

# ODSQL & API v2.1 Tutorial 2024

This document explains how to use the API of City of Melbourne Open Data (CoM). Only the HTTP (GET) method is supported, all API endpoints return JSON. Endpoints use [ODSQL](#).

## Explore API v2.1 OAS3

</api/explore/v2.1/swagger.json>

The Opendatasoft Explore API v2 is organized around REST. It provides access to all the data available through the platform in a coherent, hierarchical way.

Only the HTTP **GET** method is supported.

All API endpoints return JSON.

Endpoints are organized in a hierarchical way describing the relative relationship between objects.

All responses contain a list of links allowing easy and relevant navigation through the API endpoints.

All endpoints use the [Opendatasoft Query Language \(ODSQL\)](#). This means that, most of the time, parameters work the same way for all endpoints.

While the **records** endpoint is subject to a **limited number of returned records**, the **exports** endpoint has no limitations.

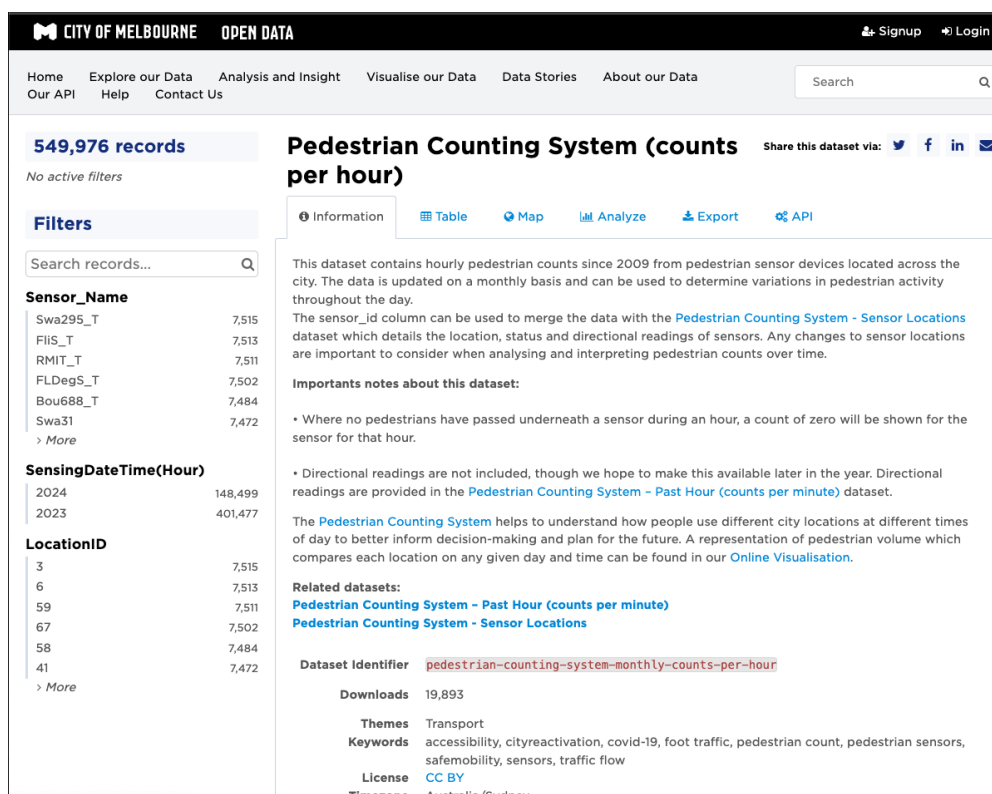
[Contact the developer](#)

## Note

- These instructions only apply to **API v2.1** and the [City of Melbourne Open Data](#).
- **records** endpoint is subject to a [limited number of returned records](#)
- **exports** endpoints has no limitations!
- Each organization's API settings are different!

*Some parts of the documentation, such as loop acquisition. The government may ban it in subsequent updates, and it does not apply to team API keys with advanced permissions. For commercial enterprises, loop acquisition is likely to be disabled and may be defaulted by the system as an attack or illegal request.*

## Explore Datasets



The screenshot shows the City of Melbourne Open Data website. The header includes the City of Melbourne logo, 'OPEN DATA', and links for Signup and Login. The navigation bar has links for Home, Explore our Data, Analysis and Insight, Visualise our Data, Data Stories, and About our Data. A search bar is also present.

The main content area displays the 'Pedestrian Counting System (counts per hour)' dataset. It shows 549,976 records and no active filters. The left sidebar contains a 'Filters' section with a search bar and a list of filters: Sensor\_Name, SensingDateTime(Hour), and LocationID. The main content area includes a description of the dataset, important notes, related datasets, and dataset metadata.

**Pedestrian Counting System (counts per hour)**

This dataset contains hourly pedestrian counts since 2009 from pedestrian sensor devices located across the city. The data is updated on a monthly basis and can be used to determine variations in pedestrian activity throughout the day.

The sensor\_id column can be used to merge the data with the [Pedestrian Counting System - Sensor Locations](#) dataset which details the location, status and directional readings of sensors. Any changes to sensor locations are important to consider when analysing and interpreting pedestrian counts over time.

**Important notes about this dataset:**

- Where no pedestrians have passed underneath a sensor during an hour, a count of zero will be shown for the sensor for that hour.
- Directional readings are not included, though we hope to make this available later in the year. Directional readings are provided in the [Pedestrian Counting System - Past Hour \(counts per minute\)](#) dataset.

The [Pedestrian Counting System](#) helps to understand how people use different city locations at different times of day to better inform decision-making and plan for the future. A representation of pedestrian volume which compares each location on any given day and time can be found in our [Online Visualisation](#).

**Related datasets:**

- [Pedestrian Counting System - Past Hour \(counts per minute\)](#)
- [Pedestrian Counting System - Sensor Locations](#)

**Dataset Identifier:** `pedestrian-counting-system-monthly-counts-per-hour`

**Downloads:** 19,893

**Themes:** Transport

**Keywords:** accessibility, cityreactivation, covid-19, foot traffic, pedestrian count, pedestrian sensors, safemobility, sensors, traffic flow

**License:** CC BY

**Timezone:** Australia/Sydney

Dataset [Pedestrian Counting System \(counts per hour\)](#)

## Appy for Personal API Key

**Do not share your API key, keep it secret.**

[Instructions on applying for a API key](#)

## API Keys

```
API_KEY = os.environ.get('MELBOURNE_API_KEY', input('Please enter your API key: '))
BASE_URL = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/'
```

**DO NOT EXPOSE YOUR API KEY – IT IS A SECURITY RISK!**

### Option 1. Remove API KEY Before Publishing

**Risky you could forget!**

```
def fetch_data(base_url, dataset, api_key, num_records=99, offset=0):
    all_records = []
    max_offset = 9900

    while True:
        if offset > max_offset:
            break

        filters = f'{dataset}/records?limit={num_records}&offset={offset}'
        url = f'{base_url}{filters}&api_key={api_key}'

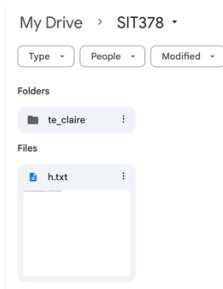
        try:
            result = requests.get(url, timeout = 10)
            result.raise_for_status()
            records = result.json().get('results')
        except requests.exceptions.RequestException as e:
            raise Exception(f'API request failed: {e}')
        if records is None:
            break
        all_records.extend(records)
        if len(records) < num_records:
            break

        offset += num_records

    df = pd.DataFrame(all_records)
    return df

BASE_URL = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/'
API_KEY = ''
```

### Option 2. Pull API key from Separate File (Locally or GitHub)



```
1 # Dependencies
2 import warnings
3 warnings.filterwarnings("ignore")
4 pd.set_option('display.max_columns', None)
5
6 import requests
7 import numpy as np
8 import pandas as pd

▼ Cloud or Local IDE (Run notebook/ script)

• To collect API from directory located in Google Collab

[ ] 1 from google.colab import drive
    2 drive.mount('/content/drive')
    3 with open('/content/drive/My Drive/SIT378/h.txt', 'r') as file:
    4     api_key = file.read().strip()
    5
    6 import os
    7 api_key = os.getenv(api_key)
```

Mounted at /content/drive

Load Dependencies and load Cloud (Google Collab)

### Option 3. Using Google Cloud Secret Manager

```
1 from google.cloud import secretmanager
2 client = secretmanager.SecretManagerServiceClient()
3 name = f"projects/your-project-id/secrets/your-secret-name/versions/latest"
4 response = client.access_secret_version(request={"name": name})
5 api_key = response.payload.data.decode("UTF-8")
```

### Pre-fill parameters (Optional: filtering)

Opendatasoft Query Language (ODSQL) allows you to filter datasets or records and build aggregations.

[https://help.opendatasoft.com/apis/ods-explore-v2/#section/Opendatasoft-Query-Language-\(ODSQL\)](https://help.opendatasoft.com/apis/ods-explore-v2/#section/Opendatasoft-Query-Language-(ODSQL))

Using the Console you can prefill your parameters and create a GET curl request, you can then use Python HTTP Request (Import requests) library to call API by interacting via HTTP methods.

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

**Catalog** API to enumerate datasets

- GET /catalog/datasets Query catalog datasets
- GET /catalog/exports List export formats
- GET /catalog/exports/{format} Export a catalog
- GET /catalog/exports/csv Export a catalog in CSV
- GET /catalog/exports/dcat(dcat\_ap\_format) Export a catalog in RDF/XML (DCAT)
- GET /catalog/facets List facet values
- GET /catalog/datasets/{dataset\_id} Show dataset information

Returns a list of available endpoints for the specified dataset, with metadata and endpoints.  
The response includes the following links:  
the attachments endpoint  
the files endpoint  
the records endpoint  
the catalog endpoint.

**Parameters** Cancel

Name	Description
<b>dataset_id</b> <span>required</span>	The identifier of the dataset to be queried. You can find it in the "Information" tab of the dataset page or in the dataset URL, right after /datasets/.
string (path)	pedestrian-counting-system-monthly-counts-per-hour
<b>select</b>	Examples: select-size - Example of select, which only return the "size" field. select-size * 2 as bigger_size - Example of a complex expression with a label, which returns a new field named "bigger_size" and containing the double of size field value. A select expression can be used to add, remove or change the fields to return. An expression can be: a wildcard ("*"): all fields are returned. A field name: only the specified field is returned. An include/exclude function: All fields matching the include or exclude expression are included or excluded. This expression can contain wildcard. A complex expression: The result of the expression is returned. A label can be set for this expression, and in that case, the field will be named after this label.
string (query)	*
<b>lang</b>	A language value.
string (query)	If specified, the lang value override the default language, which is "fr". The language is used to format string, for example in the date_format function.
en	
<b>timezone</b>	Set the timezone for datetime fields. Timezone IDs are defined by the <a href="#">Unicode CLDR project</a> . The list of timezone IDs is available in <a href="#">timezone.xml</a> .
string (query)	Examples: UTC timezone
UTC	
<b>include_links</b>	If set to true, this parameter will add HATEOAS links in the response.
boolean (query)	false
<b>include_app metas</b>	If set to true, this parameter will add application metadata to the response.
boolean (query)	false

Execute Clear

The ODSQL is split into five different kinds of clauses:

- The **select** clause allows choosing the returned fields, giving them an alias, manipulating them with functions like count, sum, min, max, etc.
- The **where** clause acts as a filter for the returned datasets or records, thanks to boolean operations, filter functions, arithmetic expressions, etc.
- The **group by** clause allows aggregating rows together based on fields, numeric ranges, or dates.
- The **order by** and **limit** clauses allow choosing the order and quantity of rows received as a response.

include\_app\_metas If set to true, this parameter will add application metadata to the response.

boolean (query) false

Execute Clear

**Responses**

Curl

```
curl -X GET \
  'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/pedestrian-counting-system-monthly-counts-per-hour?select=*&lang=en&timezone=UTC' \
  -H 'accept: application/json; charset=utf-8'
```

Request URL

```
https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/pedestrian-counting-system-monthly-counts-per-hour?select=*&lang=en&timezone=UTC&include_links=false&include_app_metas=false
```

Server response

Code Details

CoM API endpoints: To interact with a data catalog.

- Endpoints allow you to enumerate datasets
- List export formats
- Export data
- List facet values
- Manage individual dataset records

Catalog API

- GET [/catalog/datasets](#)
- **Purpose** To list all the datasets available in the catalog
  - Used to get an overview of the datasets available in the system
- GET [/catalog/exports](#)
- **Purpose** To list all export formats that the catalog supports
  - Useful for understanding what formats the data can be exported (CSV, JSON)
- GET [/catalog/exports/{format}](#)
- **Purpose** To export the entire catalog in a specific format
  - Used when you want to download the entire catalog in one of the supported formats
- GET [/catalog/exports/csv](#)
- **Purpose** Specifically for exporting the catalog in CSV format.
  - A direct endpoint for exporting data in a common, easily usable format.
- GET [/catalog/exports/dcat{dcat\\_ap\\_format}](#)
- **Purpose** To export the catalog in RDF/XML format using DCAT
  - Exporting data in a format that's suitable for integrating with other data catalogs or systems following the DCAT standard
- GET [/catalog/facets](#)
- **Purpose** To list all the facet values available in the catalog
  - Facets are used to filter or categorize datasets/ helps understand the categorization
- GET [/catalog/datasets/{dataset\\_id}](#)
- **Purpose** To show detailed information about a specific dataset
  - When you need metadata or details about a particular dataset

Dataset API

- GET [/catalog/datasets/{dataset\\_id}/records](#)
- **Purpose** To query records within a specific dataset
  - To retrieve the data entries or records from a specific dataset
- GET [/catalog/datasets/{dataset\\_id}/exports](#)
- **Purpose** To list the export formats available for a specific dataset
- Understands in what formats you can export the data from this dataset
- GET [/catalog/datasets/{dataset\\_id}/exports/{format}](#)
- **Purpose** To export a specific dataset in a specified format
  - To download data from a specific dataset in a particular format
- GET [/catalog/datasets/{dataset\\_id}/exports/csv](#)
- **Purpose** To export a specific dataset in CSV format
  - Direct endpoint for exporting dataset data in CSV, a commonly used data format
- GET [/catalog/datasets/{dataset\\_id}/exports/gpx](#)
- **Purpose** To export a specific dataset in GPX format
  - Useful for datasets related to geographical data, which GPX format is well-suited for
- GET [/catalog/datasets/{dataset\\_id}/facets](#)
- **Purpose** To list the facets for a specific dataset
  - To get an understanding of the different dimensions or categories within a dataset
- GET [/catalog/datasets/{dataset\\_id}/attachments](#)
- **Purpose** To list attachments for a specific dataset
  - When datasets have additional files or documents attached, this endpoint lets you enumerate them
- GET [/catalog/datasets/{dataset\\_id}/records/{record\\_id}](#)
- **Purpose** To read a specific record within a dataset
  - To get detailed information about a particular entry or record in a dataset

## 1. exports API (Endpoint)

\*Preferred method by CoM as it has no limitations

Example: Single Request for CSV file

GET/Catalog/exports

GET <https://data.melbourne.vic.gov.au/api/catalog/datasets/pedestrian-counting-system-monthly-counts-per-hour>

*This can take a little while as you are pulling the whole dataset*

Feel free to add a [progress bar](#) etc

Export Endpoint

Single Request for CSV File Download

GET/catalog/exports/catalog/datasets/

- ODSQL Function Export CSV or json\_format
- Read response directly into dataframe
- `response.content.decode('utf-8')` converts binary response into UTF-8 string (encoded)
- Data uses a delimiter (;)

```

1 import requests
2 import pandas as pd
3 from io import StringIO
4
5 # https://data.melbourne.vic.gov.au/explore/dataset/pedestrian-counting-system-monthly-counts-per-hour/information/
6 dataset_id = 'pedestrian-counting-system-monthly-counts-per-hour'
7
8 base_url = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/'
9 apikey = api_key
10 dataset_id = dataset_id
11 format = 'csv'
12
13 url = f'{base_url}{dataset_id}/exports/{format}'
14 params = {
15     'select': '*',
16     'limit': -1, # all records
17     'lang': 'en',
18     'timezone': 'UTC',
19     'api_key': apikey
20 }
21
22 # GET request
23 response = requests.get(url, params=params)
24
25 if response.status_code == 200:
26     # StringIO to read the CSV data
27     url_content = response.content.decode('utf-8')
28     melb_df = pd.read_csv(StringIO(url_content), delimiter=';')
29     print(melb_df.sample(10, random_state=999)) # Test
30 else:
31     print(f'Request failed with status code {response.status_code}')

```

	sensor_name	timestamp	locationid	direction_1	\
299310	Col700_T	2023-06-11T00:00:00+00:00	9	72	
273077	SprFli_T	2024-01-16T19:00:00+00:00	75	30	
230538	Bou688_T	2023-08-22T21:00:00+00:00	58	794	
545967	Eli501_T	2024-03-08T17:00:00+00:00	49	24	
41658	BouHbr_T	2023-07-06T06:00:00+00:00	10	275	
311686	BouBri_T	2023-06-03T18:00:00+00:00	57	2	
503749	ACMI_T	2024-02-27T12:00:00+00:00	72	10	
362060	Col12_T	2023-12-08T17:00:00+00:00	18	471	
90696	AG_T	2023-09-29T13:00:00+00:00	29	71	
340770	ACMI_T	2023-08-06T06:00:00+00:00	72	352	

Check number of records in dataset (dataset\_id)

```
[40] 1 num_records = len(melb_df)
      2 print(f'The dataset contains {num_records} records.')
```

The dataset contains 549976 records.

1 melb\_df.head()

	sensor_name	timestamp	locationid	direction_1	direction_2	total_of_directions	location
0	SprFiI_T	2023-04-24T21:00:00+00:00	75	36	17	53	-37.81515276, 144.97467661
1	SprFiI_T	2023-04-25T00:00:00+00:00	75	28	50	78	-37.81515276, 144.97467661
2	SprFiI_T	2023-04-25T01:00:00+00:00	75	63	63	126	-37.81515276, 144.97467661
3	SprFiI_T	2023-04-25T02:00:00+00:00	75	85	89	174	-37.81515276, 144.97467661
4	SprFiI_T	2023-04-25T08:00:00+00:00	75	365	59	424	-37.81515276, 144.97467661

## Additional examples exports endpoint

**Catalog (API to enumerate datasets) or Dataset (API to work on records)**

Example: Catalog API to enumerate datasets

GET/Catalog/datasets

GET <https://data.melbourne.vic.gov.au/api/catalog/datasets>

Limit 10, this can be commented out

```
[ ] 1 import requests
      2 url = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets'
      3 params = {
      4     'select': '*',
      5     'limit': 10,
      6     'offset': 0,
      7     'timezone': 'UTC',
      8     'include_links': 'false',
      9     'include_app_metas': 'false'
     10 }
     11 headers = {
     12     'accept': 'application/json; charset=utf-8'
     13 }
     14
     15 # GET request
     16 response = requests.get(url, headers=headers, params=params)
     17
     18 if response.status_code == 200: # Status code Check
     19     # Successful
     20     print(response.json())
     21 else:
     22     # Error
     23     print(f'Request failed with status code {response.status_code}')
     24
```

```
{'total_count': 228, 'results': [{'visibility': 'domain', 'dataset_id': 'city-of-melbourne-floor-space-forecasts-by-small-area-2020-2040', 'dataset_uid': 'da_mvdwug', 'has_records': True}]}
```

*Example: Show dataset Information*

GET/Catalog/datasets/{dataset\_id}

`GET <https://data.melbourne.vic.gov.au/api/catalog/datasets/pedestrian-counting-system-monthly-counts-per-hour>`

```
[1] 1 # dataset_id
2 # https://data.melbourne.vic.gov.au/explore/dataset/pedestrian-counting-system-monthly-counts-per-hour/information/
3 dataset_id = 'pedestrian-counting-system-monthly-counts-per-hour'

1 import requests
2 url = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/' + dataset_id
3 # or use full URL
4 # https://data.melbourne.vic.gov.au/explore/dataset/pedestrian-counting-system-monthly-counts-per-hour/information/
5
6 params = {
7     'select': '*',
8     'lang': 'en',
9     'timezone': 'UTC',
10    'include_links': 'false',
11    'include_app_metas': 'false'
12 }
13 headers = {
14     'accept': 'application/json; charset=utf-8'
15 }
16
17 # Make the GET request
18 response = requests.get(url, headers=headers, params=params)
19 if response.status_code == 200:
20     # Successful
21     print(response.json())
22 else:
23     # Error
24     print(f'Request failed with status code {response.status_code}')
25
```

{'visibility': 'domain', 'dataset\_id': 'pedestrian-counting-system-monthly-counts-per-hour', 'dataset\_uid': 'da\_3k86ps', 'has\_records': True, 'features': ['geo', 'analyze', 'timeserie'],

*Limit parameter controls the number of records or datasets returned in the response.*

*Example: Check Available export formats for dataset*

GET/Catalog/exports

`GET <https://data.melbourne.vic.gov.au/api/catalog/datasets/pedestrian-counting-system-monthly-counts-per-hour>`

```
[14] 1 import requests
2 url = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/exports'
3 params = {
4     'select': '*',
5     'lang': 'en',
6     'timezone': 'UTC',
7     'include_links': 'false',
8     'include_app_metas': 'false'
9 }
10 headers = {
11     'accept': 'application/json; charset=utf-8'
12 }
13
14 # Make the GET request
15 response = requests.get(url, headers=headers, params=params)
16 if response.status_code == 200:
17     # Successful
18     print(response.json())
19 else:
20     # Error
21     print(f'Request failed with status code {response.status_code}')
22
```

., 'href': 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/exports/dcat', {'rel': 'rss', 'href': 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/exports/rss'},

```
{dataset}/records?limit={num_records}&offset={offset}
```

Te' Claire 2024.v1