Cats for Adoption: Web Scraping and Analysis

OVERVIEW

A data analysis of cats available for adoption using Python web scraping, PostgreSQL, and Pandas.

WEB SCRAPING

The website used for this project was adoptapet.com and a random location in the California area. They have a good selection of different datapoints that can be extracted.

This is the main results page for each individual cat. We mostly want to grab the URL for each cat but we can also get the cats name as well.

Adopt a Pet .com™

North America's Largest Non-Profit Pet Adoption Website

Powered by the Pet Lovers at chewy petco foundati

Find a Pet | Rehome a Pet | How It Works | Adoption Info | Pet Care & Health | Get Involved | Store | Favorite Pets

We Found 461 Cats Near 93611

Email me more purr-fect pets!

Be the first to know when new pets pop up that match your search. No account needed.

Create an Alert

Distance

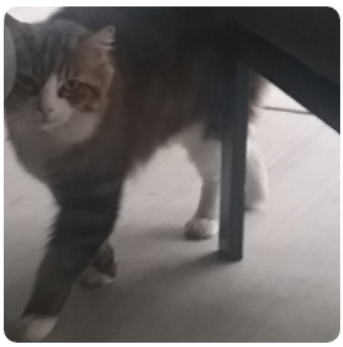
Search Radius
50 miles or less

Breed

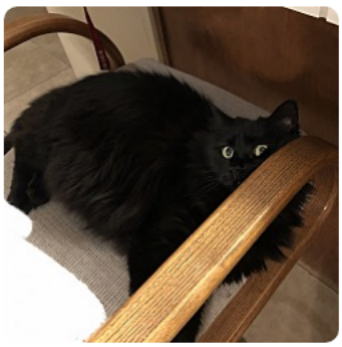
Age

Sex

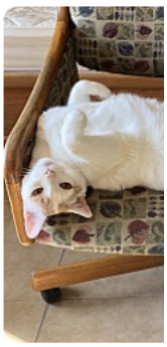
Color



Cashie
Female, Adult
Clovis, CA

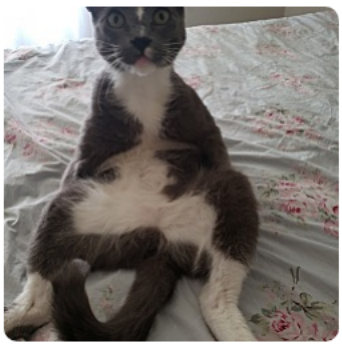


Cupcake
Female, Adult, Bonded Pair
Clovis, CA

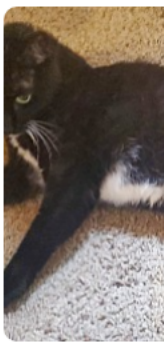


Yoda
Male, Adult, Bonc
Clovis, CA

Coconut
Female, Adult
Clovis, CA

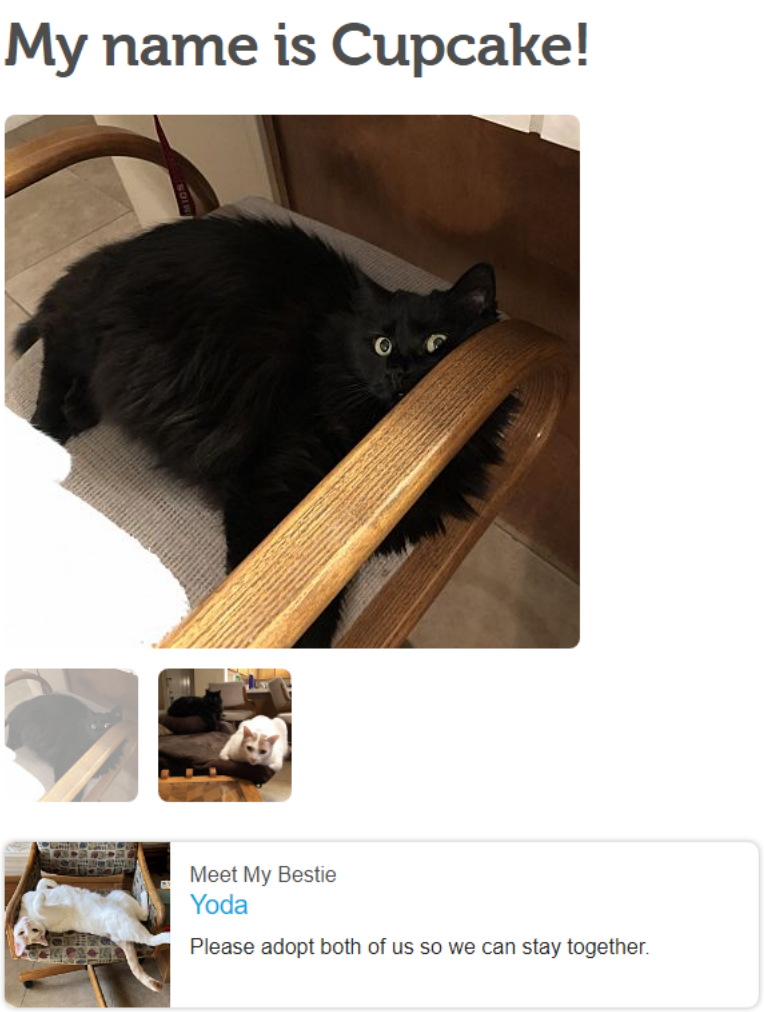


Sanford
Male, Adult
Clovis, CA



Persephone
Female, Young
Clovis, CA

Clicking on each cat will get us more details under the "Facts About Me" section. We will run a scrape on this container as well.



I opted to set up the database manually in pgAdmin with the Columns that we will be using.

scrape1.py

For our imports, Selenium will be the main driver for scraping dynamic javascript. The PostgreSQL will be handled via Psycopg2

```
import requests
from selenium import webdriver
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
```

```
import psycopg2
import time
```

First, we connect to our local database.

```
con = psycopg2.connect(
        host =  "localhost",
        database = "db_cats",
        user = "username",
        password = "password")
```

Secondly, we will run this function to find the data container with the cat name and URL using the xpath location. We can then execute the SQL to insert the scraped content into our database.

```
def scrape_page(url):
        driver = webdriver.Chrome()
        driver.get(url)

        # Wait for page to load
        try:
                element = WebDriverWait(driver, 10).until(EC.presence_of_element_located((
        except TimeoutException:
                print("Timed out")
        finally:
                print("Page Loaded")

        # SQL Cursor
        cur = con.cursor()

        # Wait for data container to load
        time.sleep(5)

        pet_card = driver.find_elements_by_class_name("search__result")

        for each in pet_card:
                name = each.find_element_by_xpath('.//div//div//*[@class="pet__name"]').te
                url = each.find_element_by_xpath('.//*[@class="pet__item__link"]').get_att

                cur.execute("insert into cats (name, url) values (%s, %s)", (name, url,))

        # commit changes
        con.commit()

        # close cursor
        cur.close()
```

I opted to take the first 10 pages only, which would give us 420 results.

```
for i in range(1,11):
        url = "https://www.adoptapet.com/pet-search?clan_id=2&geo_range=50&location=93611&
        scrape_page(url)

# close connection
con.close()
```

Here is our database with the updated cat name and URL Our next Python script will scrape the remaining details.

Data Output | Explain | Messages | Notifications

	id [PK] bigint	name character varying	breed character varying	color character varying	age character varying	sex character varying	pet_id bigint	hair character varying
1	1	Cupcake	[null]	[null]	[null]	[null]	[null]	[null]
2	2	Sanford	[null]	[null]	[null]	[null]	[null]	[null]
3	3	Persephone	[null]	[null]	[null]	[null]	[null]	[null]
4	4	Aria	[null]	[null]	[null]	[null]	[null]	[null]
5	5	Spreckles	[null]	[null]	[null]	[null]	[null]	[null]

scrape2.py

Lets select all of the current data from the database.

```
cur.execute("SELECT id, name, url FROM cats")

rows = cur.fetchall()
```

Our second function is similar to the previous script, it will find the remaining elements by xpath location and update the corresponding row in our database.

```
def scrape_db(url):
        driver = webdriver.Chrome()
        driver.get(url)

        facts = driver.find_elements_by_class_name("pet-facts__content")
        for each in facts:
                try:
                        sel_breed = each.find_element_by_xpath('.//div//div[1]//*[@class="
                        sel_color = each.find_element_by_xpath('.//div//div[2]//*[@class="
                        sel_age = each.find_element_by_xpath('.//div//div[3]//*[@class="h4_
                        sel_sex = each.find_element_by_xpath('.//div[2]//div[1]//*[@class='
                        sel_pet_id = each.find_element_by_xpath('.//div[2]//div[2]//*[@cla
                        sel_hair = each.find_element_by_xpath('.//div[2]//div[3]//*[@class=

                        cur.execute("UPDATE cats SET breed = (%s) WHERE url = (%s)", (sel_
```

```
                        cur.execute("UPDATE cats SET color = (%s) WHERE url = (%s)", (sel_
                        cur.execute("UPDATE cats SET age = (%s) WHERE url = (%s)", (sel_age
                        cur.execute("UPDATE cats SET sex = (%s) WHERE url = (%s)", (sel_sex
                        cur.execute("UPDATE cats SET pet_id = (%s) WHERE url = (%s)", (sel_
                        cur.execute("UPDATE cats SET hair = (%s) WHERE url = (%s)", (sel_ha

                except:
                        continue

        con.commit()
```

As this will take some time, I included a print function to display the current ID being scraped.

```
for r in rows:
        print("scraping ID:", r[0])
        scrape_db(r[2])
```

Our finished database! Some entries are null due to broken webpages, we'll fix this later in Jupyter Notebook using Pandas.

Data Output | Explain | Messages | Notifications

	id [PK] bigint	name character varying	breed character varying	color character varying	age character varying	sex character varying	pet_id character varying	hai cha
1	291	Grumpy	Domestic Shorthair	Orange or Red	Kitten	Female	44105449	
2	294	Bashful	Domestic Shorthair	All Black	Kitten	Female	44105495	
3	298	Raisin	American Shorthair	Brown or Chocolate	Young	Female	1284975	Sho
4	300	Romeo	American Shorthair	Black & White or Tuxedo	Kitten	Male	1285022	Sho
5	305	Marshmallow	Domestic Longhair	Brown or Chocolate (M...	Adult	Female	970559	
6	59	Francine	[null]	[null]	[null]	[null]	[null]	[nu

DATA ANALYSIS

Moving onto Jupyter Notebook, let's import our libraries.

```
import pandas as pd
import psycopg2
import sqlalchemy
import matplotlib.pyplot as plt
import numpy as np
```

And then connect to our PostgreSQL database

```
from sqlalchemy import create_engine

# Postgres username, password, and database name
POSTGRES_ADDRESS = "localhost"
```

```
POSTGRES_PORT = "5432"
POSTGRES_USERNAME = "username"
POSTGRES_PASSWORD = "password"
POSTGRES_DBNAME = "db_cats"

# A long string that contains the necessary Postgres login information
postgres_str = ("postgresql://{username}:{password}@{ipaddress}:{port}/{dbname}".format(use
                                                                                        pas
                                                                                        ipa
                                                                                        por
                                                                                        dbn


# Create the connection
cnx = create_engine(postgres_str)
```

We want to select all of our SQL data again, and add it to a Pandas dataframe.

```
df = pd.read_sql_query("""SELECT * FROM cats WHERE breed IS NOT NULL LIMIT 420;""", cnx)
```

Lets have a look at the distinct column values in our new dataframe.

```
df.nunique()
```

```
id        406
name      379
breed      23
color      24
age         7
sex         3
pet_id    406
hair        4
url       406
dtype: int64
```

This is mostly okay, but I think we can clean up the data in our Age column and bring it down to two variables instead.

```
df.age.value_counts()
```

```
Adult                   143
Young                   119
                         76
Kitten                   45
Senior                   20
1 year old, Kitten        2
10 years old, Senior      1
Name: age, dtype: int64
```

Firstly, replace all empty strings in our dataframe with NaN values. This will help with matplotlib coming up shortly.

```
df = df.replace("", np.nan)
```

Finally, replace the remainder Age strings into either "Young" or "Adult" variables

```
df.age = df.age.replace("1 year old, Kitten","Young")
df.age = df.age.replace("Kitten","Young")
df.age = df.age.replace("10 years old, Senior", "Adult")
df.age = df.age.replace("Senior", "Adult")
```

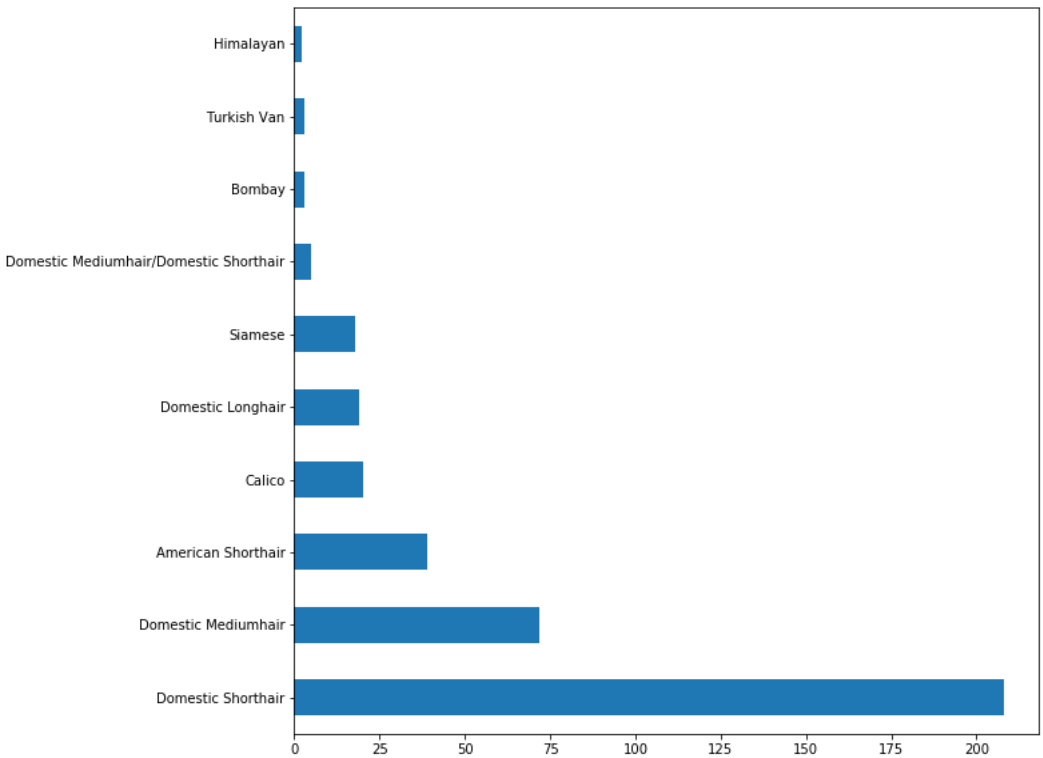
Looking much better now.

```
df.age.value_counts()
```

```
Young    166
Adult    164
Name: age, dtype: int64
```

So what kind of cat breeds were we able to scrape? Lets plot the top 10 distinct breeds.

```
fig1 = df.breed.value_counts()[:10].plot(kind="barh", figsize=(10,10))
```



The results were dominated by the Domestic cat, of various hair size. Lets make a new dataframe to check breeds by gender too.

```
df2 = df.groupby(["breed", "sex"])["breed"].count().unstack("sex").fillna(0)
```

We will also need to append a "Total" column to the new dataframe so we can sort and filter it.

```
df2["Total"] = df2.sum(axis=1)
```

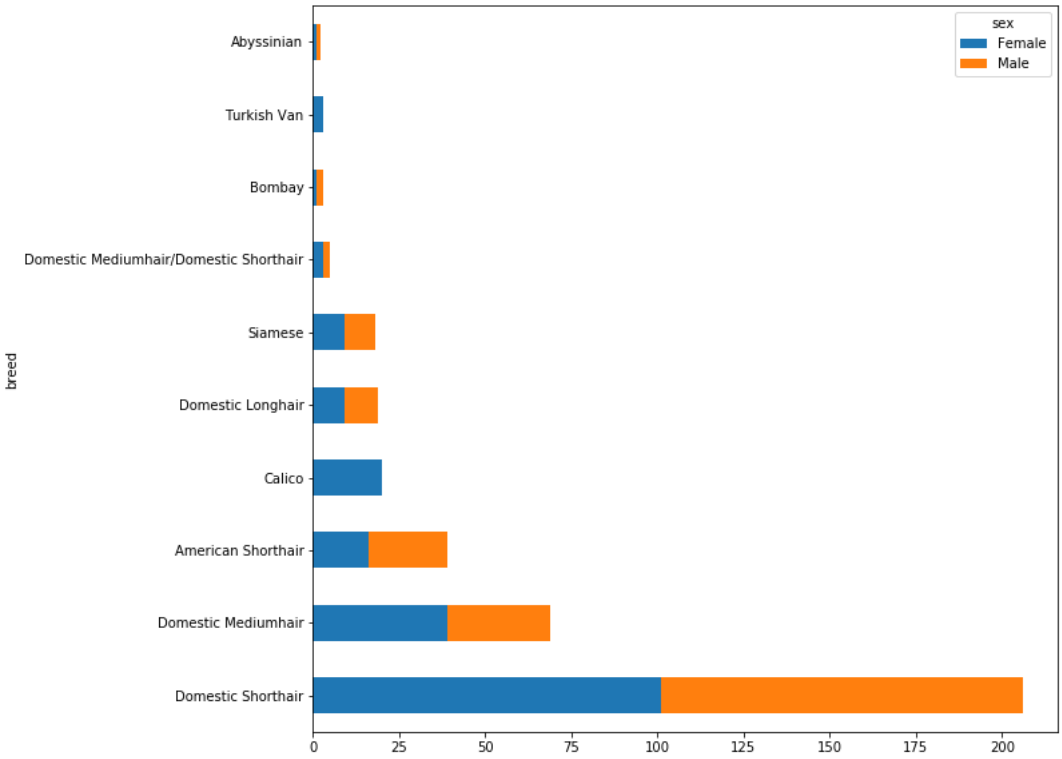
sex breed	Female	Male	Total
Abyssinian	1.0	1.0	2.0
American Bobtail	1.0	0.0	1.0
American Shorthair	16.0	23.0	39.0
American Wirehair	1.0	1.0	2.0
Bombay	1.0	2.0	3.0
British Shorthair	2.0	0.0	2.0
Calico	20.0	0.0	20.0
Domestic Longhair	9.0	10.0	19.0
Domestic Mediumhair	39.0	30.0	69.0
Domestic Mediumhair/Domestic Shorthair	3.0	2.0	5.0
Domestic Shorthair	101.0	105.0	206.0
Egyptian Mau	0.0	1.0	1.0
Himalayan	2.0	0.0	2.0
Maine Coon	1.0	0.0	1.0
Norwegian Forest Cat	1.0	0.0	1.0
Polydactyl/Hemingway	1.0	0.0	1.0
Ragdoll	1.0	1.0	2.0
Russian Blue	0.0	1.0	1.0
Scottish Fold	1.0	0.0	1.0
Siamese	9.0	9.0	18.0
Siamese/Domestic Shorthair	0.0	1.0	1.0
Turkish Angora	1.0	0.0	1.0
Turkish Van	3.0	0.0	3.0

Lets keep the top 10 breeds by the combined Male and Female values.

```
df2 = df2.nlargest(10, "Total")
```

We can also display this as a stached chart for better visual representation.

```
fig2 = df2[["Female","Male"]].plot(kind="barh", stacked="True", figsize=(10,10))
```



We have a nice stacked chart now, but why are there no male Calico cats available for adoption?

Cool, we learned a new cat fact today.

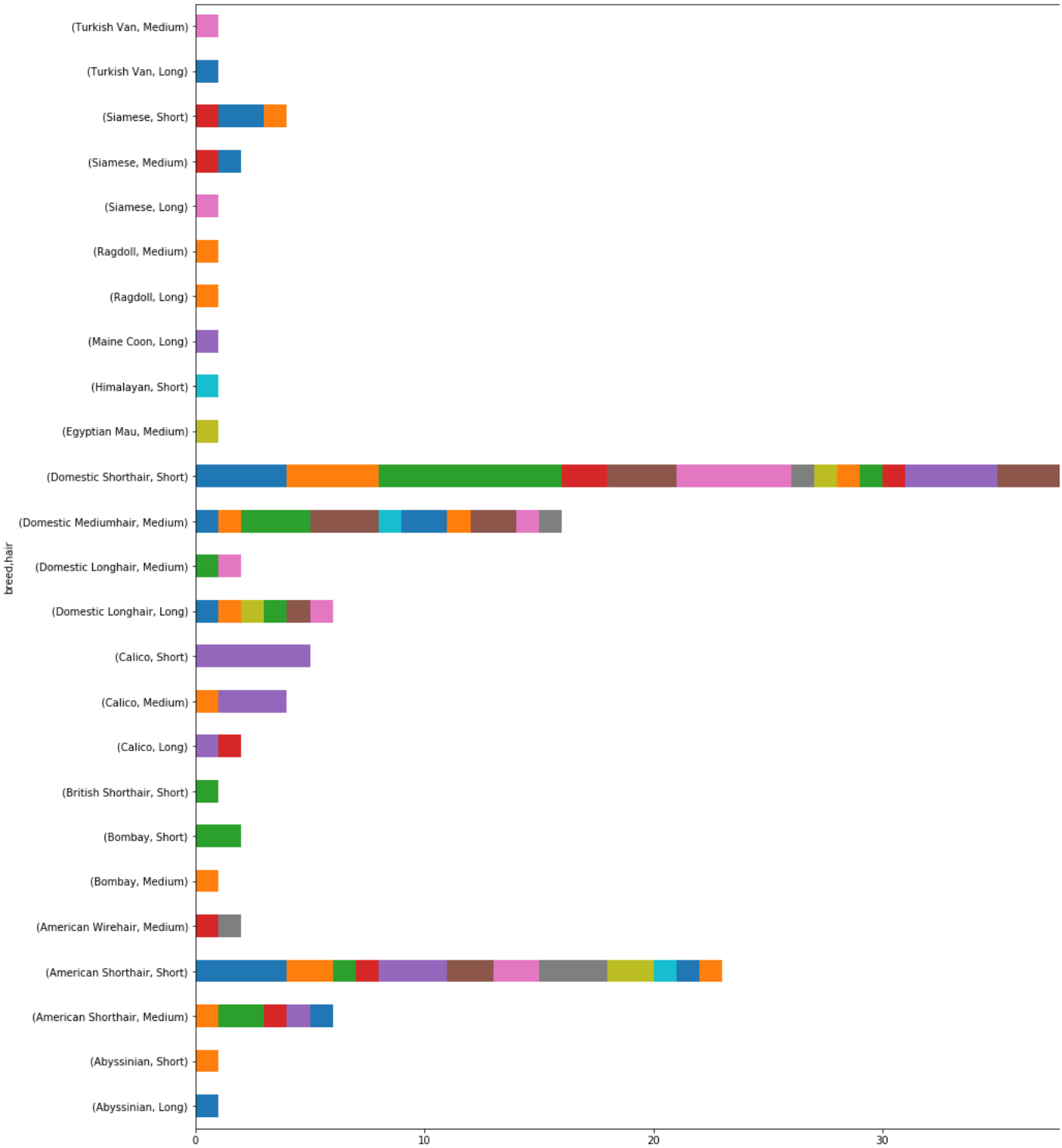
Instead of looking at gender, we can instead try the same breed analysis with color and hair type.

```
df3 = df.groupby(['breed', 'color', "hair"])['breed'].count().unstack('color').fillna(0)
```

	color	Black & White or Tuxedo	Black (Mostly)	All Black	Brown or Chocolate	Calico or Dilute Calico	Gray or Blue	Gray or Blue (Mostly)	Gray, Blue or Silver Tabby	Orange or Red Tabby	Spotted Tabby/Leopard Spotted
breed	hair										
Abyssinian	Long	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Short	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
American Shorthair	Medium	0.0	1.0	2.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
	Short	4.0	2.0	1.0	1.0	3.0	2.0	2.0	3.0	2.0	1.0
American Wirehair	Medium	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
Bombay	Medium	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Short	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
British Shorthair	Short	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Calico	Long	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
	Medium	0.0	1.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0
	Short	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0
Domestic Longhair	Long	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
	Medium	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Domestic Mediumhair	Medium	1.0	1.0	3.0	0.0	0.0	3.0	0.0	0.0	0.0	1.0
Domestic Shorthair	Short	4.0	4.0	8.0	2.0	0.0	3.0	5.0	1.0	1.0	0.0
Egyptian Mau	Medium	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
Himalayan	Short	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Maine Coon	Long	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Ragdoll	Long	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Medium	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Siamese	Long	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
	Medium	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
	Short	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
Turkish Van	Long	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Medium	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

We can plot this to a stacked chart as well.

```
fig2 = df3.plot(kind="barh", stacked="True", figsize=(20,20))
```



You can download the full code on my GitHub page, and also check out the Jupyter Notebook.