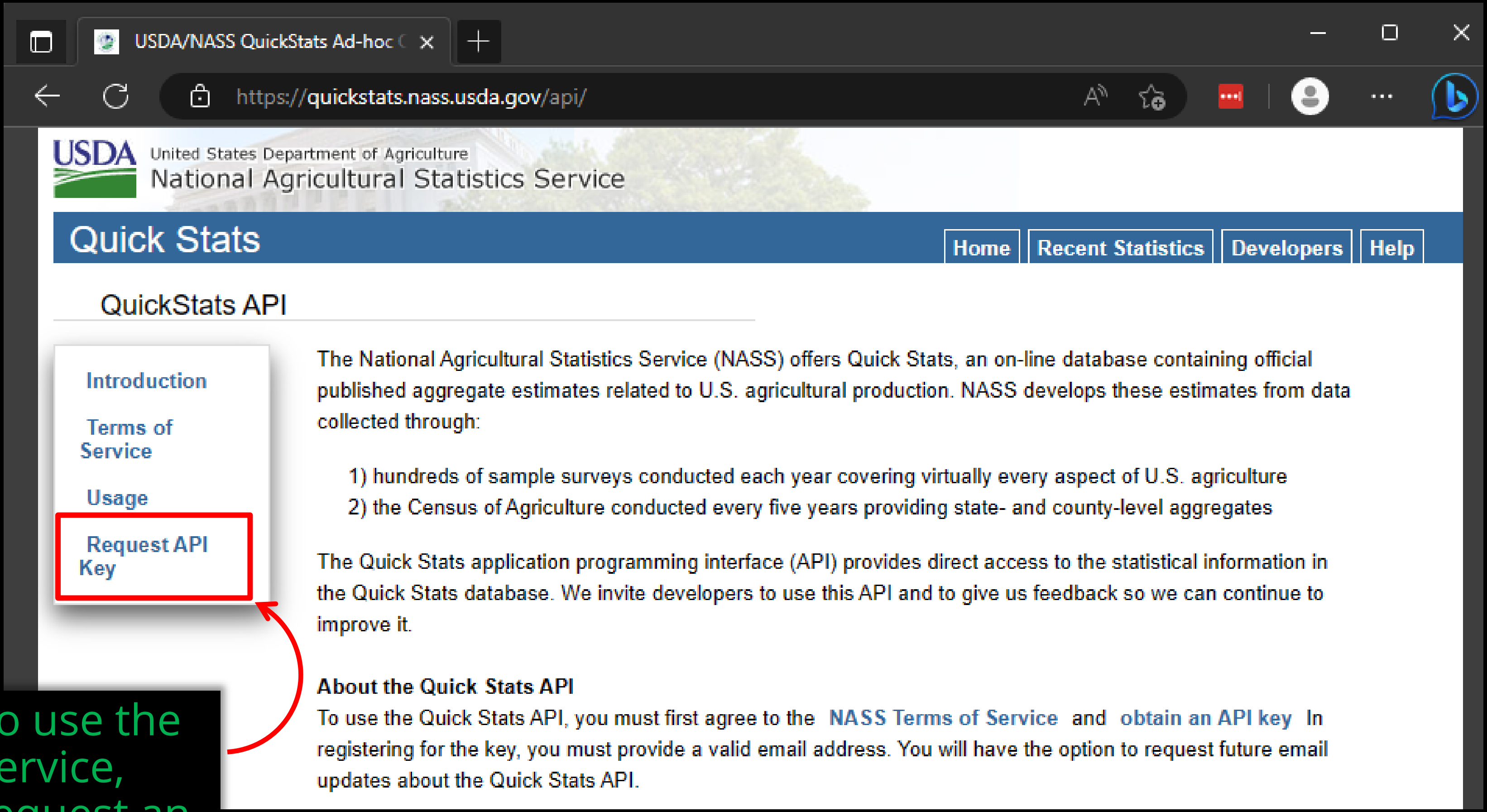
Power Query (M): Querying agriculture data from the *USDA NASS Quick Stats API*

The United States Department of Agriculture National Agricultural Statistics Service provides access to public data via their Quick Stats API


USDA/NASS QuickStats Ad-hoc
https://quickstats.nass.usda.gov/api/
USDA United States Department of Agriculture
National Agricultural Statistics Service
Quick Stats
Home
Recent Statistics
Developers
Help
Introduction
Terms of Service
Usage
Request API Key

QuickStats API

The National Agricultural Statistics Service (NASS) offers Quick Stats, an on-line database containing official published aggregate estimates related to U.S. agricultural production. NASS develops these estimates from data collected through:

1) hundreds of sample surveys conducted each year covering virtually every aspect of U.S. agriculture
2) the Census of Agriculture conducted every five years providing state- and county-level aggregates

The Quick Stats application programming interface (API) provides direct access to the statistical information in the Quick Stats database. We invite developers to use this API and to give us feedback so we can continue to improve it.

About the Quick Stats API

To use the Quick Stats API, you must first agree to the NASS Terms of Service and obtain an API key In registering for the key, you must provide a valid email address. You will have the option to request future email updates about the Quick Stats API.

To use the service, request an API key

There are three resources available to use

1. *GET /api/get_param_values – this will retrieve the domain of a parameter – i.e. the unique values that are valid for that column*

2. *GET /api/api_GET – this will retrieve a dataset from the API according to the parameters you provide. There is a 50k row limit.*

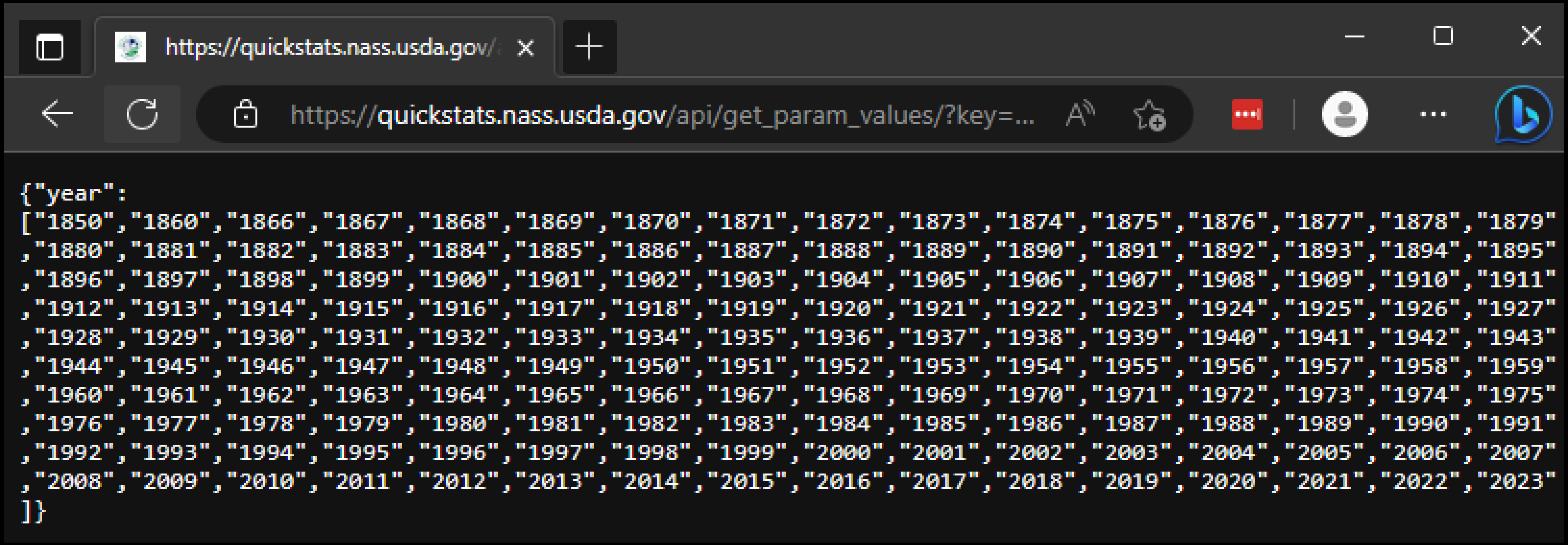
3. *GET /api/get_counts – this will return a row count for a query to be passed to the api_GET resource*

The simplest of these is *get_param_values*

We pass a column name as a parameter and receive a list of unique values in that column.

E.g. get a list of unique years available in the service, (replacing the "X" with a requested key)

```
https://quickstats.nass.usda.gov/api/get_param_values/?key=X&param=year
```



https://quickstats.nass.usda.gov/api/get_param_values/?key=...

```
{"year":
["1850","1860","1866","1867","1868","1869","1870","1871","1872","1873","1874","1875","1876","1877","1878","1879"
,"1880","1881","1882","1883","1884","1885","1886","1887","1888","1889","1890","1891","1892","1893","1894","1895"
,"1896","1897","1898","1899","1900","1901","1902","1903","1904","1905","1906","1907","1908","1909","1910","1911"
,"1912","1913","1914","1915","1916","1917","1918","1919","1920","1921","1922","1923","1924","1925","1926","1927"
,"1928","1929","1930","1931","1932","1933","1934","1935","1936","1937","1938","1939","1940","1941","1942","1943"
,"1944","1945","1946","1947","1948","1949","1950","1951","1952","1953","1954","1955","1956","1957","1958","1959"
,"1960","1961","1962","1963","1964","1965","1966","1967","1968","1969","1970","1971","1972","1973","1974","1975"
,"1976","1977","1978","1979","1980","1981","1982","1983","1984","1985","1986","1987","1988","1989","1990","1991"
,"1992","1993","1994","1995","1996","1997","1998","1999","2000","2001","2002","2003","2004","2005","2006","2007"
,"2008","2009","2010","2011","2012","2013","2014","2015","2016","2017","2018","2019","2020","2021","2022","2023"
]}
```

We can access this API with Power Query

Pass a column name into the function

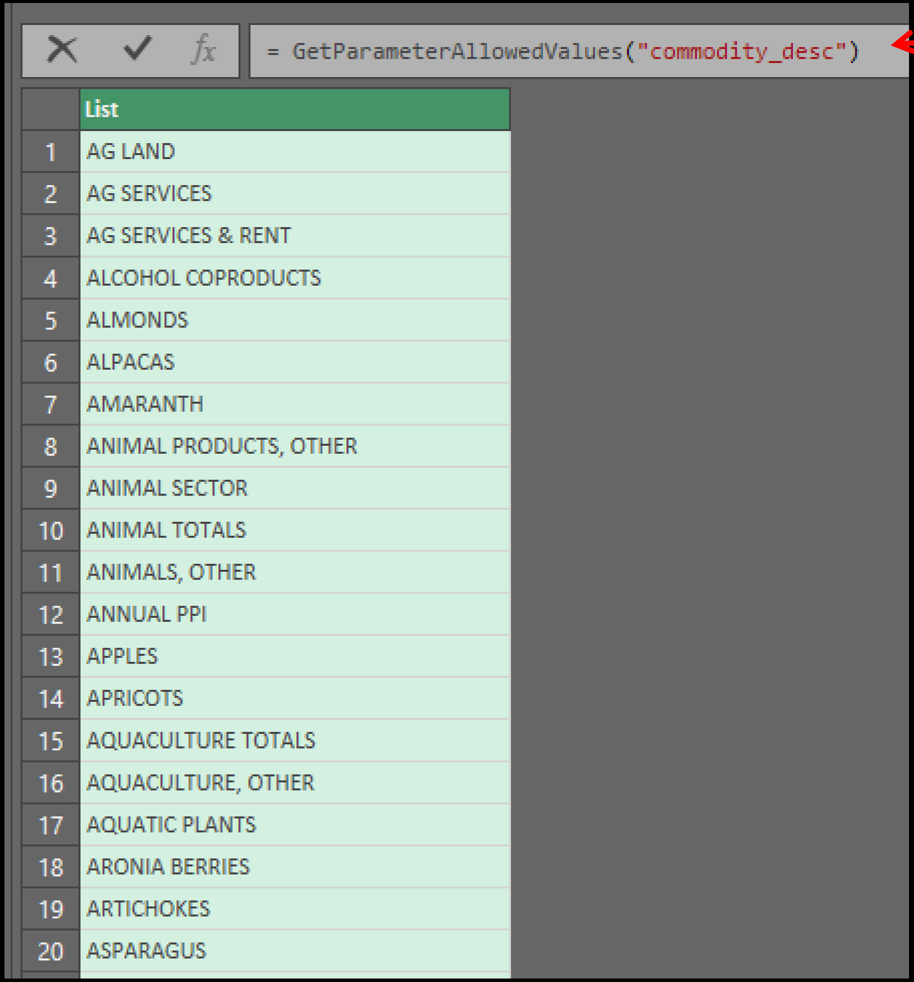
```
// GetParameterAllowedValues

(param_name as text) =>
let
    url = "http://quickstats.nass.usda.gov/api/get_param_values/?",

    query = "param=" & param_name,

    Source = Json.Document(
        Web.Contents(url & query & "&format=JSON",
                     [ApiKeyName="key"])
    ),

    Result = Record.Field(Source, param_name)
in
    Result
```

Simple string concatenation builds the URL for the API

When using an API key, we specify the key parameter name of the URL in the ApiKeyName field of the options record of Web.Contents

Json.Document returns a record with a field whose name is the column name. The field contains a list of the values returned.

Using the function couldn't be easier...



```
= GetParameterAllowedValues("commodity_desc")
```

	List
1	AG LAND
2	AG SERVICES
3	AG SERVICES & RENT
4	ALCOHOL COPRODUCTS
5	ALMONDS
6	ALPACAS
7	AMARANTH
8	ANIMAL PRODUCTS, OTHER
9	ANIMAL SECTOR
10	ANIMAL TOTALS
11	ANIMALS, OTHER
12	ANNUAL PPI
13	APPLES
14	APRICOTS
15	AQUACULTURE TOTALS
16	AQUACULTURE, OTHER
17	AQUATIC PLANTS
18	ARONIA BERRIES
19	ARTICHOKES
20	ASPARAGUS

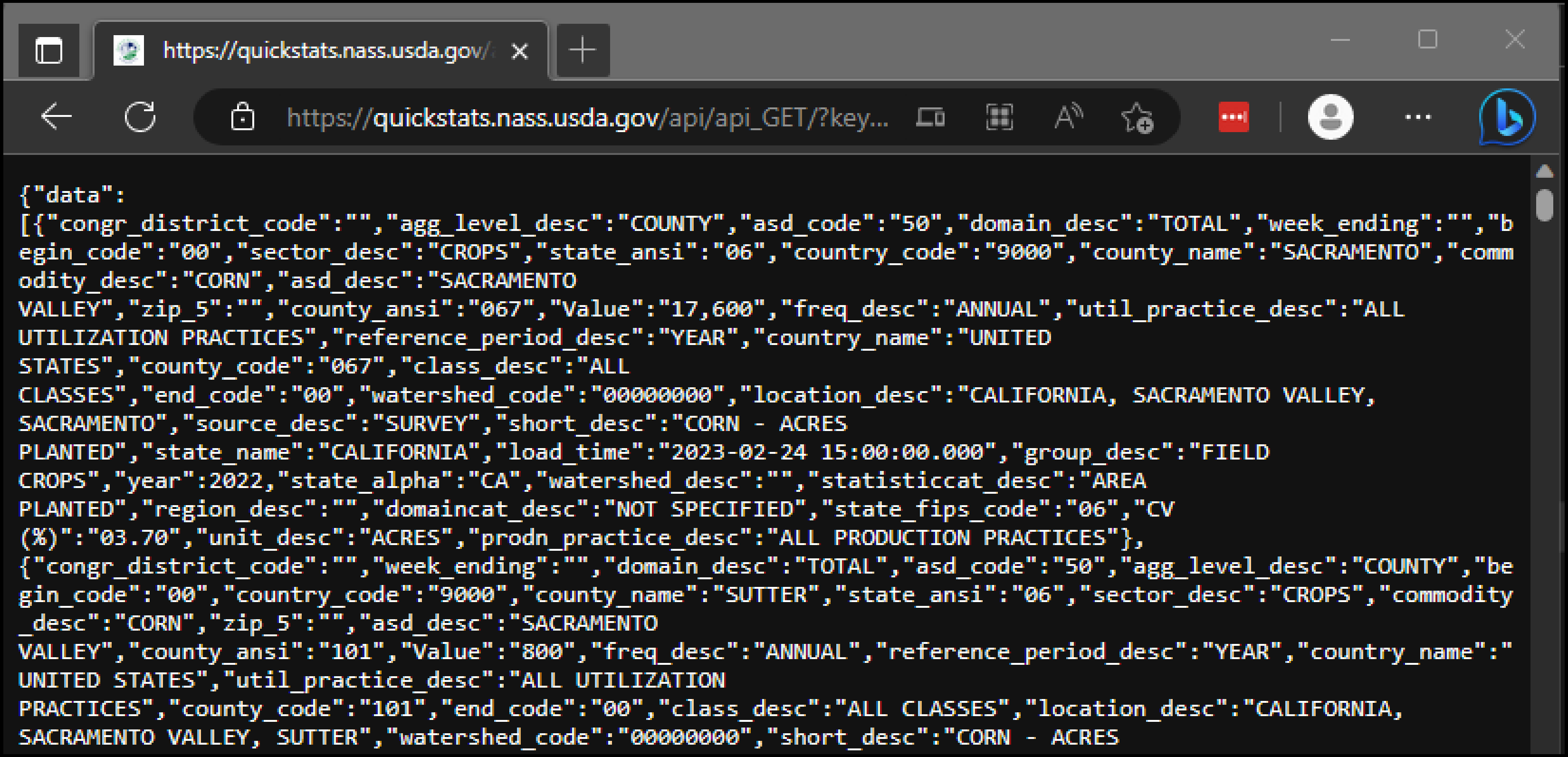
Pass a column name into the function

A list of unique values in that column is returned

Building a URL for *api_GET* can get tricky

An example to get two commodities in one year in three states:

```
https://quickstats.nass.usda.gov/api/api_GET/?key=X&co
mmodity_desc=CORN&commodity_desc=WHEAT&year=2022&state
_alpha=CA&state_alpha=OR&state_alpha=WA&format=JSON
```

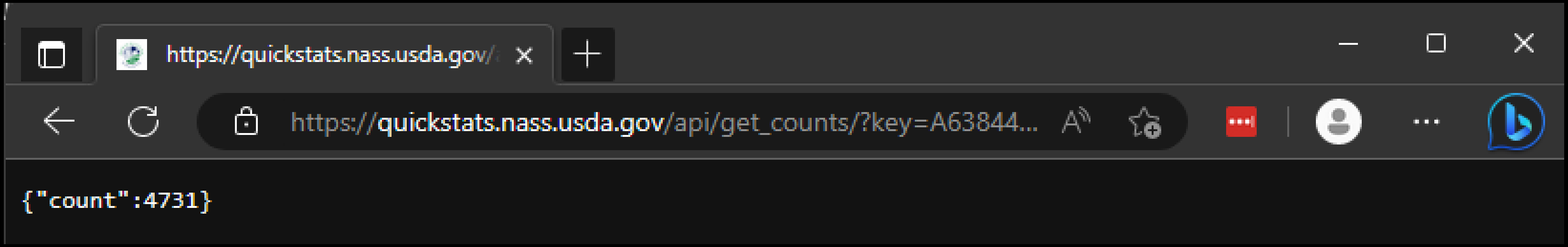


https://quickstats.nass.usda.gov/
https://quickstats.nass.usda.gov/api/api_GET/?key...

```
{"data":
[{"congr_district_code":"","agg_level_desc":"COUNTY","asd_code":"50","domain_desc":"TOTAL","week_ending":"","b
egin_code":"00","sector_desc":"CROPS","state_ansi":"06","country_code":"9000","county_name":"SACRAMENTO","comm
odity_desc":"CORN","asd_desc":"SACRAMENTO
VALLEY","zip_5":"","county_ansi":"067","Value":"17,600","freq_desc":"ANNUAL","util_practice_desc":"ALL
UTILIZATION PRACTICES","reference_period_desc":"YEAR","country_name":"UNITED
STATES","county_code":"067","class_desc":"ALL
CLASSES","end_code":"00","watershed_code":"00000000","location_desc":"CALIFORNIA, SACRAMENTO VALLEY,
SACRAMENTO","source_desc":"SURVEY","short_desc":"CORN - ACRES
PLANTED","state_name":"CALIFORNIA","load_time":"2023-02-24 15:00:00.000","group_desc":"FIELD
CROPS","year":2022,"state_alpha":"CA","watershed_desc":"","statisticcat_desc":"AREA
PLANTED","region_desc":"","domaincat_desc":"NOT SPECIFIED","state_fips_code":"06","CV
(%)":"03.70","unit_desc":"ACRES","prodn_practice_desc":"ALL PRODUCTION PRACTICES"},
{"congr_district_code":"","week_ending":"","domain_desc":"TOTAL","asd_code":"50","agg_level_desc":"COUNTY","be
gin_code":"00","country_code":"9000","county_name":"SUTTER","state_ansi":"06","sector_desc":"CROPS","commodity
_desc":"CORN","zip_5":"","asd_desc":"SACRAMENTO
VALLEY","county_ansi":"101","Value":"800","freq_desc":"ANNUAL","reference_period_desc":"YEAR","country_name":"
UNITED STATES","util_practice_desc":"ALL UTILIZATION
PRACTICES","county_code":"101","end_code":"00","class_desc":"ALL CLASSES","location_desc":"CALIFORNIA,
SACRAMENTO VALLEY, SUTTER","watershed_code":"00000000","short_desc":"CORN - ACRES
```

The same query can be used in *get_counts*

```
https://quickstats.nass.usda.gov/api/get_counts/?key=X
&commodity_desc=CORN&commodity_desc=WHEAT&year=2022&st
ate_alpha=CA&state_alpha=OR&state_alpha=WA&format=JSON
```



https://quickstats.nass.usda.gov/
https://quickstats.nass.usda.gov/api/get_counts/?key=A63844...

```
{"count":4731}
```

But there are 39 possible attributes to query!

source_desc	domain_desc	county_code	year
sector_desc	domaincat_desc	county_name	freq_desc
group_desc	agg_level_desc	region_desc	begin_code
commodity_desc	state_ansi	zip_5	end_code
class_desc	state_fips_code	watershed_code	reference_period_desc
prodn_practice_desc	state_alpha	watershed_desc	week_ending
util_practice_desc	state_name	congr_district_code	load_time
statisticcat_desc	asd_code	country_code	value
unit_desc	asd_desc	country_name	CV %
short_desc	county_ansi	location_desc	

So let's use a function to build the query!

Shows what kind of data the function expects

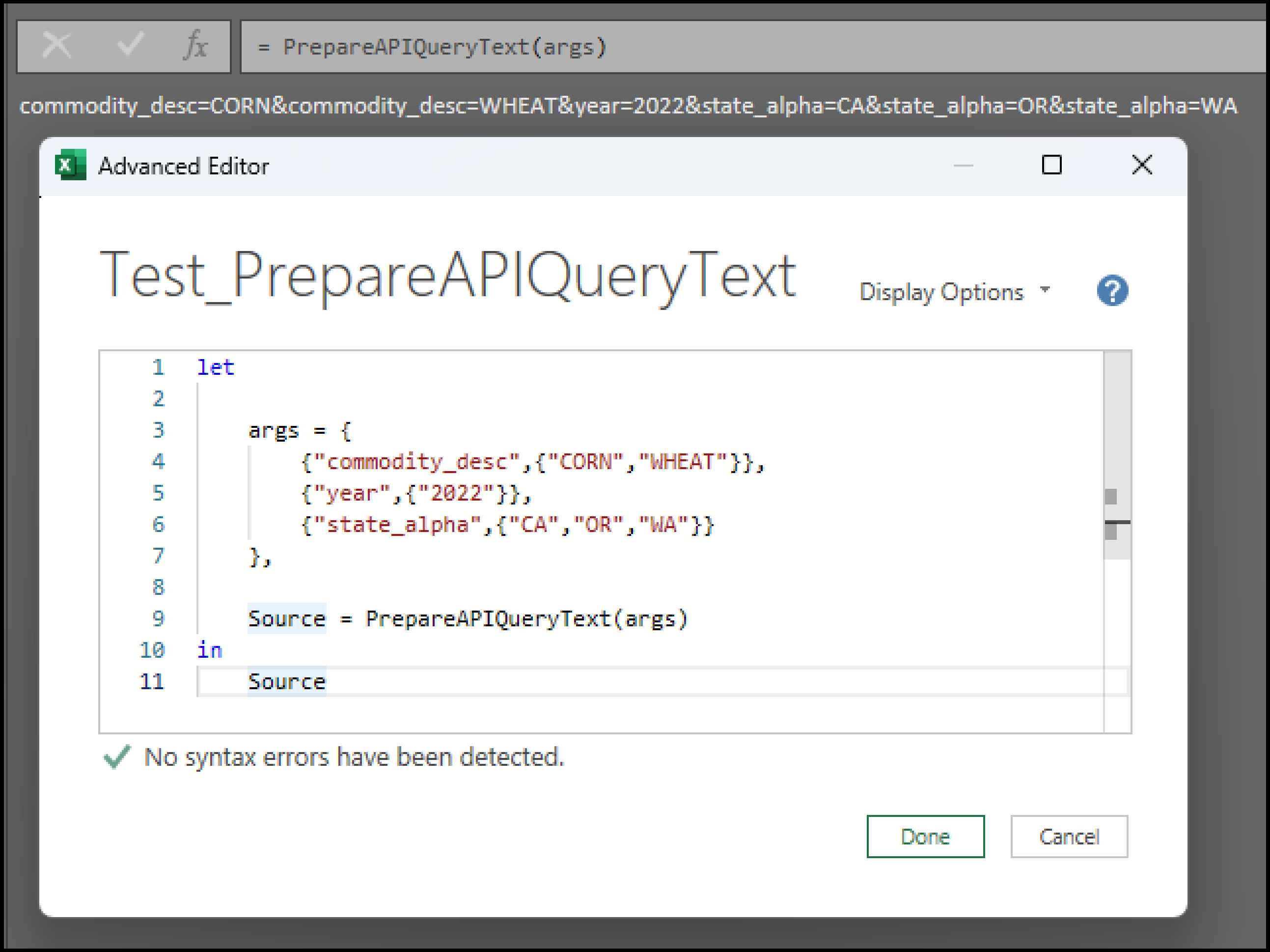
Iterates through a sub-list and builds a query string

Applies the function to the list of arguments

Returns a query string to be used with an API call

```
(args as list) as text =>
let
    /*
    args is a list of lists where each sub-list is a {param, values} pair
    args = {
        {"commodity_desc",{"CORN","WHEAT"}},
        {"year",{"2002","2012","2022"}},
        {"domain_desc",{"AREA HARVESTED"}}
    },
    */

    // takes one of the sub-lists above and converts it into a param string
    fn_build_query_string = (arg as list) =>
        Text.Combine( List.Transform(arg{1}, each "&" & arg{0} & "=" & _) ),

    // applies the function to the args
    query_params = List.Transform(
        args,
        fn_build_query_string
    ),

    // combines the args into a single query and removes the leading ampersand
    query = Text.Range( Text.Combine(query_params) , 1)
in
    query
```

The function returns the query for the API URL



```
= PrepareAPIQueryText(args)
```

commodity_desc=CORN&commodity_desc=WHEAT&year=2022&state_alpha=CA&state_alpha=OR&state_alpha=WA

Test_PrepareAPIQueryText

```
let

    args = {
        {"commodity_desc",{"CORN","WHEAT"}},
        {"year",{"2022"}},
        {"state_alpha",{"CA","OR","WA"}}
    },

    Source = PrepareAPIQueryText(args)
in
    Source
```

✓ No syntax errors have been detected.

We can now get some data!

```
(args as list) =>
let
    // builds the query text
    query = PrepareAPIQueryText(args),

    // Retrieves the data from the API
    Source = Json.Document(
        Web.Contents(
            "https://quickstats.nass.usda.gov/api/api_GET/?" & query & "&format=JSON",
            [ApiKeyName="key"]
        )
    ),

    // Convert the Json to toa table where each row is a record
    data = Table.FromList(Source[data], Splitter.SplitByNothing()),

    // Expand the records to columns
    expand = Table.ExpandRecordColumn(data, "Column1", output_columns)
in
    expand
```

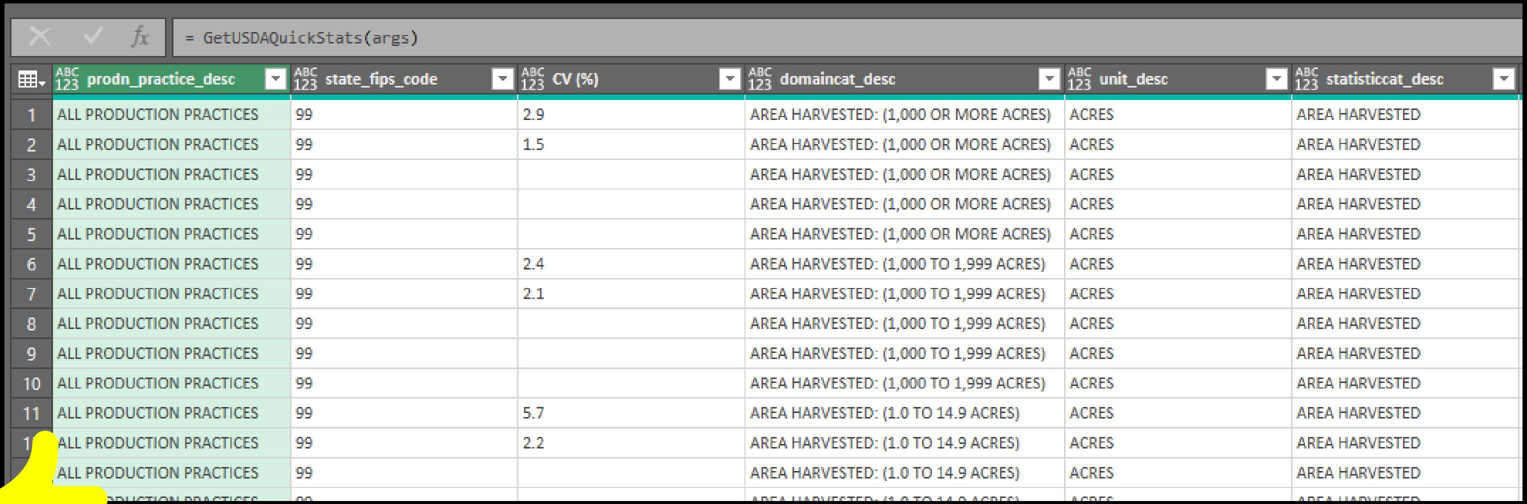
We get the query using the helper function

Place the call to the API

Convert the returned list of JSON objects to a table of records

Expands the record column to a table

We can now get some data!



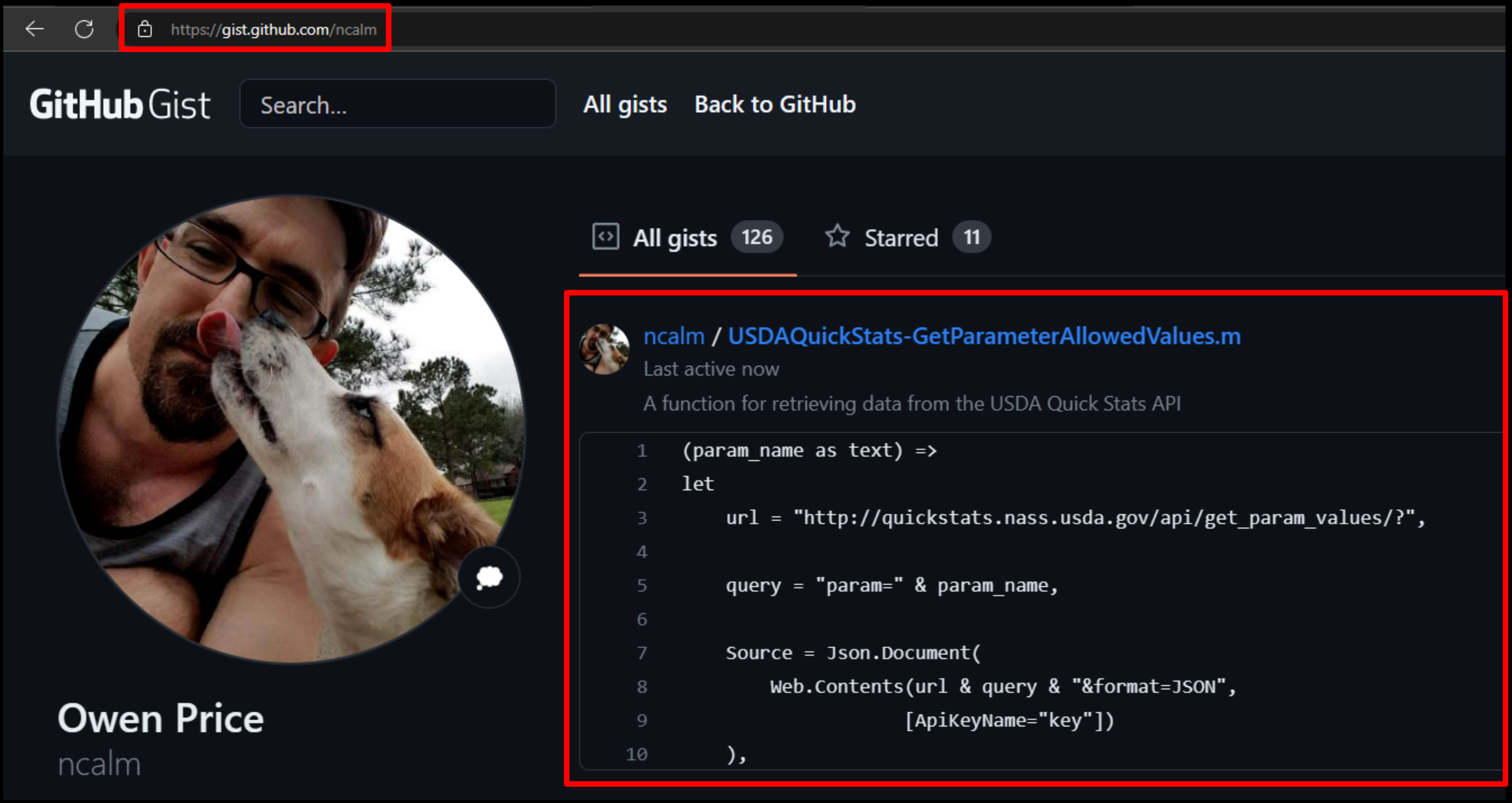
```
= GetUSDAQuickStats(args)
```

	prodn_practice_desc	state_fips_code	CV (%)	domaincat_desc	unit_desc	statisticcat_desc
1	ALL PRODUCTION PRACTICES	99	2.9	AREA HARVESTED: (1,000 OR MORE ACRES)	ACRES	AREA HARVESTED
2	ALL PRODUCTION PRACTICES	99	1.5	AREA HARVESTED: (1,000 OR MORE ACRES)	ACRES	AREA HARVESTED
3	ALL PRODUCTION PRACTICES	99		AREA HARVESTED: (1,000 OR MORE ACRES)	ACRES	AREA HARVESTED
4	ALL PRODUCTION PRACTICES	99		AREA HARVESTED: (1,000 OR MORE ACRES)	ACRES	AREA HARVESTED
5	ALL PRODUCTION PRACTICES	99		AREA HARVESTED: (1,000 OR MORE ACRES)	ACRES	AREA HARVESTED
6	ALL PRODUCTION PRACTICES	99	2.4	AREA HARVESTED: (1,000 TO 1,999 ACRES)	ACRES	AREA HARVESTED
7	ALL PRODUCTION PRACTICES	99	2.1	AREA HARVESTED: (1,000 TO 1,999 ACRES)	ACRES	AREA HARVESTED
8	ALL PRODUCTION PRACTICES	99		AREA HARVESTED: (1,000 TO 1,999 ACRES)	ACRES	AREA HARVESTED
9	ALL PRODUCTION PRACTICES	99		AREA HARVESTED: (1,000 TO 1,999 ACRES)	ACRES	AREA HARVESTED
10	ALL PRODUCTION PRACTICES	99		AREA HARVESTED: (1,000 TO 1,999 ACRES)	ACRES	AREA HARVESTED
11	ALL PRODUCTION PRACTICES	99	5.7	AREA HARVESTED: (1.0 TO 14.9 ACRES)	ACRES	AREA HARVESTED
1	ALL PRODUCTION PRACTICES	99	2.2	AREA HARVESTED: (1.0 TO 14.9 ACRES)	ACRES	AREA HARVESTED
	ALL PRODUCTION PRACTICES	99		AREA HARVESTED: (1.0 TO 14.9 ACRES)	ACRES	AREA HARVESTED

TAKEAWAYS

1. *The USDA NASS Quick Stats API is a rich source of agricultural data*
2. *When using an API, specify the key parameter name in the ApiKeyName field of the Web.Contents options record*
3. *By using helper functions to build URLs, we can simplify calls to related APIs*

Grab the code



ncalm / USDAQuickStats-GetParameterAllowedValues.m
Last active now
A function for retrieving data from the USDA Quick Stats API

```
(param_name as text) =>
let
    url = "http://quickstats.nass.usda.gov/api/get_param_values/?",

    query = "param=" & param_name,

    Source = Json.Document(
        Web.Contents(url & query & "&format=JSON",
                    [ApiKeyName="key"])
    ),
```

flexyourdata.com | youtube.com/@flexyourdata | linkedin.com/in/owenhprice

Attribution

This product uses the NASS API but is not endorsed or certified by NASS.