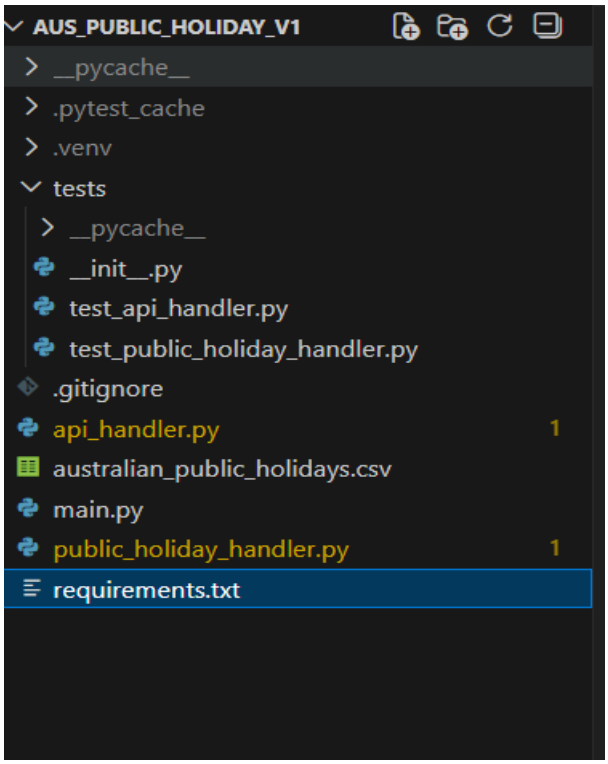
Python Coding Test Submission

- **Submitted by:** Venketesh Pratap Mall
- **Date:** February 4, 2026
- **GitHub Repository:** https://github.com/VenketeshPratap/aus_public_holiday_v2

Folder Structure-



- Running `python main.py`

```
(.venv) PS C:\Users\venke\OneDrive\Desktop\Python_Project\aus_public_holiday_v1> python main.py
   _id      Date              Holiday Name                                  Information                                More Information Jurisdiction
0    1  20230101           New Year's Day  New Year's Day is the first day of the calenda...  https://www.cmtedd.act.gov.au/communication/ho...          act
1    2  20230102  New Year's Day (observed)  As 1 January 2023 falls on a Sunday in 2023, t...  https://www.cmtedd.act.gov.au/communication/ho...          act
2    3  20230126             Australia Day                    Always celebrated on 26 January  https://www.cmtedd.act.gov.au/communication/ho...          act
3    4  20230313              Canberra Day  Held on the second Monday of March each year i...  https://www.cmtedd.act.gov.au/communication/ho...          act
4    5  20230407               Good Friday  Easter is celebrated with Good Friday and East...  https://www.cmtedd.act.gov.au/communication/ho...          act
(.venv) PS C:\Users\venke\OneDrive\Desktop\Python_Project\aus_public_holiday_v1>
```

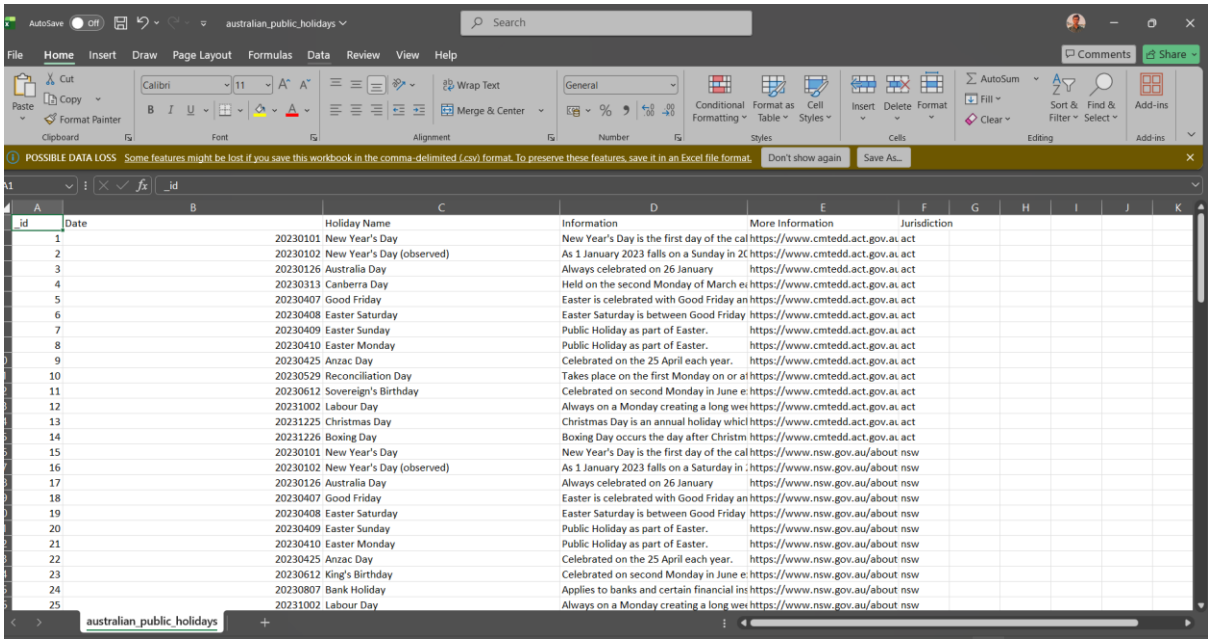
- Running `pytest -v`

```
(.venv) PS C:\Users\venke\OneDrive\Desktop\Python_Project\aus_public_holiday_v1> pytest -v
============================= test session starts =============================
platform win32 -- Python 3.12.6, pytest-9.0.2, pluggy-1.6.0 -- C:\Users\venke\OneDrive\Desktop\Python_Project\aus_public_holiday_v1\.venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\venke\OneDrive\Desktop\Python_Project\aus_public_holiday_v1
collected 5 items

tests/test_api_handler.py::test_api_handler_invalid_url PASSED           [ 20%]
tests/test_api_handler.py::test_api_handler_success PASSED               [ 40%]
tests/test_public_holiday_handler.py::test_dataframe_creation PASSED     [ 60%]
tests/test_public_holiday_handler.py::test_save_to_csv PASSED            [ 80%]
tests/test_public_holiday_handler.py::test_dataframe_columns PASSED      [100%]

============================== 5 passed in 4.01s ==============================
```

Saved the full table as CSV in the same folder.



POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format. Don't show again Save As...

id	Date	Holiday Name	Information	More Information	Jurisdiction
1	20230101	New Year's Day	New Year's Day is the first day of the cal	https://www.cmtedd.act.gov.au	act
2	20230102	New Year's Day (observed)	As 1 January 2023 falls on a Sunday in 2(	https://www.cmtedd.act.gov.au	act
3	20230126	Australia Day	Always celebrated on 26 January	https://www.cmtedd.act.gov.au	act
4	20230313	Canberra Day	Held on the second Monday of March ea	https://www.cmtedd.act.gov.au	act
5	20230407	Good Friday	Easter is celebrated with Good Friday an	https://www.cmtedd.act.gov.au	act
6	20230408	Easter Saturday	Easter Saturday is between Good Friday	https://www.cmtedd.act.gov.au	act
7	20230409	Easter Sunday	Public Holiday as part of Easter.	https://www.cmtedd.act.gov.au	act
8	20230410	Easter Monday	Public Holiday as part of Easter.	https://www.cmtedd.act.gov.au	act
9	20230425	Anzac Day	Celebrated on the 25 April each year.	https://www.cmtedd.act.gov.au	act
10	20230529	Reconciliation Day	Takes place on the first Monday on or af	https://www.cmtedd.act.gov.au	act
11	20230612	Sovereign's Birthday	Celebrated on second Monday in June e	https://www.cmtedd.act.gov.au	act
12	20231002	Labour Day	Always on a Monday creating a long wee	https://www.cmtedd.act.gov.au	act
13	20231225	Christmas Day	Christmas Day is an annual holiday whicl	https://www.cmtedd.act.gov.au	act
14	20231226	Boxing Day	Boxing Day occurs the day after Christm	https://www.cmtedd.act.gov.au	act
15	20230101	New Year's Day	New Year's Day is the first day of the cal	https://www.nsw.gov.au/about	nsw
16	20230102	New Year's Day (observed)	As 1 January 2023 falls on a Saturday in	https://www.nsw.gov.au/about	nsw
17	20230126	Australia Day	Always celebrated on 26 January	https://www.nsw.gov.au/about	nsw
18	20230407	Good Friday	Easter is celebrated with Good Friday an	https://www.nsw.gov.au/about	nsw
19	20230408	Easter Saturday	Easter Saturday is between Good Friday	https://www.nsw.gov.au/about	nsw
20	20230409	Easter Sunday	Public Holiday as part of Easter.	https://www.nsw.gov.au/about	nsw
21	20230410	Easter Monday	Public Holiday as part of Easter.	https://www.nsw.gov.au/about	nsw
22	20230425	Anzac Day	Celebrated on the 25 April each year.	https://www.nsw.gov.au/about	nsw
23	20230612	King's Birthday	Celebrated on second Monday in June e	https://www.nsw.gov.au/about	nsw
24	20230807	Bank Holiday	Applies to banks and certain financial ins	https://www.nsw.gov.au/about	nsw
25	20231002	Labour Day	Always on a Monday creating a long wee	https://www.nsw.gov.au/about	nsw

Api_handler.py

```python
import requests
from typing import Dict, Any


class APIHandler:
    """
    Handles communication with the Australian Government CKAN API.
    """

    def __init__(self, base_url: str):
        """
        Initialize APIHandler with base API URL.

        :param base_url: Base CKAN API URL
        """
        self.base_url = base_url.rstrip("/")

    def fetch_data(
        self, resource_id: str, limit: int
    ) -> Dict[str, Any]:
        """
        Fetch data from CKAN datastore_search endpoint.

        :param resource_id: CKAN resource ID
        :param limit: Number of rows to retrieve
        :return: JSON response
        """
        url = f"{self.base_url}/datastore_search"
        params = {
            "resource_id": resource_id,
            "limit": limit,
        }

        response = requests.get(url, params=params, timeout=10)

        if response.status_code != 200:
            raise RuntimeError("Failed to fetch data from API")

        data = response.json()

        if not data.get("success"):
            raise RuntimeError("API returned unsuccessful response")

        return data
```

public_holiday_handler.py

```python
import pandas as pd
from api_handler import APIHandler
class PublicHolidayHandler:
    """
    Prepares public holiday data for analysis.
    """
    def __init__(
        self,
        api_handler: APIHandler,
        resource_id: str,
        limit: int = 1000,
    ):
        """
        Initialize handler and retrieve public holiday data.
        :param api_handler: APIHandler instance
        :param resource_id: CKAN resource ID
        :param limit: Number of rows to fetch
        """
        try:
            self.raw_data = api_handler.fetch_data(
                resource_id=resource_id,
                limit=limit,
            )
        except Exception as exc:
            raise RuntimeError(
                "Failed to retrieve public holiday data"
            ) from exc

    def to_dataframe(self) -> pd.DataFrame:
        """
        Convert public holiday JSON data to Pandas DataFrame.

        :return: Pandas DataFrame
        """
        records = self.raw_data["result"]["records"]
        return pd.DataFrame(records)

    def save_to_csv(self, file_path: str) -> None:
        """
        Save public holiday data as CSV.

        :param file_path: Output CSV file path
        """
        df = self.to_dataframe()
        df.to_csv(file_path, index=False)
```

Main.py

```python
from api_handler import APIHandler
from public_holiday_handler import PublicHolidayHandler


def main():
    BASE_URL = "https://data.gov.au/data/api/action"
    RESOURCE_ID = "d256f989-8f49-46eb-9770-1c6ee9bd2661"

    api_handler = APIHandler(BASE_URL)

    holiday_handler = PublicHolidayHandler(
        api_handler=api_handler,
        resource_id=RESOURCE_ID,
        limit=1000,
    )

    df = holiday_handler.to_dataframe()

    # Print first 5 rows
    print(df.head())

    # Save full table as CSV
    holiday_handler.save_to_csv(
        "australian_public_holidays.csv"
    )


if __name__ == "__main__":
    main()
```

Unit- Test

Tests/test_api_handler.py

```python
import pytest
import requests
from api_handler import APIHandler


def test_api_handler_invalid_url():
    api_handler = API Handler("https://test-url")

    with pytest.raises(Exception):
        api_handler.fetch_data(
            resource_id="dummy",
            limit=5,
        )


def test_api_handler_success(monkeypatch):
    """
    Test that APIHandler returns JSON data on successful API call.
    """

    class MockResponse:
        status_code = 200

        def json(self):
            return {"success": True, "result": {"records": []}}

    def mock_get(*args, **kwargs):
        return MockResponse()

    monkeypatch.setattr(requests, "get", mock_get)

    api_handler = APIHandler("https://fake-url")
    data = api_handler.fetch_data("dummy", limit=5)

    assert data["success"] is True
```

tests/test_public_holiday_handler.py

```
import pandas as pd
from public_holiday_handler import PublicHolidayHandler


class MockAPIHandler:
    def fetch_data(self, resource_id, limit):
        return {
            "success": True,
            "result": {
                "records": [
                    {"date": "2024-01-01", "name": "New Year"},
                    {"date": "2024-01-26", "name": "Australia Day"},
                ]
            },
        }


def test_dataframe_creation():
    handler = PublicHolidayHandler(
        api_handler=MockAPIHandler(),
        resource_id="dummy",
        limit=2,
    )

    df = handler.to_dataframe()

    assert isinstance(df, pd.DataFrame)
    assert len(df) == 2


def test_save_to_csv(tmp_path):
    """
    Test that CSV file is created successfully.
    """

    class MockAPIHandler:
        def fetch_data(self, resource_id, limit):
            return {
                "success": True,
                "result": {
                    "records": [
                        {"date": "2024-01-01", "name": "New Year"}
                    ]
                },
            }

    handler = PublicHolidayHandler(
```

```
        api_handler=MockAPIHandler(),
        resource_id="dummy",
        limit=1,
    )

    file_path = tmp_path / "holidays.csv"
    handler.save_to_csv(str(file_path))

    assert file_path.exists()


def test_dataframe_columns():
    """
    Test that expected columns exist in DataFrame.
    """

    class MockAPIHandler:
        def fetch_data(self, resource_id, limit):
            return {
                "success": True,
                "result": {
                    "records": [
                        {"date": "2024-01-01", "name": "New Year"}
                    ]
                },
            }

    handler = PublicHolidayHandler(
        api_handler=MockAPIHandler(),
        resource_id="dummy",
        limit=1,
    )

    df = handler.to_dataframe()

    assert "date" in df.columns
    assert "name" in df.columns
```