SYRACUSE UNIVERSITY

MEMO

To: Deborah V. Landowski, PhD

From: Dan Tully

Date: 12/11/22

Re: IST 652 Final Project

Introduction

This assignment is to acquire, access, and transform data from structured and semi-structured data sources. I will use pandas, numpy, and pandasql for the data wrangling. Finally, I will use matplotlib and seaborn to help visualize the data. Throughout this report, I will look at the data dictionary to help explain the data, the data types, and will also identify any areas of the data that might require cleaning. Once the data is normalized, I will begin to explore some questions regarding the Pokémon collections. I will conclude with some observations about the Pokémon collections in this dataset.

About the Data

This dataset pokemon was obtained from: https://www.kaggle.com/datasets/terminus7/pokemon-challenge/versions/1. I also found some data on https://pokemondb.net/tools/text-list. This site provides Pokémon attributes from various generation collections. I downloaded the generations one at a time so I have six files to input (generations 1 through 6). After I added a new column identifying the generation, I appended the six files to one (pkmn) dataset. I did this so that I would be able to do analysis by generation as well as the other attributes.

```
Read in two datasets:
(1) pkmn (pokemondb) dataset
(2) pokemon (kaggle) dataset:

First two records of the pkmn_pokemondb dataset:
```

	Number	Name	Form	Type 1	Type 2	HP	Attack	Defense	Sp.Attack	Sp.Defense	Speed
0	1	Bulbasaur	NaN	Grass	Poison	45	49	49	65	65	45
1	2	Ivysaur	NaN	Grass	Poison	60	62	63	80	80	60

```
First two records of the pokemon_kaggle dataset:
```

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False

Pokémon Column Data Definition

The data in both data sets are described as:

- **Number (#)**: ID for each Pokémon
- **Name**: Name of each Pokémon
- **Type 1**: Each Pokémon has a type, this determines weakness/resistance to attacks
- **Type 2**: Some Pokémon are dual type
- **HP**: hit points, or health, defines how much damage a Pokémon can withstand before fainting
- **Attack**: the base modifier for normal attacks
- **Defense**: the base damage resistance against normal attacks
- **Sp.Attack (SP. Atk)**: special attack, the base modifier for special attacks
- **Sp.Defense (SP. Def)**: the base damage resistance against special attacks
- **Speed**: determines which Pokémon attacks first each round
- **gen (Generation)**: determines which generation the Pokémon belongs

Exploration and Cleaning

I am going to explore the data types and see if there is any data cleaning necessary. We are going to review the data types, drop any columns that we do not need for our analysis, and check the columns for NULL (blanks). I did find and use some code from https://www.kaggle.com/code/mmetter/pokemon-data-analysis-tutorial/ to help with the data cleaning and analysis.

```
Examine the column headings between the two data sources
columns in the pkmn_pokemondb data set Index(['Number', 'Name', 'Form', 'Type 1', '
Type 2', 'HP', 'Attack', 'Defense',
       'Sp.Attack', 'Sp.Defense', 'Speed', 'gen'],
      dtype='object')
columns in the pokemon_kaggle data set Index(['#', 'Name', 'Type 1', 'Type 2', 'HP
', 'Attack', 'Defense', 'Sp. Atk',
       'Sp. Def', 'Speed', 'Generation', 'Legendary'],
      dtype='object')

This is the (rows, columns) of the pkmn dataset: (1845, 12)
This is the (rows, columns) of the pokemon dataset: (800, 12)
Pivoting these data sets side-by-side we clearly see the quantities do not match.
```

pokemon_kaggle

	Name
Generation	
1	165
2	106
3	160
4	121
5	165
6	82

pkmn_pokemondb

	Name
gen	
1	1190
2	100
3	141
4	118
5	165
6	131

```
I dropped Legendary from the pokemon dataset and Form from the pkmn dataset.
I will not need them for this analysis.
```

```
Updated the pokemon column names to match the pkmn dataset to make cross evaluation
easier.
Updated some data types from both datasets, again so they match, which makes cross
evaluation easier.
Data types for pokemon:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Number      800 non-null    object
 1   Name        799 non-null    object
 2   Type1       800 non-null    object
 3   Type2       414 non-null    object
 4   HP          800 non-null    int64 
 5   Attack      800 non-null    int64 
 6   Defense     800 non-null    int64 
 7   Sp.Attack   800 non-null    int64 
 8   Sp.Defense  800 non-null    int64 
 9   Speed       800 non-null    int64 
 10  gen         800 non-null    object
dtypes: int64(6), object(5)
memory usage: 68.9+ KB


Data types for pkmn:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1845 entries, 0 to 130
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Number      1845 non-null   object
 1   Name        1845 non-null   object
 2   Type1       1845 non-null   object
 3   Type2       995 non-null    object
 4   HP          1845 non-null   int64 
 5   Attack      1845 non-null   int64 
 6   Defense     1845 non-null   int64 
 7   Sp.Attack   1845 non-null   int64 
 8   Sp.Defense  1845 non-null   int64 
 9   Speed       1845 non-null   int64 
 10  gen         1845 non-null   object
dtypes: int64(6), object(5)
memory usage: 173.0+ KB

Looking at the data sets side by side, showing which field have nulls and how many
nulls are in that column.
```

pokemon_kaggle

	0
Number	0
Name	1
Type1	0
Type2	386
HP	0
Attack	0
Defense	0
Sp.Attack	0
Sp.Defense	0
Speed	0
gen	0

pkmn_pokemondb

	0
Number	0
Name	0
Type1	0
Type2	850
HP	0
Attack	0
Defense	0
Sp.Attack	0
Sp.Defense	0
Speed	0
gen	0

```
The pokemon_kaggle set is missing a Name and some Type 2 attribute items.
I am going to use the pkmn file for most of my analysis since it appears to have co
mplete data.
Even though I will not be using that pokemon data set, I will still find the missin
g Pokémon name.

This is the Pokémon from the pokemon dataset that is missing a name, #63, from Gene
ration 1
```

Out[8]:

	Number	Name	Type1	Type2	HP	Attack	Defense	Sp.Attack	Sp.Defense	Speed	gen
62	63	NaN	Fighting	NaN	65	105	60	60	70	95	1

```
I discovered that the numbers assigned from each of the datasets are different so I
searched pkmn in order to find the missing name.  To find the correct one I had to
look at the one that came before and the one that came after in the raw data.
```

Out[9]:

	Number	Name	Type1	Type2	HP	Attack	Defense	Sp.Attack	Sp.Defense	Speed	gen
75	57	Primeape	Fighting	NaN	65	105	60	60	70	95	1

```
Assigned the correct Pokémon name from pokemon dataset that was missing.
```

Out[10]:

	Number	Name	Type1	Type2	HP	Attack	Defense	Sp.Attack	Sp.Defense	Speed	gen
62	63	Primeape	Fighting	NaN	65	105	60	60	70	95	1

```
Below are the descriptive statistics for the remaining columns in the pkmn dataset:
```

Out[11]:

	HP	Attack	Defense	Sp.Attack	Sp.Defense	Speed
count	1845.000000	1845.000000	1845.000000	1845.000000	1845.000000	1845.000000
mean	70.729539	80.740379	74.777778	73.230894	72.459621	69.144173
std	26.091405	32.611636	31.090614	32.931465	27.886218	29.973250
min	1.000000	5.000000	5.000000	10.000000	20.000000	5.000000
25%	52.000000	55.000000	50.000000	50.000000	50.000000	45.000000
50%	70.000000	78.000000	70.000000	65.000000	70.000000	65.000000
75%	85.000000	100.000000	90.000000	95.000000	90.000000	90.000000
max	255.000000	190.000000	250.000000	194.000000	250.000000	200.000000

```
Using Pandas pivot table with counts I was able to find which generations have whic
h type 1 & 2 attributes in the pkmn dataset.
```

Type 1

gen	1	2	3	4	5	6
Type1						
Bug	91	10	12	12	18	7
Dark	56	5	4	3	13	6
Dragon	47	0	7	3	9	10
Electric	73	6	4	12	8	5
Fairy	31	5	0	1	0	9
Fighting	49	2	4	2	7	5
Fire	75	8	7	5	9	12
Flying	10	0	0	0	2	2
Ghost	47	1	4	7	5	12
Grass	104	9	12	14	15	8
Ground	46	3	6	4	10	1
Ice	43	4	7	3	6	3
Normal	131	15	18	17	18	8
Poison	46	1	3	6	2	2
Psychic	82	7	11	7	14	10
Rock	67	4	8	6	6	11
Steel	43	2	9	3	4	9
Water	149	18	25	13	19	11

Type 2

gen	1	2	3	4	5	6
Type2						
Bug	9	0	2	1	0	0
Dark	33	1	6	4	3	6
Dragon	38	1	2	4	3	8
Electric	13	2	0	0	4	0
Fairy	42	3	5	1	2	9
Fighting	42	1	3	5	10	6
Fire	20	2	0	1	7	2
Flying	122	19	12	16	19	12
Ghost	37	0	2	2	4	6
Grass	33	1	5	2	5	10
Ground	43	7	7	7	2	6
Ice	22	1	0	4	3	3
Normal	18	0	0	0	0	4
Poison	46	3	2	2	5	3
Psychic	48	3	12	2	2	8
Rock	19	3	4	1	4	0
Steel	40	2	0	8	8	2
Water	23	0	4	2	0	4

```
Using merge with the indicator turned on we can see only about 40% of the Pokémon d
ata overlaps between each dataset on Name and Generation.  This is why I will only
use the pkmn_pokemondb dataset for most of this analysis.
```

Out[13]:
```
_merge
left_only       95
right_only    1073
both           772
Name: Name, dtype: int64
```

Questions

This section identifies the three questions that I seek to answer using the pkmn dataset and twitter data. The first question, I summarizes type information as well as show the correlation of the attributes that make up the total to see which have the greatest impact. In the second question, I will explore the data by generations. In the third question, I will explore twitter text message to obtain a sentiment and determine what is trending. By exploring these questions, I hope to identify a greater understanding of the Pokémon.

Question 1

Which type has the best and worst total scoring while exploring the numeric attributes that have the greatest impact on that total? Using the pkmn dataset, I will explore some descriptive statistics and correlation.

```
Adding a total score to the dataset by adding up the HP through Speed attributes.
(total = HP + Attack + Defense + Sp.Attack + Sp.Defense + Speed)

Display the first five records showing the new total attribute column.
```

Out[15]:

	Number	Name	Type1	Type2	HP	Attack	Defense	Sp.Attack	Sp.Defense	Speed	gen
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1
3	3	Venusaur	Grass	Poison	80	100	123	122	120	80	1
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1

```
In these results we can see the Psychic type shows up the most.
Below are the best Avg.total by the type of Pokémon:
```

Out[16]:

	type	Avg.HP	Avg.Attack	Avg.Defense	Avg.Speed	Avg.total
0	Ground~Fire	100.0	180.0	160.0	90.0	770.0
1	Psychic~Dragon	97.0	167.0	97.0	129.0	754.0
2	Psychic~Steel	117.0	147.0	117.0	87.0	680.0
3	Psychic~Ice	100.0	165.0	150.0	50.0	680.0
4	Psychic~Dark	80.0	160.0	60.0	80.0	680.0

```
In these results we can see the Bug type shows up the most.
Below are the worst Avg.total by the type of Pokémon:
```

Out[17]:

	type	Avg.HP	Avg.Attack	Avg.Defense	Avg.Speed	Avg.total
0	Bug~Ghost	1.0	90.0	45.0	40.0	236.0
1	Normal~Fairy	92.6	41.9	53.4	29.4	324.4
2	Bug~Water	45.0	55.0	61.0	62.5	324.5
3	Ice~Bug	50.0	45.0	47.5	42.5	330.0
4	Poison~Bug	40.0	50.0	90.0	65.0	330.0

```
Correlation matrix
Below is showing which attributes Avg.HP and Avg.Attack have the greatest relations
hip with Avg.total.
```

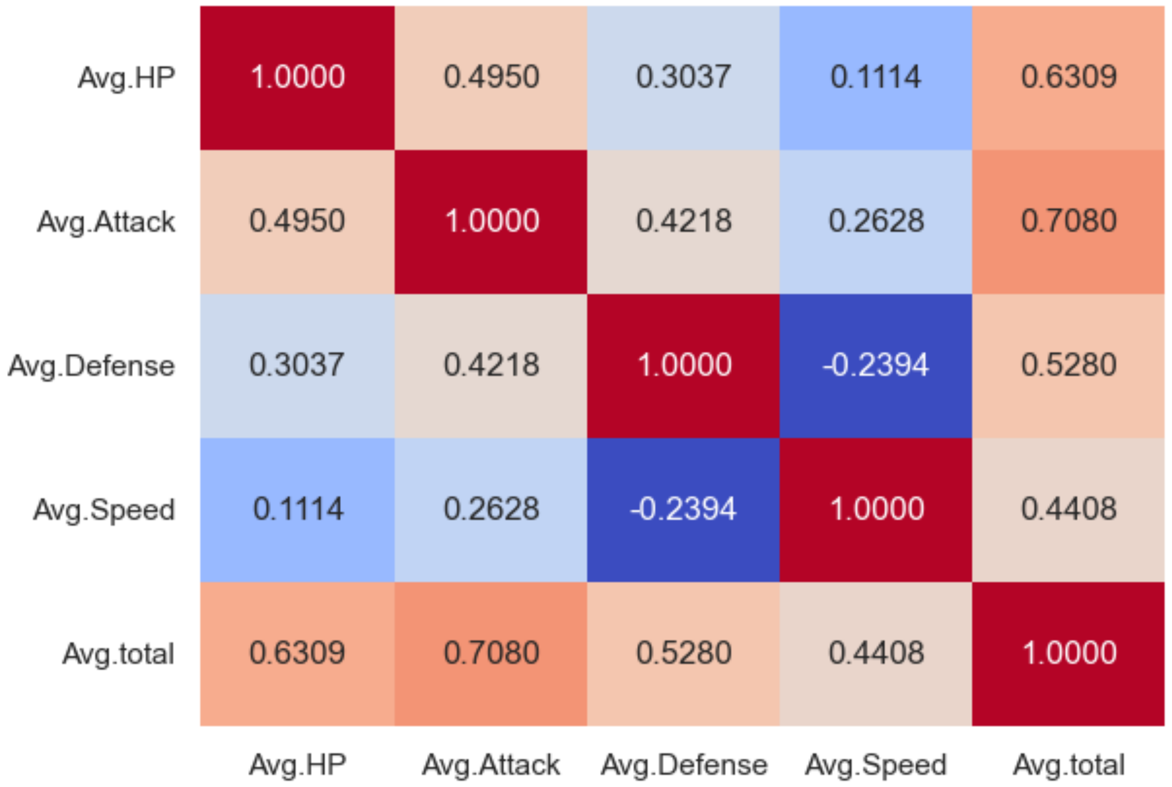
Out[18]:

	Avg.HP	Avg.Attack	Avg.Defense	Avg.Speed	Avg.total
Avg.HP	1.000000	0.495031	0.303739	0.111414	0.630868
Avg.Attack	0.495031	1.000000	0.421847	0.262814	0.707975
Avg.Defense	0.303739	0.421847	1.000000	-0.239432	0.527983
Avg.Speed	0.111414	0.262814	-0.239432	1.000000	0.440783
Avg.total	0.630868	0.707975	0.527983	0.440783	1.000000

```
Correlation matrix heatmap
Below is showing which attributes Avg.HP and Avg.Attack have the greatest relations
hip with Avg.total.
```

Out[19]:

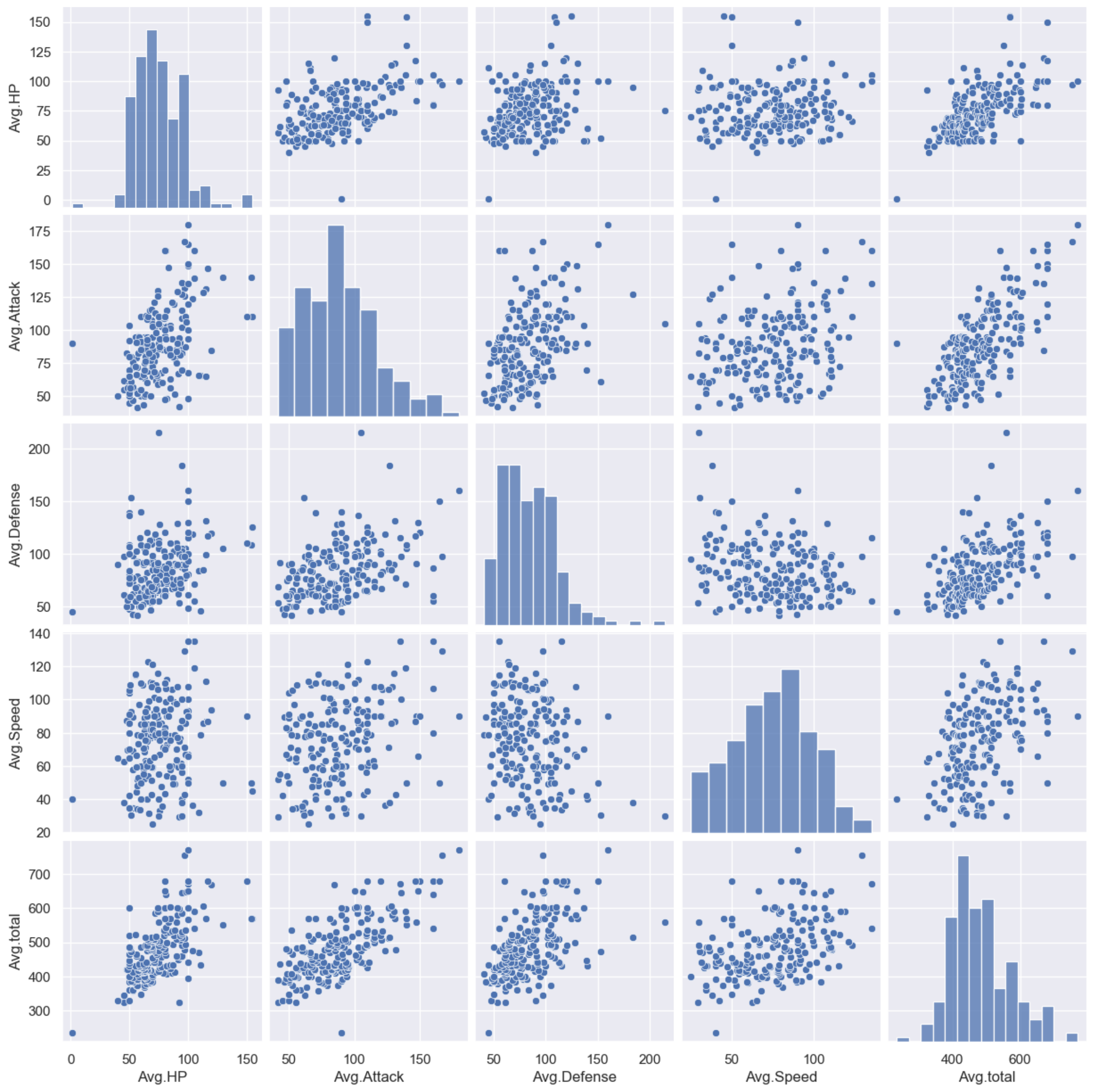
```
[Text(0, 0.5, 'Avg.HP'),
 Text(0, 1.5, 'Avg.Attack'),
 Text(0, 2.5, 'Avg.Defense'),
 Text(0, 3.5, 'Avg.Speed'),
 Text(0, 4.5, 'Avg.total')]
```



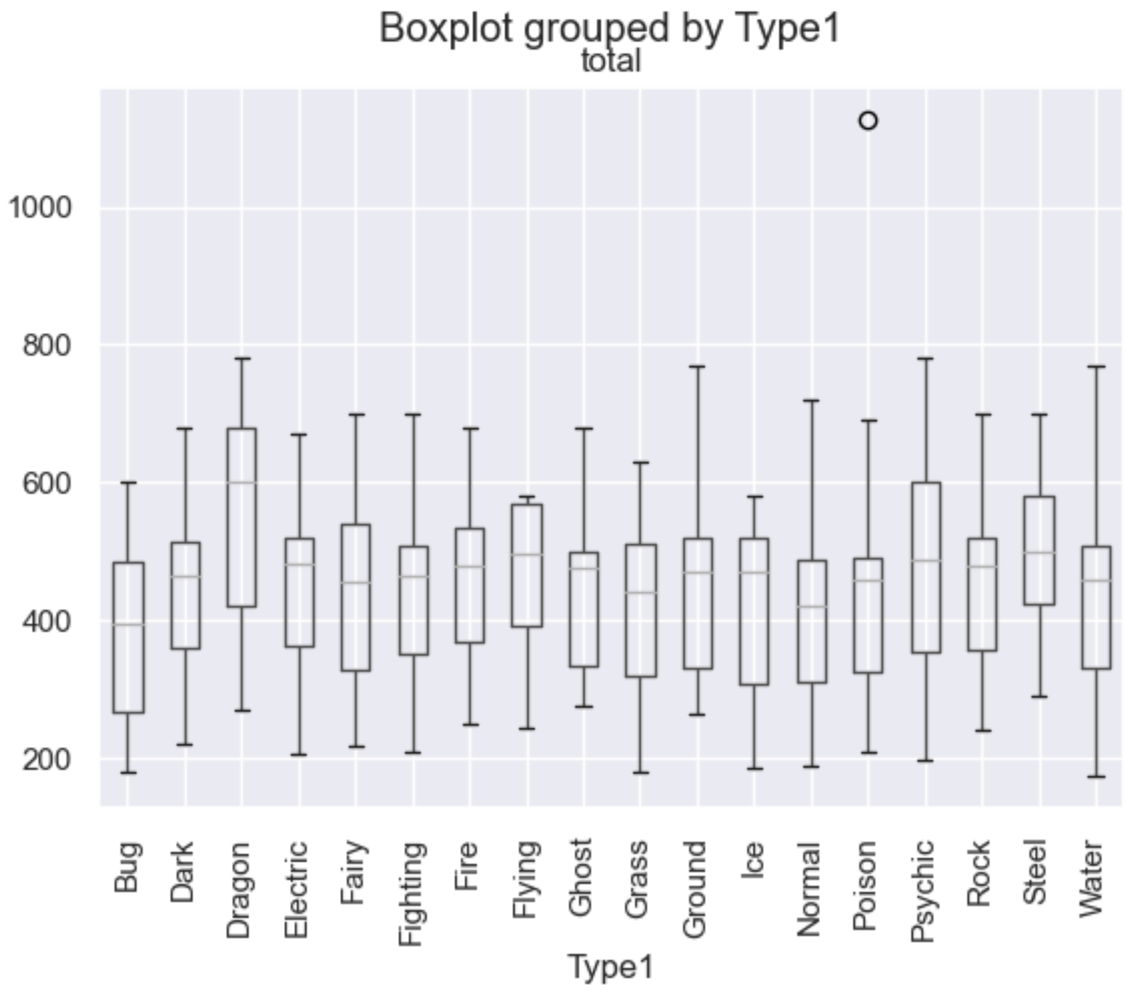
	Avg.HP	Avg.Attack	Avg.Defense	Avg.Speed	Avg.total
Avg.HP	1.0000	0.4950	0.3037	0.1114	0.6309
Avg.Attack	0.4950	1.0000	0.4218	0.2628	0.7080
Avg.Defense	0.3037	0.4218	1.0000	-0.2394	0.5280
Avg.Speed	0.1114	0.2628	-0.2394	1.0000	0.4408
Avg.total	0.6309	0.7080	0.5280	0.4408	1.0000

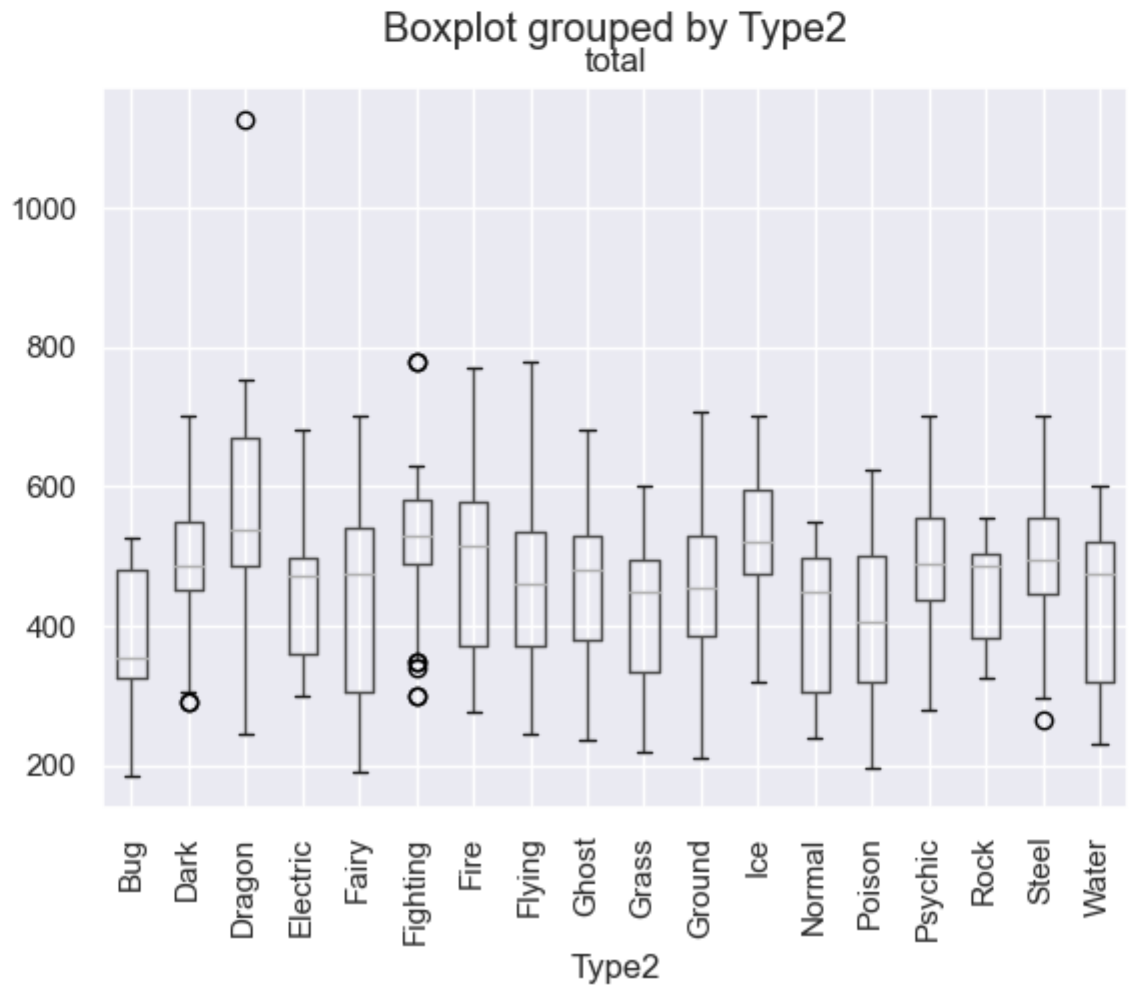
```
Pokémon Feature Correlation Plot
Below is showing which attributes Avg.HP and Avg.Attack have the greatest relations
hip with Avg.total.
```

```
Out[20]: <seaborn.axisgrid.PairGrid at 0x1e8e33be088>
```



Pokémon Boxplot Grouped by Type
Below is showing overall that bug type mean is the worst but Dragon (not Psychic) has the best total mean.


Boxplot grouped by Type1
total
1000
800
600
400
200
Bug
Dark
Dragon
Electric
Fairy
Fighting
Fire
Flying
Ghost
Grass
Ground
Ice
Normal
Poison
Psychic
Rock
Steel
Water
Type1



Q1 Description of the Program

I utilized python pandas, pandasql, ipython.display, and seaborn packages for the data wrangling and to help visualize the data. The python pandas programming is efficient for manipulating and analyzing data. First I summed the attributes to get a total column added. Then using SQL I was able to combine Types and average the base stat attributes. Created a correlation matrix to show the strength of any statistical relationships between attributes. Also used a seaborn heatmap to further emphasis the correlation matrix. Used seaborn to scatter plot those relationships. Concluded, using box plots to visualize the mean and the quartiles of the total for each type of Pokémon. These python tools demonstrates the flexibility in comparing, reshaping, and pivoting of a structured datasets.

Q1 Description of the Output and Analysis

In this analysis we concluded that the attributes Avg.HP and Avg.Attack have the greatest relationship with Avg.total. We also noted that overall the bug type mean has the worst average total points, but Dragon (not Psychic) has the best mean. The best type being Dragon (on average) over Psychic was surprising since Psychic type showed up the most at the top of the list. Just goes to show you, you can't judge a book by its cover.

Question 2

Do Pokemon have better scoring over time? Explore the data by generation.

```
Sorted the data set descending by total and as we can see there are Generations 1,
6, and 4 identified at the top of the list.
NOTE: The top one appeared to be another potential outlier (total_stat == 1125), bu
t I looked it up on a Pokémon site and it appears to be correct. Site: https://poke
mondb.net/pokedex/eternatus
```

Out[22]:

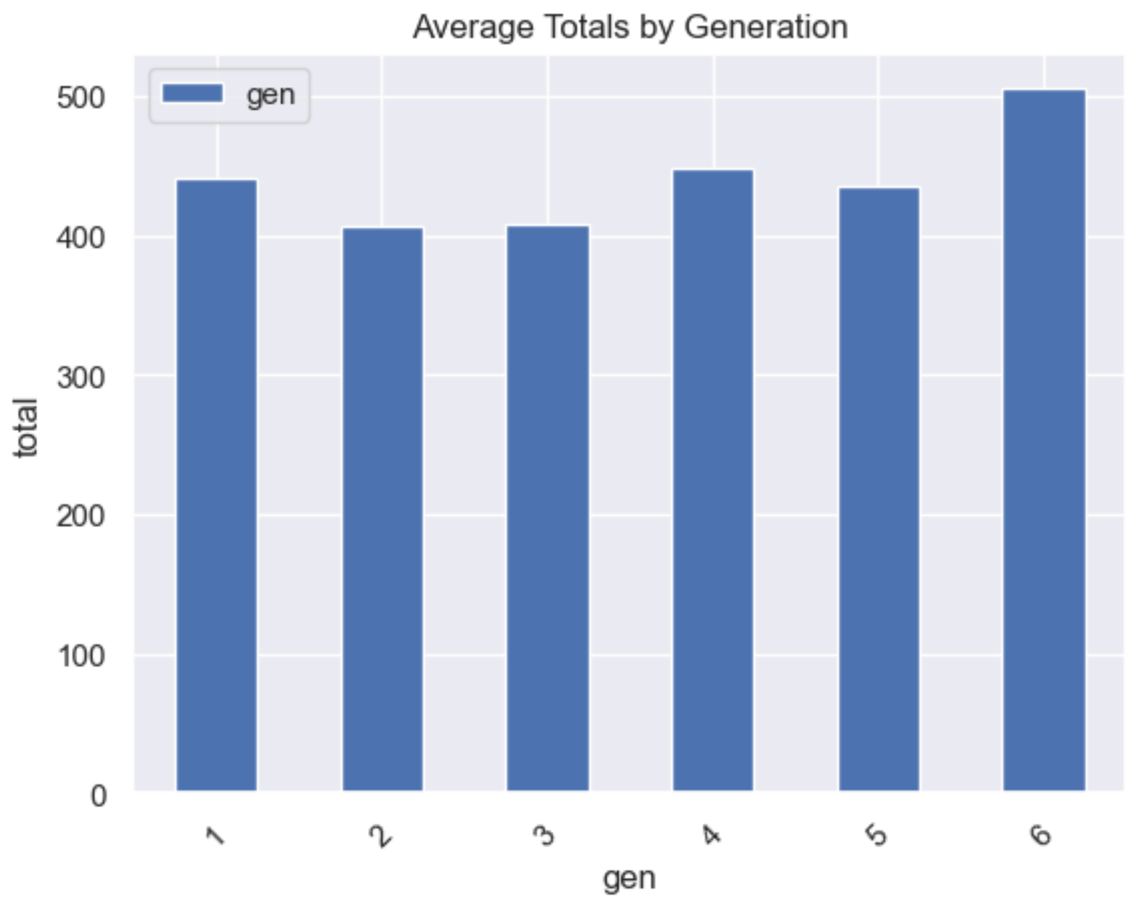
	Number	Name	Type1	Type2	HP	Attack	Defense	Sp.Attack	Sp.Defense	Speed
1058	890	Eternatus	Poison	Dragon	255	115	250	125	250	130
14	150	Mewtwo	Psychic	NaN	106	150	70	194	120	140
13	150	Mewtwo	Psychic	Fighting	106	190	100	154	100	130
475	384	Rayquaza	Dragon	Flying	105	180	100	180	100	115
42	384	Rayquaza	Dragon	Flying	105	180	100	180	100	115
201	150	Mewtwo	Psychic	Fighting	106	190	100	154	100	130
202	150	Mewtwo	Psychic	NaN	106	150	70	194	120	140
40	382	Kyogre	Water	NaN	100	150	90	180	160	90
473	383	Groudon	Ground	Fire	100	180	160	150	90	90
471	382	Kyogre	Water	NaN	100	150	90	180	160	90
41	383	Groudon	Ground	Fire	100	180	160	150	90	90
961	800	Necrozma	Psychic	Dragon	97	167	97	167	97	129
117	493	Arceus	Normal	NaN	120	120	120	120	120	120
605	493	Arceus	Normal	NaN	120	120	120	120	120	120
865	718	Zygarde	Dragon	Ground	216	100	121	91	95	85

```
Pivoted the data set by the average of total attribute and as we see Generation 6,
4, and 1 have the highest averages.
```

Out[23]:

	total
gen	
1	440.900000
2	407.180000
3	408.248227
4	447.898305
5	434.896970
6	505.610687

```
This Barplot graphically shows Generation 6, 4, and 1 have the highest averages.
AxesSubplot(0.125,0.11;0.775x0.77)
```



```
This pivot highlights the count and mean by type and generation.
```

	count						mean					
gen	**1**	**2**	**3**	**4**	**5**	**6**	**1**	**2**	**3**	**4**	**5**	**6**
Type1												
Bug	91	10	12	12	18	7	381	395	327	348	416	445
Dark	56	5	4	3	13	6	449	438	371	538	417	515
Dragon	47	0	7	3	9	10	529	0	527	436	575	612
Electric	73	6	4	12	8	5	447	383	395	472	446	477
Fairy	31	5	0	1	0	9	449	323	0	545	0	450
Fighting	49	2	4	2	7	5	457	332	350	405	429	495
Fire	75	8	7	5	9	12	455	444	414	477	431	505
Flying	10	0	0	0	2	2	450	0	0	0	580	390
Ghost	47	1	4	7	5	12	438	435	375	530	390	437
Grass	104	9	12	14	15	8	418	377	392	450	411	497
Ground	46	3	6	4	10	1	439	420	436	475	445	770
Ice	43	4	7	3	6	3	438	333	430	511	426	466
Normal	131	15	18	17	18	8	409	391	372	427	416	463
Poison	46	1	3	6	2	2	430	535	409	404	401	407
Psychic	82	7	11	7	14	10	486	476	446	490	411	596
Rock	67	4	8	6	6	11	447	430	446	417	455	517
Steel	43	2	9	3	4	9	485	487	463	493	460	518
Water	149	18	25	13	19	11	435	420	406	443	433	536

Q2 Description of the Program

I utilized python pandas, ipython.display, and seaborn packages for the data wrangling and to help visualize the data. The python pandas programming is efficient for manipulating and analyzing data. First using sort I was able to list the totals in descending order. I then used the pivot table function to transform the dataset to generations while aggregating the mean. A standard barplot visualized the data trending clearly. Finally, using pivot table and seaborn for visual effect I grouped, by type and generation, the count and mean. These python tools demonstrates the flexibility in comparing, reshaping, and pivoting of a structured datasets.

Q2 Description of the Output and Analysis

After reviewing the generational analysis I can conclude that the overall average totals slope larger as the generations progress from 1 through 6 as demonstrated by the Average Totals by Generation barplot above. We also see that the counts are significantly lower as compared to the first generation which also reaffirms the later generations have to have higher individual averages to raise their total averages. So, if you looking for a winning Pokémon for battles, it is probably better to acquire a newer generation Pokémon.

Question 3

What is being discussed most on twitter about Pokémon? Sentiment Analysis of the twitter text as well as trending hashtags.

Out[26]:

	Datetime	Tweet Id	Text	User
0	2022-11-30 23:58:54+00:00	1598104315549593600	Perhaps Pokemon's foray onto the Switch is a s...	Renegade_Fox
1	2022-11-30 23:47:31+00:00	1598101448877015040	@jdrider02 People that ask smash movie should ...	Chikle_TikalTak
2	2022-11-30 23:37:18+00:00	1598098876560351232	These were my champion teams for Pokémon Scarl...	iTyero
3	2022-11-30 23:28:58+00:00	1598096782856421376	there's still so much of Pokémon metagame i do...	GammaMesarthim
4	2022-11-30 23:27:50+00:00	1598096494095400962	Tinkaton is the best Pokemon of the generation	ottdogbuns
...	...	...	...	...
996	2022-11-27 17:27:19+00:00	1596918605072936960	I'm really digging this new generation of Poké...	_ShaneTheShaman
997	2022-11-27 17:19:48+00:00	1596916712447672320	The ancient and future pokemon are going to be...	BiHoBeetle
998	2022-11-27 17:17:56+00:00	1596916245131300864	I'm gonna try something, also you can try this...	Cow_The_God
999	2022-11-27 17:08:15+00:00	1596913808232312833	@jordantyranny The current generation, Scarlet...	jordanclock
1000	2022-11-27 17:05:52+00:00	1596913207092080641	The pokemon designs this generation were hones...	Windigo_go

1001 rows × 4 columns

```
Examining the first 5 tweets in the Sentiment Intensity Analyzer:

Perhaps Pokemon's foray onto the Switch is a sign that it is time for Game Freak to
re-evaluate and 'evolve' the generational formula. "Less is more", and perhaps Gene
ration VIII and IX are prime examples of an extended timeframe being required for d
evelopment.
compound: -0.4404, neg: 0.066, neu: 0.934, pos: 0.0,
@jdrider02 People that ask smash movie should realize that the only thing that will
sastify them is either a new subspace emissary or a animated series that works like
those pokemon generations and whatever name they had
compound: 0.3612, neg: 0.0, neu: 0.932, pos: 0.068,
These were my champion teams for Pokémon Scarlet &amp; Violet! I prefer my Violet t
eam (right), but there are a ton of cool Pokémon in this generation so far. And thi
s is before I even had any of the post-game mons. https://t.co/yQsWystt2j
compound: 0.69, neg: 0.0, neu: 0.867, pos: 0.133,
there’s still so much of Pokémon metagame i don’t understand like why does everyone
ever have Leftovers on everything every generation it’s always healed so little how
does it matter like literally at all in the grand scheme of anything
compound: 0.7964, neg: 0.0, neu: 0.794, pos: 0.206,
Tinkaton is the best Pokemon of the generation
compound: 0.6369, neg: 0.0, neu: 0.625, pos: 0.375,

Examining the first 10 tweets through the TextBlob sentiment, calculating polarity
and subjectivity:
```

```
Out[28]: [Sentiment(polarity=-0.4, subjectivity=0.4),
          Sentiment(polarity=0.16666666666666669, subjectivity=0.2833333333333333),
          Sentiment(polarity=0.08522727272727272, subjectivity=0.7272727272727273),
          Sentiment(polarity=0.24523809523809523, subjectivity=0.7285714285714286),
          Sentiment(polarity=0.0, subjectivity=0.0),
          Sentiment(polarity=0.19777462121212122, subjectivity=0.4401515151515151),
          Sentiment(polarity=-0.05, subjectivity=0.08333333333333334),
          Sentiment(polarity=0.0, subjectivity=0.0),
          Sentiment(polarity=0.0, subjectivity=0.0),
          Sentiment(polarity=0.0, subjectivity=0.0)]
```

```
Examining the overall tweets through the TextBlob sentiment, calculating polarity a
nd subjectivity:
```

Out[29]:

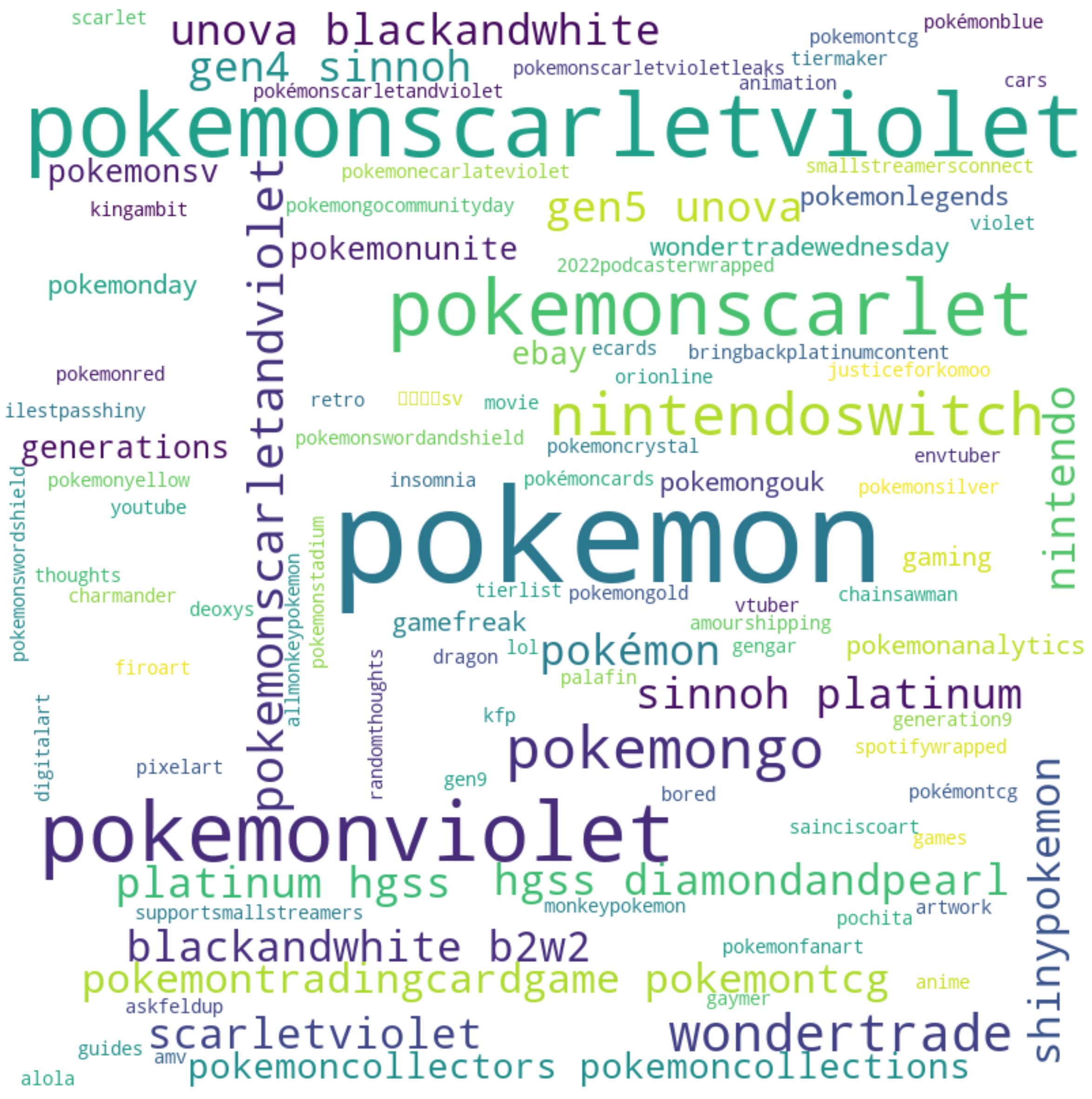
Polarity is positive 😊
Sentence is objective 🧑‍🏫

```
Examining the overall tweets extracting the hashtags.


The following are items are in this DataFrame: hashtag    300
dtype: int64
Below are the first 15 with their frequency count:
```

```
Out[31]: Pokemon                     36
         PokemonScarletViolet        20
         PokemonViolet               15
         PokemonScarlet              14
         pokemon                     11
         NintendoSwitch               8
         WonderTrade                  6
         B2W2                         4
         Unova                        4
         PokemonTCG                   4
         ShinyPokemon                 4
         ScarletViolet                4
         PokemonScarletandviolet      4
         PokemonGO                    4
         PokemonCollectors            4
         Name: hashtag, dtype: int64
```

Below is a WordCloud of all the extracted hashtags:



Q3 Description of the Program

I utilized python pandas, snscrape, nltk, textblob, ipython.display, and wordcloud packages for the data wrangling and to help visualize the data. The python pandas programming is efficient for manipulating and analyzing data. First using snscrape, I was able to obtain 1000 tweets with around 6 lines of code. I then tested the text of the tweets through the sentiment analyzer nltk and textblob to determine that the overall text polarity is positive and the subjectivity is objective. We then used text splitting and count to obtain the hashtags values and frequencies allowing us to see what hashtag is most used. These python tools demonstrates the flexibility in comparing, reshaping, and pivoting of semi-structured datasets.

Q3 Description of the Output and Analysis

It was quite noticeable that one-third (5 out of 15) of these hashtags are talking about Scarlet and Violet. That is most likely because of the recent release date. What I was not sure about was why NintendoSwitch was so popular in the Pokémon twitter text? Then, I came across this article which explained it: "Pokémon Scarlet and Pokémon Violet is currently set to launch November 18, 2022... Pokémon Scarlet and Pokémon Violet will continue the series tradition of **being released exclusively on Nintendo's latest hardware**, in this case, the Nintendo Switch. If you're a Pokémon fan... you already have a Switch ready and waiting." (Source: https://www.digitaltrends.com/gaming/pokemon-scarlet-violet-release-date-trailer-gameplay-news/)

Conclusion | Summary

In this report I acquired, accessed, and transformed data from a structured data source. I used python pandas, pandasql, and numpy packages for the data wrangling. I used matplotlib and seaborn to help visualize the data. Throughout this report, I referenced the data dictionary to help explain the data, the data types to help analyze the data, and any areas of the data that might have required cleaning was cleaned. With the data normalized, I explored some questions to answer regarding Pokémon. My main observation is that, I have a lot more to learn about Pokémon. There is a lot of information about Pokémon and how they calculate battles is still somewhat of a mystery to me. In this report, as noted above, the numbering of Pokémon should be standardized but the two original datasets I retrieved from different sources had different numbers assigned to the Pokémon or some of the Pokémon where missing. This made comparisons more difficult, added confusion, and certainly brought doubt about the completeness and/or accuracy of the population overall. The data set (pkmn) I compiled from pokemondb seemed to have more complete information. By reviewing the total(stat) I can conclude that the type Dragon Pokémon in the later generation yield the higher average total(stat) making them terrific Pokémon all around. The more I learn, the more I realize I do not know about Pokémon. There is more analysis that can be done on Pokémon to include: rules comparison of the various games, cost analysis between generations, and of course what the consumers are saying on social media. I hope this introduction to Pokémon was helpful and interesting.