Knowledge Transfer Document: Fetching Food Security Data Across Years Using the Explore API

Objective

In this document, we will learn how to fetch and combine "Food Security" data from the City of Melbourne's open data platform for the years 2018–2023. We will use a Python script to automate data retrieval via API, process the data, and save it as a single consolidated file for further analysis.

1. What is the Explore API?

The Explore API allows us to programmatically access datasets published by the City of Melbourne. Each dataset is identified by a unique `dataset_id`, and we can use API parameters to filter and query the data as needed.

2. Benefits of Using APIs

a) **Automation:** Automatically fetch and process data without manual downloads.
b) **Scalability:** Handle data for multiple years or categories efficiently.
c) **Dynamic Updates:** Always access the most recent data.
d) **Filter and Query:** Retrieve specific data based on keywords, categories, or timeframes.

3. Tools and Setup

The following libraries are used:
- requests: To fetch data from the API.
- pandas: To process and save data.

4. Script Workflow

To achieve our objective, we will follow these steps:

- **Define the API URL and Dataset IDs:** We will use a unique dataset_id for each year. The dataset_id can be found in the Information tab of the dataset.
- **Fetch Data for Each Year:** We will loop through the dataset IDs to construct API requests. Since we need 6-year survey dataset we can store all the unique (dataset_id)s in a list and can loop into it further.
- **Handle API Response:** We will validate the response and parse it into a usable format.
- **Filter for "Food Security" Records:** We will retain only the relevant rows based on the topic column. As we are working with only Food security records, we are filtering it. Alternatively, we can fetch full dataset, and further do the filtration in the code.

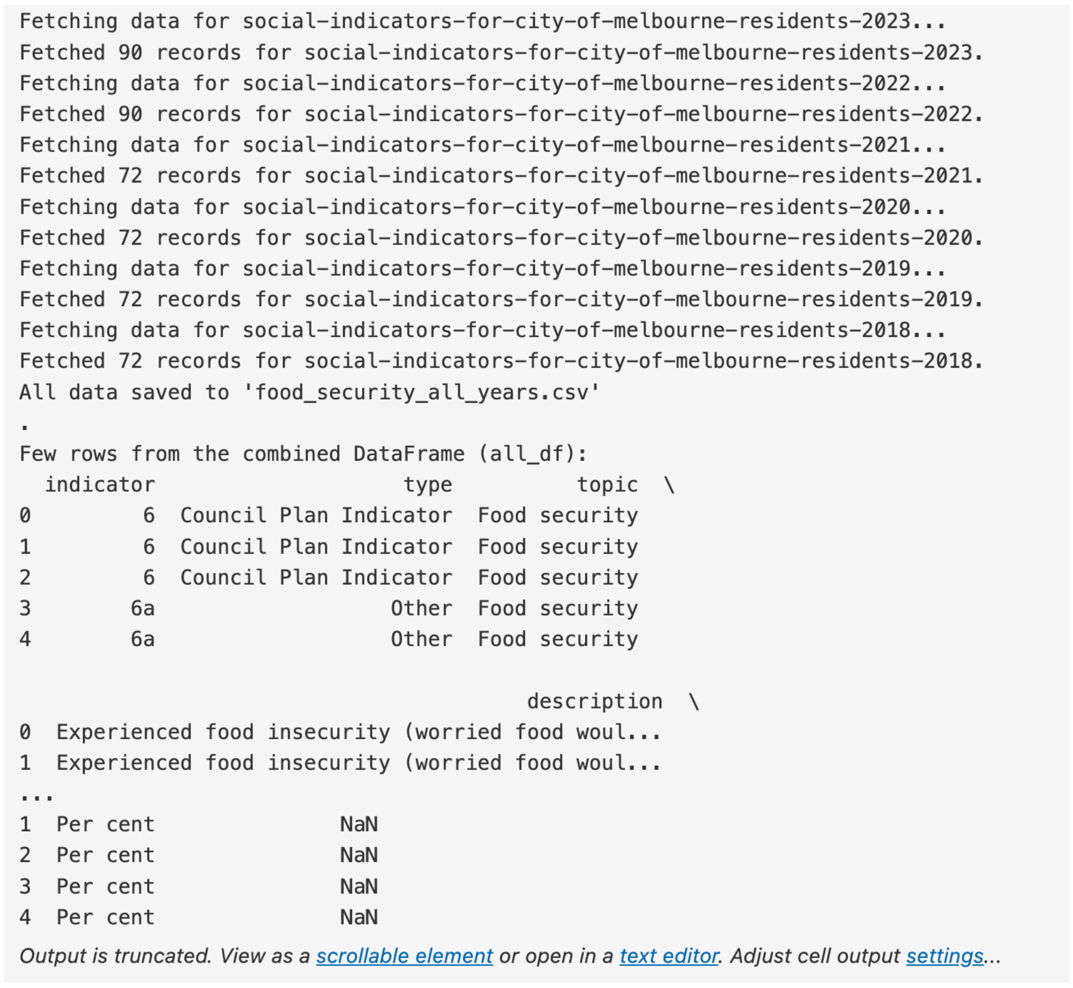
- **Combine Data:** We will append filtered records from all years into a single dataset.
- **Save Combined Data:** Finally, we will export the consolidated dataset as a CSV file.

5. Full Script

```
import requests

import pandas as pd

#the base URL for the Explore API v2

url_use =
"https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/{dataset_id}/exports/
json"

#list of dataset IDs for each year

DATASET_IDS = [

    "social-indicators-for-city-of-melbourne-residents-2023",

    "social-indicators-for-city-of-melbourne-residents-2022",

    "social-indicators-for-city-of-melbourne-residents-2021",

    "social-indicators-for-city-of-melbourne-residents-2020",

    "social-indicators-for-city-of-melbourne-residents-2019",

    "social-indicators-for-city-of-melbourne-residents-2018"

]

#parameters for the api call

all_years_data = []  # For storing all records across years

#fetching data through each dataset ID

for dataset_id in DATASET_IDS:

    print(f"Fetching data for {dataset_id}...")

    #constructing the API request URL

    dataset_url = url_use.format(dataset_id=dataset_id)

    #apiCall
    response = requests.get(dataset_url)

    if response.status_code != 200:

        print(f"Error: Unable to fetch data for {dataset_id}. HTTP {response.status_code}")

        continue

    #parsing json response

    data = response.json()

    #checking if response empty

    if not data:

        print(f"No data found for dataset {dataset_id}.")

        continue

    #converting json into datframe

    df = pd.DataFrame(data)

    #filtering for "Food security" in the topic column as I will be working with those
records only

    if "topic" in df.columns:

        df = df[df["topic"].str.contains("Food security", na=False)]

    #appending the filtered data to the list

    all_years_data.extend(df.to_dict("records"))

    print(f"Fetched {len(df)} records for {dataset_id}.")

#converting the combined data to a DataFrame

all_df = pd.DataFrame(all_years_data)

#saving the combined dataframe to a csv file

all_df.to_csv("food_security_all_years.csv", index=False)

print("All data saved to 'food_security_all_years.csv'\n.")

#display
```

```
print("Few rows from the combined DataFrame (all_df):")

print(all_df.head())
```

Output:



```
Fetching data for social-indicators-for-city-of-melbourne-residents-2023...
Fetched 90 records for social-indicators-for-city-of-melbourne-residents-2023.
Fetching data for social-indicators-for-city-of-melbourne-residents-2022...
Fetched 90 records for social-indicators-for-city-of-melbourne-residents-2022.
Fetching data for social-indicators-for-city-of-melbourne-residents-2021...
Fetched 72 records for social-indicators-for-city-of-melbourne-residents-2021.
Fetching data for social-indicators-for-city-of-melbourne-residents-2020...
Fetched 72 records for social-indicators-for-city-of-melbourne-residents-2020.
Fetching data for social-indicators-for-city-of-melbourne-residents-2019...
Fetched 72 records for social-indicators-for-city-of-melbourne-residents-2019.
Fetching data for social-indicators-for-city-of-melbourne-residents-2018...
Fetched 72 records for social-indicators-for-city-of-melbourne-residents-2018.
All data saved to 'food_security_all_years.csv'
.
Few rows from the combined DataFrame (all_df):
  indicator                    type          topic  \
0         6  Council Plan Indicator  Food security
1         6  Council Plan Indicator  Food security
2         6  Council Plan Indicator  Food security
3        6a                   Other  Food security
4        6a                   Other  Food security

                                         description  \
0  Experienced food insecurity (worried food woul...
1  Experienced food insecurity (worried food woul...
...
1  Per cent                NaN
2  Per cent                NaN
3  Per cent                NaN
4  Per cent                NaN
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

6. Challenges and Solutions

Challenge	Solution
HTTP Errors	We should check the API URL and dataset ID. If an error occurs, we will retry the request after a delay.
Empty Responses	We will log and skip datasets with no available data.
Missing Columns	We need to verify the dataset schema to ensure compatibility with our filtering logic.
Large Datasets	We can use pagination if the dataset is too large (not needed in this script).

More Information:

https://data.melbourne.vic.gov.au/api/explore/v2.1/console

https://help.opendatasoft.com/apis/ods-explore-v2/

Always check API documentation for available parameters and rate limits.