’	Raises ‘ValueError: "Invalid data format" ‘	**yes**



```python
#Invalid date variations
def test_invalid_date_variation(self):
    # Two-digit year format
    with self.assertRaises(ValueError):
        convert_format_date("31-12-21")
    with self.assertRaises(ValueError):
        convert_format_date("21-12-31")
    with self.assertRaises(ValueError):
        convert_format_date("31 12 99")
    with self.assertRaises(ValueError):
        convert_format_date("12/31/21")
    with self.assertRaises(ValueError):
        convert_format_date("31/12/99")
    with self.assertRaises(ValueError):
        convert_format_date("12 31 21")
    with self.assertRaises(ValueError):
        convert_format_date("21 31 12")
```

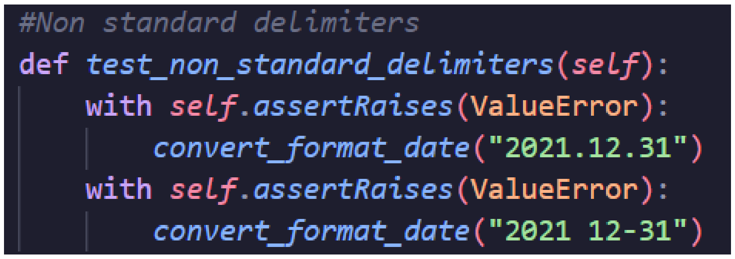
Test Case 4: Invalid Date Components (convert_format_date)

Input	Output	Was it Expected?
‘**input_date = "2021 13 01"**’	Raises ‘ValueError: "Invalid data format"’	**Yes**



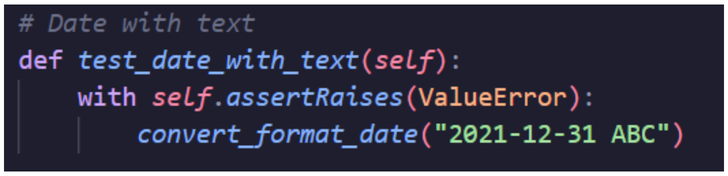
```python
#Invalid date components
def test_invalid_date_components(self):
        # Invalid month (13)
    with self.assertRaises(ValueError):
        convert_format_date("2021 13 01")
        # Invalid day for February
    with self.assertRaises(ValueError):
        convert_format_date("2021 02 31")
        # Invalid month (00)
    with self.assertRaises(ValueError):
        convert_format_date("2021 00 01")
        # Invalid day (32)
    with self.assertRaises(ValueError):
        convert_format_date("2021 01 32")
        # Invalid month and day (00)
    with self.assertRaises(ValueError):
        convert_format_date("2021 00 00")
        # Invalid day for September
    with self.assertRaises(ValueError):
        convert_format_date("2021 09 31")
```

Test Case 5: Non-Standard Delimiters (convert_format_date)

Input	**Output**	**Was it Expected?**
‘input_date = "2021.12.31"’	Raises ‘ValueError: "Invalid data format"’	**Yes**
		

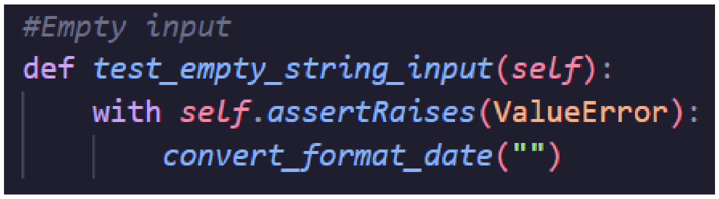
```python
#Non standard delimiters
def test_non_standard_delimiters(self):
    with self.assertRaises(ValueError):
        convert_format_date("2021.12.31")
    with self.assertRaises(ValueError):
        convert_format_date("2021 12-31")
```

Test Case 6: Date with Text (convert_format_date)

Input	**Output**	**Was it Expected?**
‘input_date = "2021-12-31 ABC"’	Raises ‘ValueError: "Invalid data format" ‘	**Yes**
		

```python
# Date with text
def test_date_with_text(self):
    with self.assertRaises(ValueError):
        convert_format_date("2021-12-31 ABC")
```

Test Case 7: Empty Input (convert_format_date)

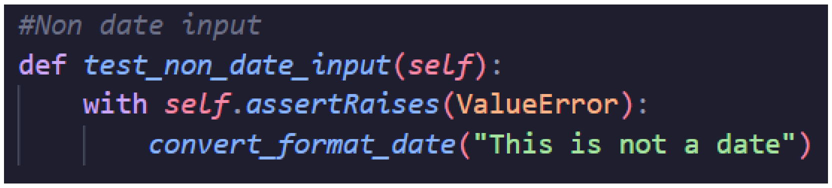
Input	**Output**	**Was it Expected?**
‘input_date = ""’	Raises ‘ValueError: "Invalid start/end date format. Please check the acceptable formats" ‘	**Yes**
		

```python
#Empty input
def test_empty_string_input(self):
    with self.assertRaises(ValueError):
        convert_format_date("")
```

Test Case 8: Non Date Input (convert_format_date)

Input	**Output**	**Was it Expected?**

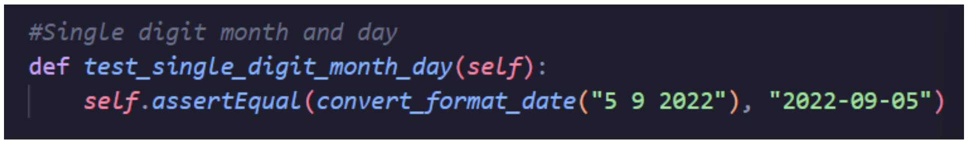
'input_date = "This is not a date"'	Raises 'ValueError: "Invalid start/end date format. Please check the acceptable formats" '	**Yes**



```
#Non date input
def test_non_date_input(self):
    with self.assertRaises(ValueError):
        convert_format_date("This is not a date")
```

Test Case 9: Single Digit Month and Day (convert_format_date)

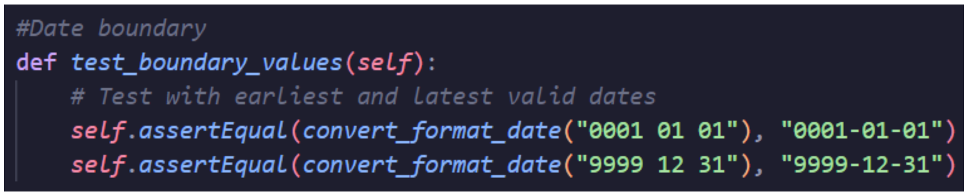
Input	**Output**	**Was it Expected?**
'input_date = "5 9 2022"'	Raises '"05-09-2022"'	**Yes**



```
#Single digit month and day
def test_single_digit_month_day(self):
    self.assertEqual(convert_format_date("5 9 2022"), "2022-09-05")
```

Test Case 10: Date Boundary (convert_format_date)

Input	**Output**	**Was it Expected?**
'input_date = "0001 01 01"'	Raises '"0001-01-01"'	**Yes**



```
#Date boundary
def test_boundary_values(self):
    # Test with earliest and latest valid dates
    self.assertEqual(convert_format_date("0001 01 01"), "0001-01-01")
    self.assertEqual(convert_format_date("9999 12 31"), "9999-12-31")
```

1.2. Inputs for Data Validation

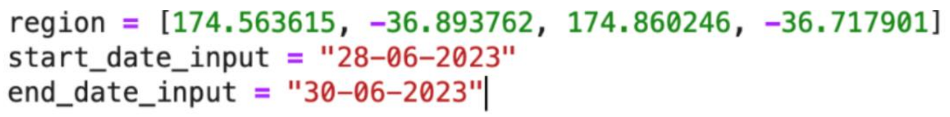
Test Case(s)	**Description**	**Expected results**	**Actual Results**	**Passed/Failed**
Valid Region and Date	The system should accept valid region and date period	The system should return correct search parameters	Same as expected result	Passed
Invalid Date Format	The system should not accept invalid date formats	The system should raise a **ValueError**	Same as expected result	Passed

Integration Testing:

We have conducted integration testing for both user input processing (date format) and data search query functionalities. These two aspects are interdependent, meaning that if there is a failure in the user input processing, it will subsequently result in a failure when searching for data. The user input processing module is subjected to multiple test cases as described below. In cases where valid date formats are provided, the date format would be changed to the default format (YYYY-MM-DD) and the data search query component will successfully retrieve and print items corresponding to the valid dates. Conversely, if the date format is invalid, the search query will generate an error instead of returning data.

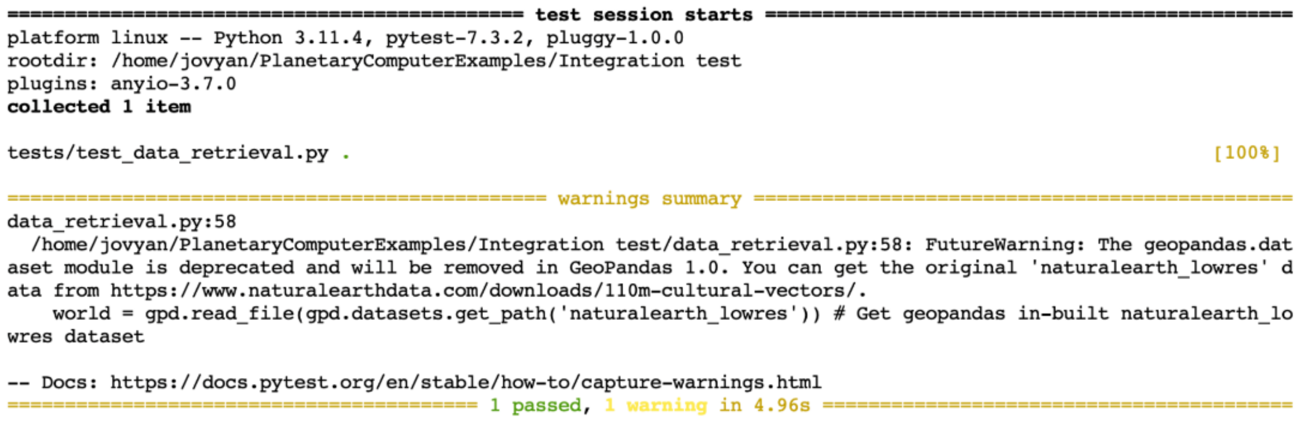
TestCase1: Valid Input Format - Dashes (DD-MM-YYYY):

Input:



```
region = [174.563615, -36.893762, 174.860246, -36.717901]
start_date_input = "28-06-2023"
end_date_input = "30-06-2023"
```

Output:



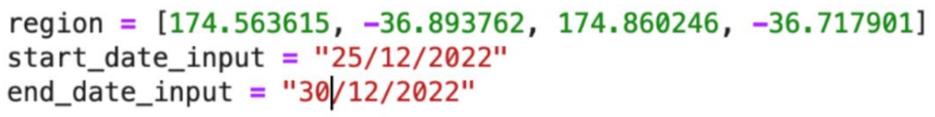
```
============================================= test session starts ==============================================
platform linux -- Python 3.11.4, pytest-7.3.2, pluggy-1.0.0
rootdir: /home/jovyan/PlanetaryComputerExamples/Integration test
plugins: anyio-3.7.0
collected 1 item

tests/test_data_retrieval.py .                                                                           [100%]

=============================================== warnings summary ===============================================
data_retrieval.py:58
  /home/jovyan/PlanetaryComputerExamples/Integration test/data_retrieval.py:58: FutureWarning: The geopandas.dat
aset module is deprecated and will be removed in GeoPandas 1.0. You can get the original 'naturalearth_lowres' d
ata from https://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
    world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # Get geopandas in-built naturalearth_lo
wres dataset

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
========================================= 1 passed, 1 warning in 4.96s =========================================
```

Expected Output: Pass

Real Output: Pass

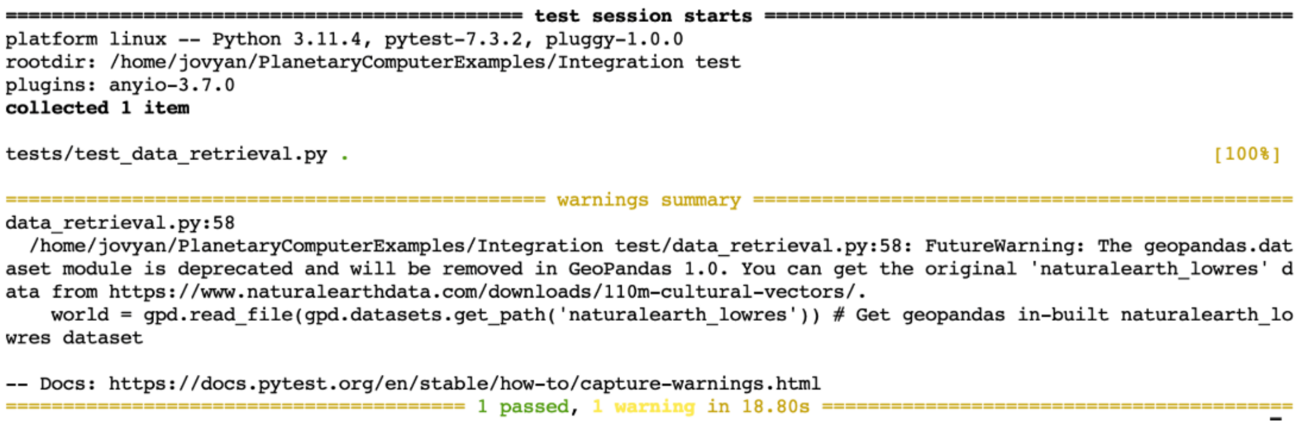
TestCase2: Valid Input Format - Slashes (DD/MM/YYYY):

Input:



```
region = [174.563615, -36.893762, 174.860246, -36.717901]
start_date_input = "25/12/2022"
end_date_input = "30/12/2022"
```

Output:



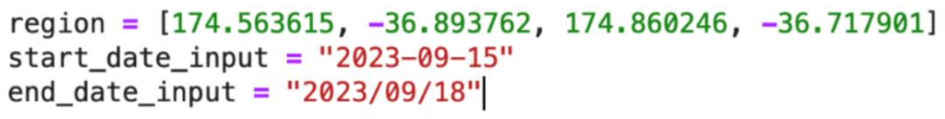
```
============================================== test session starts ==============================================
platform linux -- Python 3.11.4, pytest-7.3.2, pluggy-1.0.0
rootdir: /home/jovyan/PlanetaryComputerExamples/Integration test
plugins: anyio-3.7.0
collected 1 item

tests/test_data_retrieval.py .                                                                          [100%]

=============================================== warnings summary ================================================
data_retrieval.py:58
  /home/jovyan/PlanetaryComputerExamples/Integration test/data_retrieval.py:58: FutureWarning: The geopandas.dat
aset module is deprecated and will be removed in GeoPandas 1.0. You can get the original 'naturalearth_lowres' d
ata from https://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
    world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # Get geopandas in-built naturalearth_lo
wres dataset

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
========================================= 1 passed, 1 warning in 18.80s ==========================================
```

Expected Output: Pass

Real Output: Pass

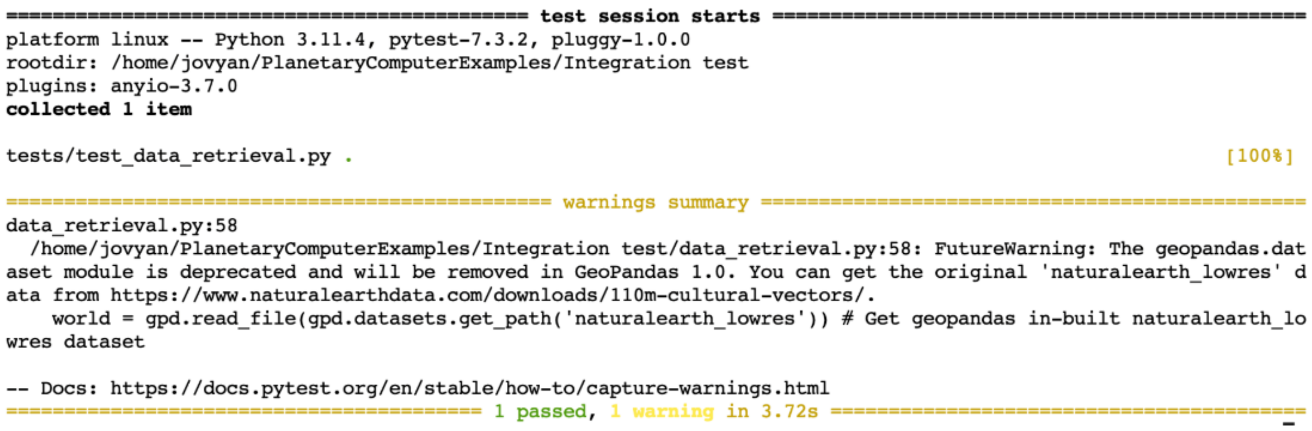
TestCase3: Valid Input Format – accepting different formats (DD-MM-YYYY/YYYY-MM-DD):

Input:



```
region = [174.563615, -36.893762, 174.860246, -36.717901]
start_date_input = "2023-09-15"
end_date_input = "2023/09/18"
```

Output:



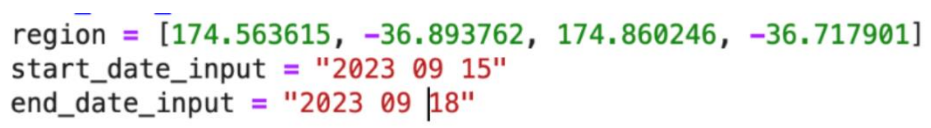
```
============================================== test session starts ==============================================
platform linux -- Python 3.11.4, pytest-7.3.2, pluggy-1.0.0
rootdir: /home/jovyan/PlanetaryComputerExamples/Integration test
plugins: anyio-3.7.0
collected 1 item

tests/test_data_retrieval.py .                                                                          [100%]

=============================================== warnings summary ================================================
data_retrieval.py:58
  /home/jovyan/PlanetaryComputerExamples/Integration test/data_retrieval.py:58: FutureWarning: The geopandas.dat
aset module is deprecated and will be removed in GeoPandas 1.0. You can get the original 'naturalearth_lowres' d
ata from https://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
    world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # Get geopandas in-built naturalearth_lo
wres dataset

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
========================================== 1 passed, 1 warning in 3.72s ==========================================
```

Expected Output: Pass

Real Output: Pass

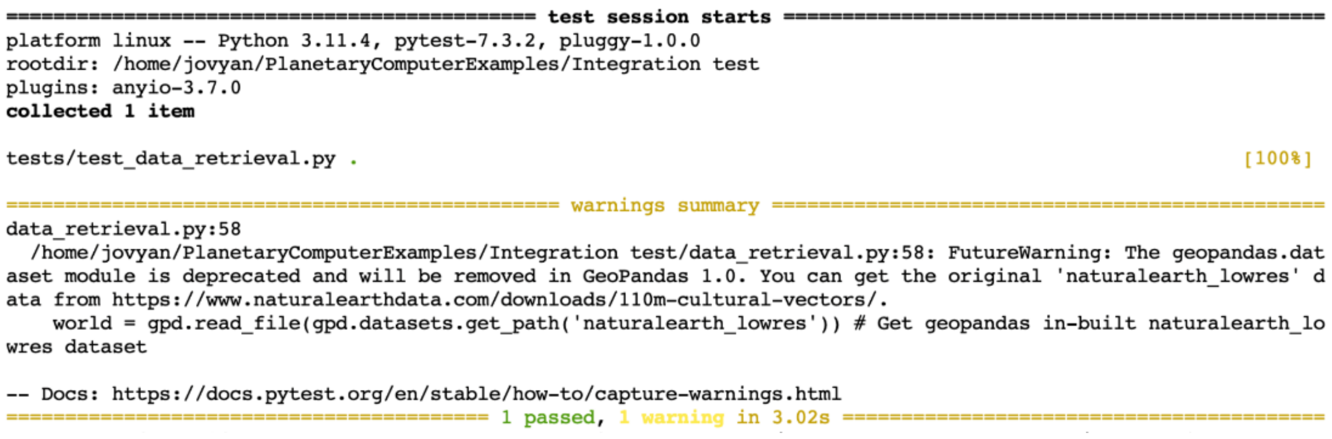
TestCase4: Valid Input Format - Spaces (DD MM YYYY):

Input:



```
region = [174.563615, -36.893762, 174.860246, -36.717901]
start_date_input = "2023 09 15"
end_date_input = "2023 09 18"
```

Output:



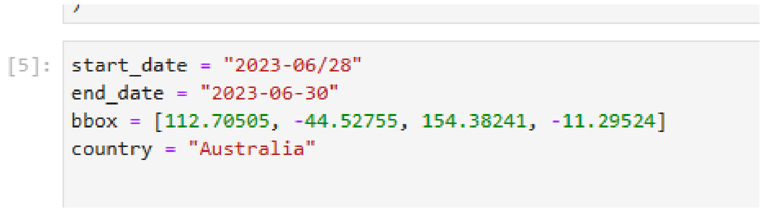
```
============================================= test session starts ==============================================
platform linux -- Python 3.11.4, pytest-7.3.2, pluggy-1.0.0
rootdir: /home/jovyan/PlanetaryComputerExamples/Integration test
plugins: anyio-3.7.0
collected 1 item

tests/test_data_retrieval.py .                                                                            [100%]

=============================================== warnings summary ===============================================
data_retrieval.py:58
  /home/jovyan/PlanetaryComputerExamples/Integration test/data_retrieval.py:58: FutureWarning: The geopandas.dat
aset module is deprecated and will be removed in GeoPandas 1.0. You can get the original 'naturalearth_lowres' d
ata from https://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
    world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # Get geopandas in-built naturalearth_lo
wres dataset

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
========================================= 1 passed, 1 warning in 3.02s =========================================
```

Expected Output: Pass

Real Output: Pass

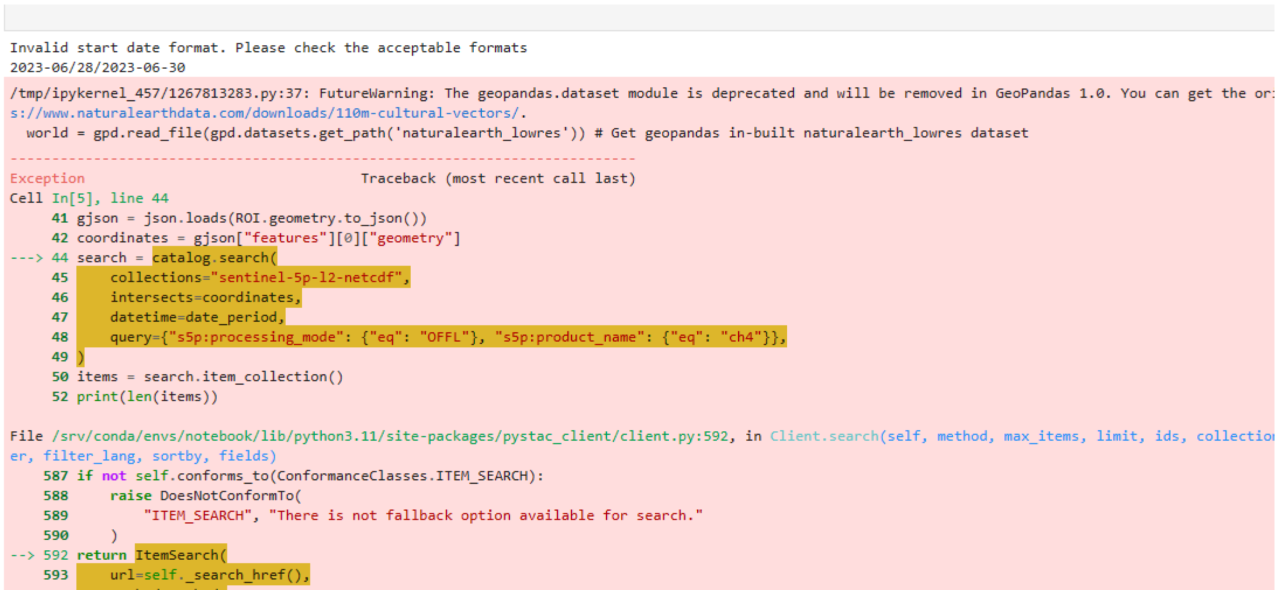
TestCase5: Invalid Input Format (Invalid format with both slashes and dashes)

Input:



```
[5]: start_date = "2023-06/28"
     end_date = "2023-06-30"
     bbox = [112.70505, -44.52755, 154.38241, -11.29524]
     country = "Australia"
```

Output:



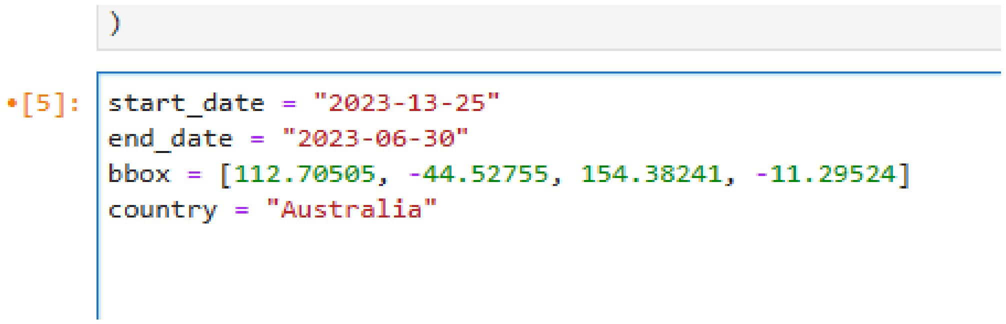
```
Invalid start date format. Please check the acceptable formats
2023-06/28/2023-06-30
/tmp/ipykernel_457/1267813283.py:37: FutureWarning: The geopandas.dataset module is deprecated and will be removed in GeoPandas 1.0. You can get the or
s://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
  world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # Get geopandas in-built naturalearth_lowres dataset
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Cell In[5], line 44
     41 gjson = json.loads(ROI.geometry.to_json())
     42 coordinates = gjson["features"][0]["geometry"]
---> 44 search = catalog.search(
     45     collections="sentinel-5p-l2-netcdf",
     46     intersects=coordinates,
     47     datetime=date_period,
     48     query={"s5p:processing_mode": {"eq": "OFFL"}, "s5p:product_name": {"eq": "ch4"}},
     49 )
     50 items = search.item_collection()
     52 print(len(items))

File /srv/conda/envs/notebook/lib/python3.11/site-packages/pystac_client/client.py:592, in Client.search(self, method, max_items, limit, ids, collectio
er, filter_lang, sortby, fields)
    587 if not self.conforms_to(ConformanceClasses.ITEM_SEARCH):
    588     raise DoesNotConformTo(
    589         "ITEM_SEARCH", "There is not fallback option available for search."
    590     )
--> 592 return ItemSearch(
    593     url=self._search_href(),
```

Expected Output: Fail

Real Output: Fail

TestCase6: Invalid Input Format (Month 13 is out of range)

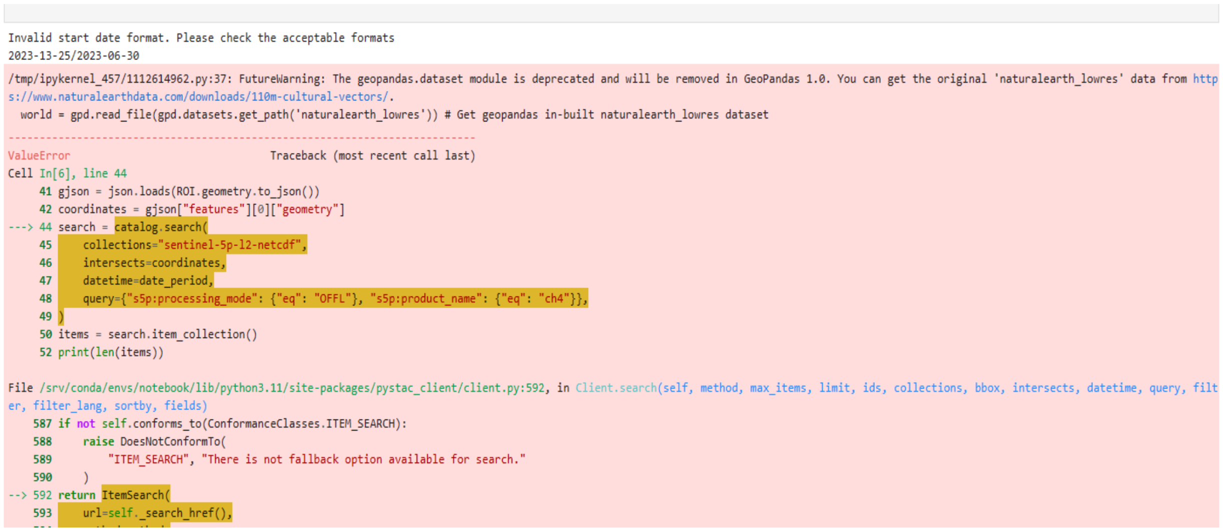
Input:



```
)
```

```
•[5]: start_date = "2023-13-25"
      end_date = "2023-06-30"
      bbox = [112.70505, -44.52755, 154.38241, -11.29524]
      country = "Australia"
```

Output:



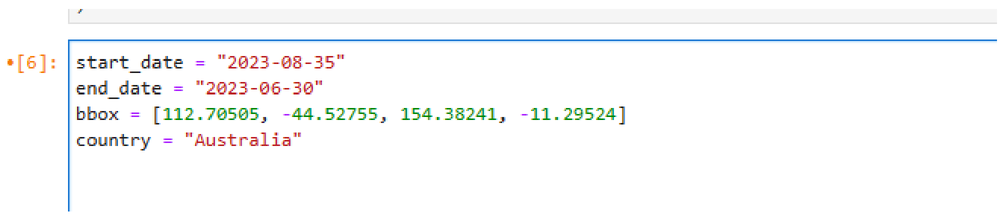
```
Invalid start date format. Please check the acceptable formats
2023-13-25/2023-06-30
/tmp/ipykernel_457/1112614962.py:37: FutureWarning: The geopandas.dataset module is deprecated and will be removed in GeoPandas 1.0. You can get the original 'naturalearth_lowres' data from http
s://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
  world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # Get geopandas in-built naturalearth_lowres dataset
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[6], line 44
     41 gjson = json.loads(ROI.geometry.to_json())
     42 coordinates = gjson["features"][0]["geometry"]
---> 44 search = catalog.search(
     45     collections="sentinel-5p-l2-netcdf",
     46     intersects=coordinates,
     47     datetime=date_period,
     48     query={"s5p:processing_mode": {"eq": "OFFL"}, "s5p:product_name": {"eq": "ch4"}},
     49 )
     50 items = search.item_collection()
     52 print(len(items))

File /srv/conda/envs/notebook/lib/python3.11/site-packages/pystac_client/client.py:592, in Client.search(self, method, max_items, limit, ids, collections, bbox, intersects, datetime, query, filt
er, filter_lang, sortby, fields)
    587 if not self.conforms_to(ConformanceClasses.ITEM_SEARCH):
    588     raise DoesNotConformTo(
    589         "ITEM_SEARCH", "There is not fallback option available for search."
    590     )
--> 592 return ItemSearch(
    593     url=self._search_href(),
```

Expected Output: Fail

Real Output: Fail

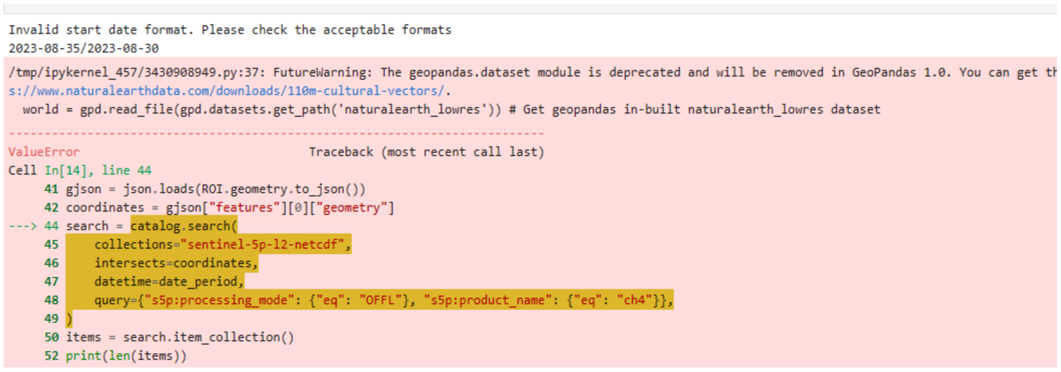
TestCase7: Invalid Input Format (Day 35 is out of range for August)

Input:



```
•[6]: start_date = "2023-08-35"
      end_date = "2023-06-30"
      bbox = [112.70505, -44.52755, 154.38241, -11.29524]
      country = "Australia"
```

Output:



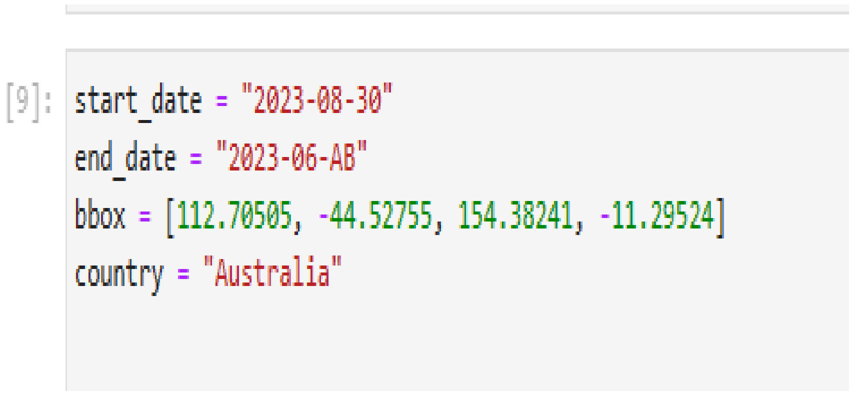
```
Invalid start date format. Please check the acceptable formats
2023-08-35/2023-08-30
/tmp/ipykernel_457/3430908949.py:37: FutureWarning: The geopandas.dataset module is deprecated and will be removed in GeoPandas 1.0. You can get th
s://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
  world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # Get geopandas in-built naturalearth_lowres dataset
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[14], line 44
     41 gjson = json.loads(ROI.geometry.to_json())
     42 coordinates = gjson["features"][0]["geometry"]
---> 44 search = catalog.search(
     45     collections="sentinel-5p-l2-netcdf",
     46     intersects=coordinates,
     47     datetime=date_period,
     48     query={"s5p:processing_mode": {"eq": "OFFL"}, "s5p:product_name": {"eq": "ch4"}},
     49 )
     50 items = search.item_collection()
     52 print(len(items))
```

Expected Output: Fail

Real Output: Fail

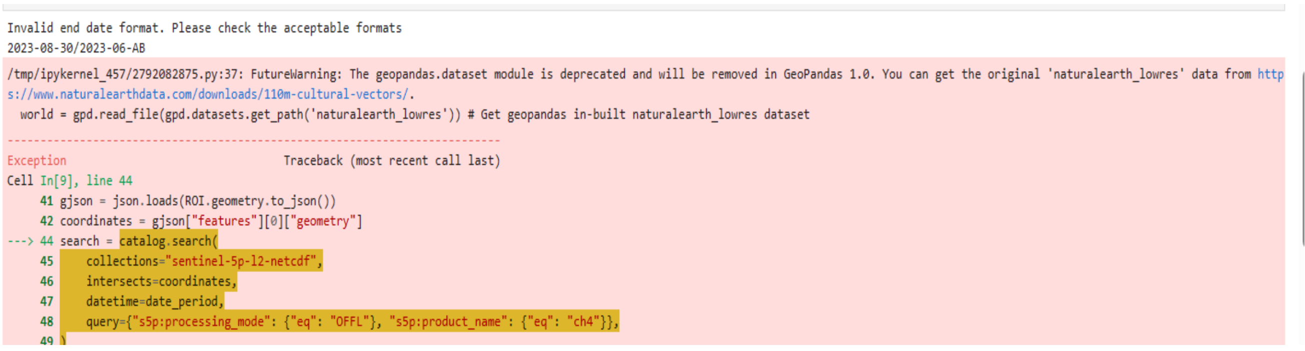
TestCase8: Invalid Input Format (Non-Numeric Characters)

Input:



```
[9]: start_date = "2023-08-30"
     end_date = "2023-06-AB"
     bbox = [112.70505, -44.52755, 154.38241, -11.29524]
     country = "Australia"
```

Output:



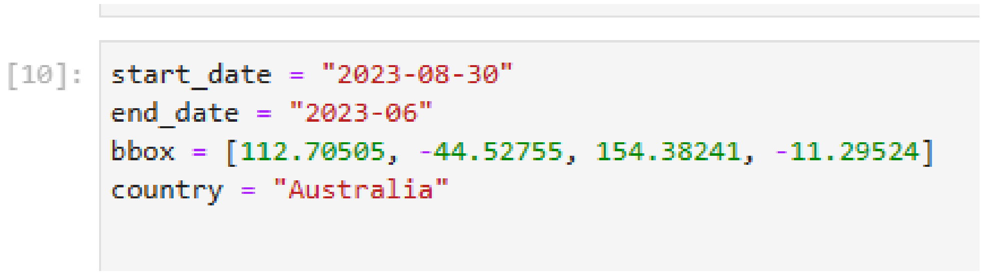
```
Invalid end date format. Please check the acceptable formats
2023-08-30/2023-06-AB
/tmp/ipykernel_457/2792082875.py:37: FutureWarning: The geopandas.dataset module is deprecated and will be removed in GeoPandas 1.0. You can get the original 'naturalearth_lowres' data from http
s://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
  world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # Get geopandas in-built naturalearth_lowres dataset
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Cell In[9], line 44
     41 gjson = json.loads(ROI.geometry.to_json())
     42 coordinates = gjson["features"][0]["geometry"]
---> 44 search = catalog.search(
     45     collections="sentinel-5p-l2-netcdf",
     46     intersects=coordinates,
     47     datetime=date_period,
     48     query={"s5p:processing_mode": {"eq": "OFFL"}, "s5p:product_name": {"eq": "ch4"}},
     49 )
```

Expected Output: Fail

Real Output: Fail

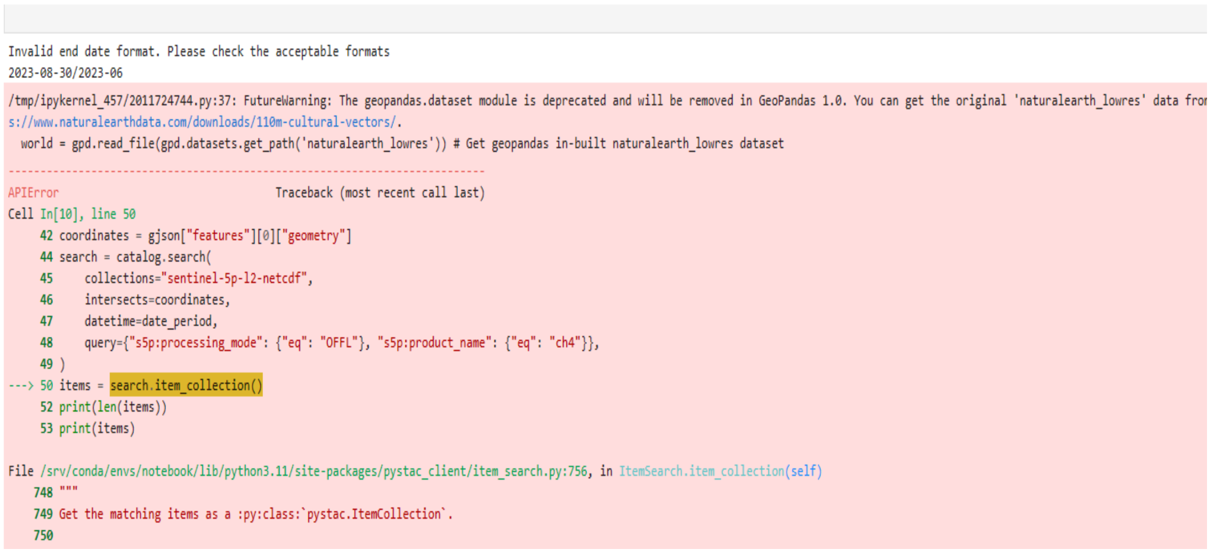
TestCase 9: Invalid Input Format (Incomplete Date)

Input:



```
[10]: start_date = "2023-08-30"
      end_date = "2023-06"
      bbox = [112.70505, -44.52755, 154.38241, -11.29524]
      country = "Australia"
```

Output:



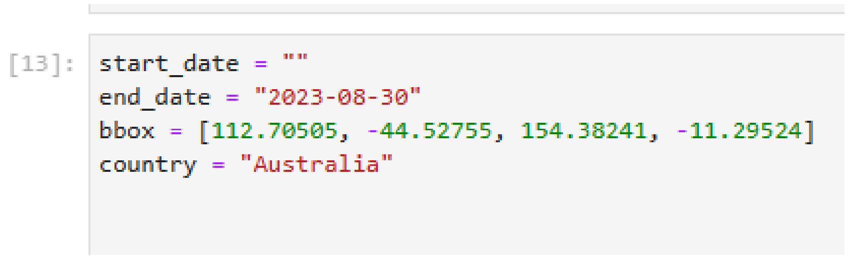
```
Invalid end date format. Please check the acceptable formats
2023-08-30/2023-06
/tmp/ipykernel_457/2011724744.py:37: FutureWarning: The geopandas.dataset module is deprecated and will be removed in GeoPandas 1.0. You can get the original 'naturalearth_lowres' data fro
s://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
  world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # Get geopandas in-built naturalearth_lowres dataset
---------------------------------------------------------------------------
APIError                                  Traceback (most recent call last)
Cell In[10], line 50
     42 coordinates = gjson["features"][0]["geometry"]
     44 search = catalog.search(
     45     collections="sentinel-5p-l2-netcdf",
     46     intersects=coordinates,
     47     datetime=date_period,
     48     query={"s5p:processing_mode": {"eq": "OFFL"}, "s5p:product_name": {"eq": "ch4"}},
     49 )
---> 50 items = search.item_collection()
     52 print(len(items))
     53 print(items)

File /srv/conda/envs/notebook/lib/python3.11/site-packages/pystac_client/item_search.py:756, in ItemSearch.item_collection(self)
    748 """
    749 Get the matching items as a :py:class:`pystac.ItemCollection`.
    750
```

Expected Output: Fail

Real Output: Fail

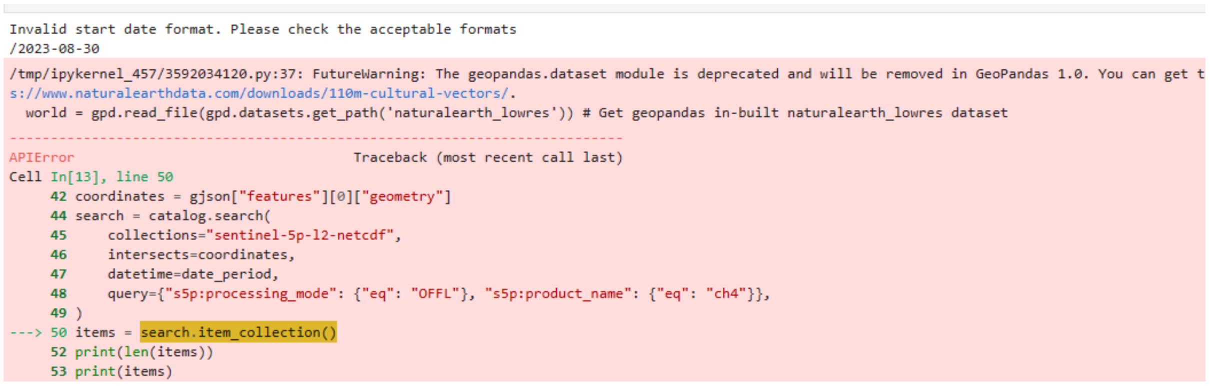
TestCase10: Invalid Input Format (Empty input date)

Input:



```
[13]: start_date = ""
      end_date = "2023-08-30"
      bbox = [112.70505, -44.52755, 154.38241, -11.29524]
      country = "Australia"
```

Output:



```
Invalid start date format. Please check the acceptable formats
/2023-08-30
/tmp/ipykernel_457/3592034120.py:37: FutureWarning: The geopandas.dataset module is deprecated and will be removed in GeoPandas 1.0. You can get t
s://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
  world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # Get geopandas in-built naturalearth_lowres dataset
---------------------------------------------------------------------------
APIError                                  Traceback (most recent call last)
Cell In[13], line 50
     42 coordinates = gjson["features"][0]["geometry"]
     44 search = catalog.search(
     45     collections="sentinel-5p-l2-netcdf",
     46     intersects=coordinates,
     47     datetime=date_period,
     48     query={"s5p:processing_mode": {"eq": "OFFL"}, "s5p:product_name": {"eq": "ch4"}},
     49 )
---> 50 items = search.item_collection()
     52 print(len(items))
     53 print(items)
```

Expected Output: Fail

Real Output: Fail

Conclusion:

We have extensively assessed the system's functioning against a wide range of test cases throughout the testing phase. These test cases include unit testing and integration testing. The system performed well in meeting both functional and non-functional requirements. All test cases passed successfully, and no defects were reported.