In [1]:
```
import pandas as pd
import numpy as np
from numpy import NaN
from sklearn.preprocessing import OneHotEncoder
```

In [2]:
```
#There are 4 datasets that we are looking to combine for analysis.
#Before combining the 4 datasets, for each dataset, we need to FIRST, check for valid en
#by comparing the entries with the possible valid entries found in the data dictionary
#provided with the datasets, SECOND, we need to check for duplicate occurences of the
#Incident Number as we will need to have UNIQUE incident numbers to join on.
```

In [3]:
```
##Importing Data Dictionary. Will do Data clearning on columns by comparing entriess wit
DataDictionary = pd.read_csv("/Users/nerdbear/Downloads/1. pca_national_human_wildlife_c
```

In [4]:
```
DataDictionary.head()
```

Out[4]:

	Data_Field	Champ_de_la_donnée	Data_Value	Valeur_de_la_donnée	Value_Description	Description_d
0	Activity Type	Type d'activité	Backpacking – Multiday Trips	Randonnée pédestre – excursion de plusieurs jours	Backpacking – Multiday Trips	Randonné excursion
1	Activity Type	Type d'activité	Beach Recreation	Activitée de plage	Beach Recreation	Activ
2	Activity Type	Type d'activité	Boating - Coastal/Marine	Navigation - côtière/marin	Boating - Coastal/Marine	Navigation -
3	Activity Type	Type d'activité	Boating - Commercial	Navigation - Commerciale	Boating - Commercial	Navigation -
4	Activity Type	Type d'activité	Boating - Motorized Pleasure Craft	Navigation de plaisance – embarcation motorisée	Boating - Motorized Pleasure Craft	Navigation d embarcati

In [5]:
```
#Drop french columns
DataDictionary = DataDictionary.drop(["Champ_de_la_donnée", "Valeur_de_la_donnée", "Desc
```

In [6]:
```
DataDictionary.head()
```

Out[6]:

	Data_Field	Data_Value	Value_Description
0	Activity Type	Backpacking – Multiday Trips	Backpacking – Multiday Trips
1	Activity Type	Beach Recreation	Beach Recreation
2	Activity Type	Boating - Coastal/Marine	Boating - Coastal/Marine
3	Activity Type	Boating - Commercial	Boating - Commercial
4	Activity Type	Boating - Motorized Pleasure Craft	Boating - Motorized Pleasure Craft

In [7]:
```
DataDictionary["Data_Field"].unique()
```

Out[7]:
```
array(['Activity Type', 'Animal Attractant', 'Animal Behaviour',
       'Animal Health Status', 'Cause of Animal Health Status',
       'Deterrents/Projectiles used', 'Field Unit', 'Incident Type',
       'Month', 'Protected Heritage Area', 'Reason for Animal Behaviour',
       'Response to Deterrent', 'Response Type', 'Species Common Name'],
      dtype=object)
```

In [8]:
```
#1 of 4 datasets
```

```
##
```

In [9]:
```
Activities = pd.read_csv("/Users/nerdbear/Downloads/3. pca-human-wildlife-coexistence-ac
#Note, encoding='cp1252' needed to be specified in order to read .csv withour parser err
```

In [10]:
```
Activities.head()
```

Out[10]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Activity Type
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	NaN
1	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Driving
2	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Driving
3	JNP2010-0023	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Railway
4	JNP2010-0016	2010-01-02	Jasper Field Unit	Jasper National Park of Canada	Railway

In [11]:
```
Activities.shape
```

Out[11]:
```
(66284, 5)
```

In [12]:
```
Activities.dtypes
```

Out[12]:
```
Incident Number            object
Incident Date              object
Field Unit                 object
Protected Heritage Area    object
Activity Type              object
dtype: object
```

In [13]:
```
#First we are going to clean the data to ensure valid entries on the string values by co
```

In [14]:
```
#Checking to see how many values in Activity Type do not match values in dictionary.
Activities["Activity Type"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Field
```

Out[14]:
```
60363
```

In [15]:
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Activities.shape[0] - Activities["Activity Type"].isin(DataDictionary["Data_Value"][Data
#There are 5921 entries that do not match values found in dictionary. Next lines of code
```

Out[15]:
```
5921
```

In [16]:
```
#Add column to dataframe that indicates which values match dictionary (True) and which d
Activities["Activity_Type_Dict"] = Activities["Activity Type"].isin(DataDictionary["Data
)
Activities.head()
```

Out[16]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Activity Type	Activity_Type_Dict
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	NaN	False
1	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Driving	True
2	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Driving	True
3	JNP2010-0023	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Railway	True

4	JNP2010-0016	2010-01-02	Jasper Field Unit	Jasper National Park of Canada	Railway	True

In [17]:
```
#Print values that do not match dictionary to see which need to be replaced.
Activities["Activity Type"][Activities["Activity_Type_Dict"]== False].unique()
```

Out[17]:
```
array([nan, 'Hiking/Walking', 'Camping-Frontcountry',
       'Resource Harvesting - Hunting', 'Picnicking/BBQ', 'Sightseeing',
       'Hiking', 'Avoidance', 'Niking / Walking', 'Picknicking / BBQ',
       'Biking / Walking', 'Wiking / Walking', 'Docking - TINP Only'],
      dtype=object)
```

In [18]:
```
#printing all activity types from dictionary to see which best match the errors listed a
DataDictionary["Data_Value"][DataDictionary["Data_Field"]== "Activity Type"].unique()
```

Out[18]:
```
array(['Backpacking - Multiday Trips', 'Beach Recreation',
       'Boating - Coastal/Marine', 'Boating - Commercial',
       'Boating - Motorized Pleasure Craft', 'Bush Party',
       'Camping - Backcountry', 'Camping - Frontcountry',
       'Camping - Huts and Lodges', 'Camping - Winter Frontcountry',
       'Canoeing - Coastal', 'Canoeing - Flatwater',
       'Canoeing - Swiftwater', 'Canyon exploration -Winter',
       'Canyoneering', 'Caving', 'Climbing - Bouldering',
       'Climbing - Mountaineering', 'Climbing - Technical Rock',
       'Climbing - Waterfall  Ice', 'Commercial Transportation Operation',
       'Cycling', 'Cycling - Mountain Biking',
       'Cycling - Road/Shared Path', 'Cycling - Winter', 'Dog Walking',
       'Dogsledding', 'Domestic Residence Activity', 'Driving',
       'Field Sports', 'Fishing',
       'Flight - BASE Jumping/ Proximity Flying',
       'Flight - Hang-gliding/Parapenting', 'Flight - Helicopter',
       'Flight - HETS', 'Flight - Sightseeing/Site Access',
       'Glacier Discovery Walk', 'Golfing',
       'Heritage Activity - Bird Watching',
       'Heritage Activity - History Activities',
       'Heritage Activity - Photography and Art',
       'Heritage Activity - Sightseeing',
       'Heritage Activity - Wildlife Observation', 'Hiking / Walking',
       'Horse Riding - Day Trip', 'Horse Riding - Multiday',
       'Ice Skating', 'Kayaking - Coastal', 'Kayaking - Flatwater',
       'Kayaking - Swiftwater', 'Mooring', 'None Specific - Emergency',
       'Not Applicable', 'Orienteering / Geocaching', 'Other',
       'Paddleboarding - Coastal', 'Paddleboarding - Flatwater',
       'Paddleboarding - Swiftwater', 'Park Operations',
       'Park Ops  - Avalanche Forecasting',
       'Park Ops - Avalanche Control', 'Park Ops - Search and Rescue',
       'Park Ops - Training', 'Picnicking / BBQ', 'Playground Activities',
       'Rafting - Flatwater', 'Rafting - Swiftwater', 'Railway',
       'Research - Scientific/Social',
       'Resource Harvesting - Hunting/Fishing/Gathering/Trapping',
       'Roller Sports', 'Running - Road', 'Running - Trail',
       'Sail Sports - Day Sailing/Touring',
       'Sail Sports - Traction//Ski/Snowboard Kiting',
       'Sail Sports - Wind / Kite Surfing', 'Scrambling', 'Scuba Diving',
       'Skiing - Crosscountry', 'Skiing/Boarding - Backcountry',
       'Skiing/Boarding - Couloirs',
       'Skiing/Boarding - Ski Resort In Bounds',
       'Skiing/Boarding - Ski Resort Out of Bounds', 'Slackline',
       'Sledding/Tobogganning', 'Snow Coach', 'Snowmobiling',
       'Snowshoeing', 'Special Event - Participative Audience',
       'Special Events - Passive Audience', 'Stakeholder Operations',
       'Surfing', 'Swimming - Cliff Jumping', 'Swimming - Coastal',
       'Swimming - Facilities', 'Swimming - Flat Water',
       'Swimming - Swiftwater', 'Townsite Activity',
```

```
       'Tram/Ski Lift/Gondola', 'Tubing / River Drifting', 'Unknown',
       'Via-Ferrata'], dtype=object)
```

In [19]:
```
#Replacing Activity Types that were mis-entered with their proper type, if none was obvi
#Not replacing NaN values with "Unknown". Will look at missing values closer later after

Activities["Activity Type"] = Activities["Activity Type"].replace({"Docking - TINP Only"
```

In [20]:
```
#Counts number of True values
Activities["Activity Type"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Field
```

Out[20]:
```
0        False
1         True
2         True
3         True
4         True
         ...
66279     True
66280     True
66281     True
66282     True
66283     True
Name: Activity Type, Length: 66284, dtype: bool
```

In [21]:
```
#Checking again (after replacement) to see if any Activity Type values still do NOT matc
Activities.shape[0] -  Activities["Activity Type"].isin(DataDictionary["Data_Value"][Dat
```

Out[21]:
```
5885
```

In [22]:
```
Activities["Activity Type"].isna().sum()
#All remaining are missing values.
```

Out[22]:
```
5885
```

In [23]:
```
#Checking to see how many values in Protected Heritage Area do not match values in dicti
Activities.shape[0] - Activities["Protected Heritage Area"].isin(DataDictionary["Data_Va
).sum()
#There are none that are not in dictionary. No replacements needed.
```

Out[23]:
```
0
```

In [24]:
```
#Checking to see how many values in Field Unit do not match values in dictionary
Activities.shape[0] - Activities["Field Unit"].isin(DataDictionary["Data_Value"][DataDic
).sum()
#There are none that are not in dictionary. No replacements needed.
```

Out[24]:
```
0
```

In [25]:
```
#Drop the columns I added during cleaning that are no longer needed
Activities = Activities.drop(["Activity_Type_Dict"], axis=1)
Activities.head()
```

Out[25]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Activity Type
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	NaN
1	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Driving
2	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Driving
3	JNP2010-0023	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Railway
4	JNP2010-0016	2010-01-02	Jasper Field Unit	Jasper National Park of Canada	Railway

In [26]:
```
#Next, we're looking for duplicate occurances of the incident number to ensure our final
```

In [27]:
```
Act_subset = Activities[["Incident Number", "Incident Date", "Field Unit", "Protected He
duplicate_Act_subset = Act_subset.duplicated(keep=False)
sum(duplicate_Act_subset)
```

Out[27]:
```
4051
```

In [28]:
```
duplicate_Act_Inc_Num = Activities.duplicated(subset="Incident Number", keep=False)
sum(duplicate_Act_Inc_Num)
```

Out[28]:
```
4051
```

In [29]:
```
sum(duplicate_Act_Inc_Num)==sum(duplicate_Act_subset)
```

Out[29]:
```
True
```

In [30]:
```
#Conclusion, The Activity Type column is the column that differs between rows - all othe
```

In [31]:
```
#I would like to encode Acitivity Type so each distinct activity type is it's own column
```

In [32]:
```
#Count distinct values in Activity Type
Activities["Activity Type"].nunique()
```

Out[32]:
```
88
```

In [33]:
```
encoder = OneHotEncoder(handle_unknown='ignore')
```

In [34]:
```
encoder_df = pd.DataFrame(encoder.fit_transform(Activities[["Activity Type"]]).toarray()
```

In [35]:
```
encoder_df.columns = encoder.get_feature_names_out(["Activity Type"])
```

In [36]:
```
encoder_df.head()
```

Out[36]:

	Activity Type_Backpacking – Multiday Trips	Activity Type_Beach Recreation	Activity Type_Boating - Coastal/Marine	Activity Type_Boating - Commercial	Activity Type_Boating - Motorized Pleasure Craft	Activity Type_Bush Party	Activi Type_Campi - Backcount
0	0.0	0.0	0.0	0.0	0.0	0.0	0
1	0.0	0.0	0.0	0.0	0.0	0.0	0
2	0.0	0.0	0.0	0.0	0.0	0.0	0
3	0.0	0.0	0.0	0.0	0.0	0.0	0
4	0.0	0.0	0.0	0.0	0.0	0.0	0

5 rows × 89 columns

In [37]:
```
Activities_encoded = Activities.join(encoder_df)
```

In [38]:
```
Activities_encoded.head()
```

Out[38]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Activity Type	Activity Type_Backpacking – Multiday Trips	Activity Type_Beach Recreation	Activity Type_Boating - Coastal/Marine	Type_ - Con

0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	NaN	0.0	0.0	0.0
1	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Driving	0.0	0.0	0.0
2	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Driving	0.0	0.0	0.0
3	JNP2010-0023	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Railway	0.0	0.0	0.0
4	JNP2010-0016	2010-01-02	Jasper Field Unit	Jasper National Park of Canada	Railway	0.0	0.0	0.0

5 rows × 94 columns

In [39]:
```
Activities_encoded.drop('Activity Type', axis = 1, inplace=True)
```

In [40]:
```
Activities_encoded.head()
```

Out[40]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Activity Type_Backpacking – Multiday Trips	Activity Type_Beach Recreation	Activity Type_Boating - Coastal/Marine	Activity Type_Boating - Commercial
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0
1	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	0.0	0.0	0.0	0.0
2	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	0.0	0.0	0.0	0.0
3	JNP2010-0023	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	0.0	0.0	0.0	0.0
4	JNP2010-0016	2010-01-02	Jasper Field Unit	Jasper National Park of Canada	0.0	0.0	0.0	0.0

5 rows × 93 columns

In [41]:
```
Activities_encoded[Activities_encoded.columns[4:105]]
```

	Activity	Activity	Activity	Activity	Activity	Activity

Out[41]:

	Type_Backpacking – Multiday Trips	Type_Beach Recreation	Type_Boating - Coastal/Marine	Type_Boating - Commercial	Type_Boating - Motorized Pleasure Craft	Type_Bush Party	Type_C - Back
0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	
66279	0.0	0.0	0.0	0.0	0.0	0.0	
66280	0.0	0.0	0.0	0.0	0.0	0.0	
66281	0.0	0.0	0.0	0.0	0.0	0.0	
66282	0.0	0.0	0.0	0.0	0.0	0.0	
66283	0.0	0.0	0.0	0.0	0.0	0.0	

66284 rows × 89 columns

In [42]:
```
Activities2 = Activities_encoded[Activities_encoded.columns[4:105]].groupby([Activities[
Activities2.head()
```

Out[42]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Activity Type_Backpacking – Multiday Trips	Activity Type_Beach Recreation	Activity Type_Boating - Coastal/Marine	Activity Type_Boating - Commercial
0	2017-HWC-0005-YKLLFU-0001	2017-08-01	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0
1	2017-HWC-0005-YKLLFU-0002	2017-09-07	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0
2	2017-HWC-0005-YKLLFU-0003	2017-07-08	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0
3	2017-HWC-0005-YKLLFU-0004	2017-06-23	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0
4	2017-HWC-0005-YKLLFU-0006	2017-06-28	Lake Louise, Yoho and Kootenay	Banff National Park of Canada	0.0	0.0	0.0	0.0

Field Unit

5 rows × 93 columns

In [43]:
```
# The way I've merged the encoded activity type columns using the (sum) function means t
```

In [44]:
```
#Confirming whether the new dataset has any duplicate incident numbers
duplicate_Act2_Inc_Num = Activities2.duplicated(subset="Incident Number", keep=False)
sum(duplicate_Act2_Inc_Num)
```

Out[44]:
```
0
```

In [45]:
```
dup_Activities = Activities[duplicate_Act_subset]
```

In [46]:
```
#Cross checking to ensure correct number of rows remain.
#Number of rows in Original Dataset, minus (number of rows in duplicates subset minus nu
Activities.shape[0] - (dup_Activities.shape[0] - dup_Activities["Incident Number"].nuniq
```

Out[46]:
```
True
```

In [47]:
```
#(In other words, I want to ensure that our new dataset has the same number of Unique in
Activities["Incident Number"].nunique() == Activities2["Incident Number"].nunique()
```

Out[47]:
```
True
```

In [48]:
```
# 2 of 4 datasets
##
```

In [49]:
```
Animals = pd.read_csv("/Users/nerdbear/Downloads/4. pca-human-wildlife-coexistence-anima
```

In [50]:
```
Animals.head()
```

Out[50]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Species Common Name	Sum of Number of Animals	Animal Health Status	Cause of Animal Health Status
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Coyote	2	Healthy	NaN
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Elk	1	Dead	Predation
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Wolf	3	Not Located	NaN
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Found Wildlife	White-tailed Deer	1	Dead	Collision
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Attractant	None	0	NaN	NaN

In [51]:
```
Animals.shape
```

Out[51]:
```
(73655, 14)
```

In [52]:
```
Animals.dtypes
```

Out[52]:
```
Incident Number                  object
Incident Date                    object
Field Unit                       object
Protected Heritage Area          object
Incident Type                    object
Species Common Name              object
Sum of Number of Animals          int64
Animal Health Status             object
Cause of Animal Health Status    object
Animal Behaviour                 object
Reason for Animal Behaviour      object
Animal Attractant                object
Deterrents Used                  object
Animal Response to Deterrents    object
dtype: object
```

In [53]:
```
#First we are going to clean the data to ensure valid entries on the string values by co
```

In [54]:
```
#Checking to see how many values in Field Unit do not match values in dictionary.
Animals["Field Unit"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Field"]== "
```

Out[54]:
```
73655
```

In [55]:
```
#Shows how many are False, therefore how many Field Units are not in the dictionary.
Animals.shape[0] - Animals["Field Unit"].isin(DataDictionary["Data_Value"][DataDictionar
#There are none that are not in dictionary. No replacements needed.
```

Out[55]:
```
0
```

In [56]:
```
#Checking to see how many values in Protected Heritage Area do not match values in dicti
Animals["Protected Heritage Area"].isin(DataDictionary["Data_Value"][DataDictionary["Dat
```

Out[56]:
```
73655
```

In [57]:
```
#Shows how many are False, therefore how many Field Units are not in the dictionary.
Animals.shape[0] - Animals["Protected Heritage Area"].isin(DataDictionary["Data_Value"][
#There are none that are not in dictionary. No replacements needed.
```

Out[57]:
```
0
```

In [58]:
```
#Checking to see how many values in Incident Type do not match values in dictionary.
Animals["Incident Type"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Field"]=
```

Out[58]:
```
73655
```

In [59]:
```
#Shows how many are False, therefore how many FIncident Type are not in the dictionary.
Animals.shape[0] - Animals["Incident Type"].isin(DataDictionary["Data_Value"][DataDictio
#There are none that are not in dictionary. No replacements needed.
```

Out[59]:
```
0
```

In [60]:
```
#Checking to see how many values in Species Common Name do not match values in dictionar
Animals["Species Common Name"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Fi
```

Out[60]:
```
73653
```

In [61]:
```
#Shows how many are False, therefore how many Species Common Name are not in the diction
Animals.shape[0] - Animals["Species Common Name"].isin(DataDictionary["Data_Value"][Data
#There are 2 that are not in dictionary. Replacements needed.
```

Out[61]:
```
2
```

In [62]:
```
#Add column to dataframe that indicates which values match dictionary (True) and which d
Animals["Species Common Name_Dict"] = Animals["Species Common Name"].isin(DataDictionary
Animals.head()
```

Out[62]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Species Common Name	Sum of Number of Animals	Animal Health Status	Cause of Animal Health Status
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Coyote	2	Healthy	NaN
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Elk	1	Dead	Predation
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Wolf	3	Not Located	NaN
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Found Wildlife	White-tailed Deer	1	Dead	Collision
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Attractant	None	0	NaN	NaN

In [63]:
```
#Print values that do not match dictionary to see which need to be replaced.
Animals["Species Common Name"][Animals["Species Common Name_Dict"]== False].unique()
```

Out[63]:
```
array(['Banff Spring Snail', 'Eurasian red squirrel'], dtype=object)
```

In [64]:
```
#printing all species common name from dictionary to see which best match the errors lis
DataDictionary["Data_Value"][DataDictionary["Data_Field"]== "Species Common Name"].uniqu
```

Out[64]:
```
array(['American Coot', 'American Dipper', 'American Dog Tick',
       'American eel', 'American Kestrel', 'American Robin',
       'American sand lance', 'American Toad', 'American Tree Sparrow',
       'Ant', 'Arctic Fox', 'Arctic Ground Squirrel', 'Atlantic Cod',
       'Atlantic Halibut', 'Atlantic Herring', 'Atlantic Salmon',
       'Atlantic White-sided Dolphin', 'Badger', 'Bald Eagle',
       'Banff Springs Snail', 'Bank Swallow', 'Barn Swallow',
       'Barred Owl', 'Basking Shark', 'Bearded Seal', 'Beaver',
       'Belted Kingfisher', 'Beluga Whale', 'Big Brown Bat', 'Big Skate',
       'Bighorn Sheep', 'Black Bear', 'Black Duck', 'Black Oystercatcher',
       'Black Rat', 'Black Scoter', 'Black Swift', 'Black Widow Spider',
       'Black-billed Murrelet', 'Black-footed Albatross',
       'Black-footed Ferret', 'Black-tailed deer',
       'Black-tailed prairie dog', 'Black-throated Sparrow',
       "Blanding's Turtle", 'Blue Grouse', 'Blue Jay', 'Blue Shark',
       'Blue Whale', 'Blueback herring', 'Bluefin tuna',
       'Blue-winged Teal', 'Bluntnose Sixgill Shark', 'Bobcat',
       'Bobolink', 'Bohemian Waxwing', 'Boreal Chorus Frog', 'Boreal Owl',
```

'Broad-winged Hawk', 'Brook Trout', 'Brown Dog Tick', 'Brown Rat',
'Brown Recluse Spider', 'Brown-headed cowbird', 'Bufflehead',
'Bullfrog', 'Bullsnake', 'Burrowing Owl', 'California Gull',
'California Sea Lion', 'Canada Goose', 'Canadian Toad',
'Canvasback', 'Capelin', 'Caribou', 'Cave Swallow',
'Cedar Waxwing', 'Chestnut-backed Chickadee', 'Chipping Sparrow',
'Cliff Swallow', 'Columbian Ground Squirrel', 'Common Eider',
'Common Gartersnake', 'Common Goldeneye', 'Common Loon',
'Common Merganser', 'Common Murre', 'Common Nighthawk',
'Common Redpoll', 'Common Tern', "Cooper's Hawk", 'Cougar',
'Coyote', 'Crow', "Dall's Porpoise", "Dall's sheep",
'Dark-eyed Junco', 'Deer Mouse', "DeKay's Brownsnake",
'Domestic Bison', 'Domestic Cat', 'Domestic Cattle',
'Domestic Chicken', 'Domestic Dog', 'Domestic Donkey',
'Domestic ferret', 'Domestic Goat', 'Domestic Horse',
'Domestic pig', 'Domestic pigeon', 'Domestic rat',
'Domestic Sheep', 'Double-crested Cormorant', 'Downy Woodpecker',
'Earwigs', 'Eastern Box Turtle', 'Eastern Chipmunk',
'Eastern Foxsnake', 'Eastern Grey Squirrel',
'Eastern Hog-nosed Snake', 'Eastern Musk Turtle',
'Eastern Painted Turtle', 'Eastern Ribbonsnake', 'Eastern Wolf',
'Elk', 'Ermine', 'Ermine haidarum', 'European rabbit',
'European Starling', 'Fallow Deer', 'Ferruginous Hawk',
'Field Sparrow', 'Fin Whale', 'Fisher',
'Five-lined Skink Carolinian',
'Five-lined Skink Great Lakes St Lawrence', "Franklin's Gull",
'Glaucous Gull', 'Golden Eagle', 'Golden-crowned Kinglet',
'Golden-mantled Ground Squirrel', 'Gopher snake', 'Gray Jay',
'Great Black-backed Gull', 'Great Blue Heron', 'Great Grey Owl',
'Great Horned Owl', 'Great White Shark', 'Greater Sage-grouse',
'Greater Short-horned Lizard', 'Greater White-fronted Goose',
'Green Sea Turtle', 'Green Sea Urchin', 'Greenland Cod',
'Greenland Halibut', 'Greenland Shark', 'Green-winged Teal',
'Grey Seal', 'Grey Whale', 'Grizzly Bear', 'Guadalupe fur seal',
'Hairy Woodpecker', 'Harbour Porpoise', 'Harbour seal',
'Harlequin duck', 'Harp Seal', 'Herring Gull', 'Hoary Bat',
'Hoary Marmot', 'Honey bee', 'Hooded Merganser', 'Hooded Seal',
'Horned Grebe', 'House Sparrow', 'Humpback Whale', 'Jackrabbit',
"Keen's Long-eared Bat", "Kemp's Ridley Sea Turtle", 'Killdeer',
'Killer Whale', 'Lake Whitefish', 'Least Chipmunk', 'Least Weasel',
'Leatherback Sea Turtle', "Lewis's Woodpecker",
"Lincoln's Sparrow", 'Little Brown Myotis',
'Loggerhead Sea Turtle', 'Loggerhead Shrike Prairie',
'Lone Star Tick', 'Long-beaked Common Dolphin',
'Long-eared Myotis', 'Long-eared Owl', 'Long-finned Pilot Whale',
'Long-legged Myotis', 'Longnose lancetfish', 'Long-tailed Duck',
'Long-tailed Weasel', 'Long-toed Salamander', 'Lynx', 'Magpie',
'Mallard', 'Marten', 'Massasauga', 'Merlin',
'Midland Painted Turtle', 'Mink', 'Minke Whale', 'Moose',
'Mosquito', 'Mountain Goat', 'Mountain Whitefish', 'Mule Deer',
'Muskox', 'Muskrat', 'Mute Swan', 'Narwhal', 'Newfoundland Marten',
'None', 'Northern Alligator Lizard', 'Northern Bottlenose Whale',
'Northern Elephant Seal', 'Northern Flicker',
'Northern Flying Squirrel', 'Northern Fur Seal', 'Northern Gannet',
'Northern Goshawk', 'Northern Harrier', 'Northern Hawk Owl',
'Northern Leopard Frog', 'Northern Map Turtle', 'Northern Myotis',
'Northern Pacific Rattlesnake', 'Northern Pygmy-Owl',
'Northern Right-whale Dolphin', 'Northern Rough-winged Swallow',
'Northern Saw-whet Owl', 'Northern Saw-whet Owl brooksi',
'Northern Shoveler', 'Northern Spring Peeper',
'Northern Waterthrush', 'Northwestern Gartersnake',
'Ocean Sunfish', 'Olive Ridley Sea Turtle', 'Opossum',
'Orange-crowned Warbler', 'Osprey',
'Pacific Coast Western Painted Turtle', 'Pacific Gophersnake',
'Pacific Salmon', 'Peary Caribou', 'Peregrine Falcon', 'Pika',
'Pileated Woodpecker', 'Pine Grosbeak', 'Pine Siskin',

```
       'Piping Plover', 'Plains Bison', 'Plains Gartersnake',
       'Polar Bear', 'Pond Slider', 'Porbeagle shark', 'Porcupine',
       'Prairie Rattlesnake', 'Pronghorn', 'Prothonotary Warbler',
       'Pygmy Whitefish', 'Raccoon', 'Rainbow Smelt', 'Raven',
       'Razorbill', 'Red Crossbill', 'Red Fox', 'Red Squirrel',
       'Red-bellied Snake', 'Red-breasted Merganser',
       'Red-breasted Nuthatch', 'Redfish', 'Redhead Duck',
       'Red-necked Grebe', 'Red-tailed Chipmunk', 'Red-tailed Hawk',
       'Red-winged Blackbird', "Richardson's Ground Squirrel",
       'Ring-billed Gull', 'Ring-necked Pheasant', 'Ring-necked Snake',
       'River Otter', 'Rock Pigeon', 'Rocky Mountain Wood Tick',
       'Rough-legged Hawk', 'Rubber Boa', 'Ruby-crowned Kinglet',
       'Ruby-throated Hummingbird', 'Ruddy Duck', 'Ruffed Grouse',
       'Rufous Hummingbird', 'Salmon Shark', 'Sandhill Crane',
       'Sea Otter', 'Semipalmated Plover', 'Sharp-shinned Hawk',
       'Sharp-tailed Grouse', 'Sharp-tailed Snake', 'Short-eared Owl',
       'Shortfin Mako Shark', 'Shortnose Lancetfish', 'Silver-haired Bat',
       'Skunk', 'Smooth Greensnake', 'Snapping Turtle', 'Snow Goose',
       'Snowshoe Hare', 'Snowy Owl', 'Song Sparrow',
       "Sowerby's Beaked Whale", 'Sperm Whale', 'Spiny Softshell',
       'Spotted Salamander', 'Spotted Turtle', 'Spruce Grouse',
       'Steller Sea Lion', "Steller's Jay", 'Striped Dolphin',
       'Surf Scoter', "Swainson's Hawk", "Swainson's Thrush", 'Swift Fox',
       'Tennessee Warbler', 'Terrestrial Gartersnake',
       'Thirteen-lined Ground Squirrel', 'Tiger Salamander',
       'Tree Swallow', 'Trumpeter swan', 'Turkey Vulture', 'Unknown',
       'Unknown bat', 'Unknown bear', 'Unknown bird', 'Unknown Bison',
       'Unknown canid', 'Unknown deer', 'Unknown Duck', 'Unknown felid',
       'Unknown fish', 'Unknown Frog or Toad', 'Unknown grouse',
       'Unknown gull', 'Unknown hawk', 'Unknown Mollusk', 'Unknown Mouse',
       'Unknown mustelid', 'Unknown Myotis bat', 'Unknown Octopus',
       'Unknown owl', 'Unknown raptor', 'Unknown rodent',
       'Unknown sea lion', 'Unknown seal', 'Unknown Shark',
       'Unknown shrew', 'Unknown snake', 'Unknown sucker',
       'Unknown ungulate', 'Unknown whale', 'Varied Thrush',
       'Violet-green Swallow', 'Wasp', 'Water Snake', 'Weevil',
       'Western Black-legged Tick', 'Western Chorus Frog',
       'Western Grebe', 'Western Hognose Snake', 'Western Meadowlark',
       'Western Painted Turtle', 'Western Screech-Owl', 'Western Tanager',
       'Western Toad', 'White Pelican', 'White-crowned Sparrow',
       'White-tailed Deer', 'White-tailed Ptarmigan',
       'White-winged Crossbill', 'Whooping Crane', 'Wild Boar',
       'Wild Horse', 'Wild Turkey', 'Willow Ptarmigan', "Wilson's Snipe",
       'Winter Wren', 'Wolf', 'Wolverine', 'Wood Bison', 'Wood Duck',
       'Wood Frog', 'Wood Turtle', 'Woodchuck', 'Woodrat',
       'Yellow Warbler', 'Yellow-bellied Marmot', 'Yellow-bellied Racer',
       'Yellow-bellied Sapsucker', 'Yellow-pine Chipmunk',
       'Yellow-throated Warbler'], dtype=object)
```

In [65]:
```
#Replacing Activity Types that were mis-entered with their proper type, if none was obvi
Animals["Species Common Name"] = Animals["Species Common Name"].replace({"Banff Spring S
```

In [66]:
```
#Recehcking how many are False, therefore how many Species Common Name are not in the di
Animals.shape[0] - Animals["Species Common Name"].isin(DataDictionary["Data_Value"][Data
#There are none that are not in dictionary. No replacements needed.
```

Out[66]:
```
0
```

In [67]:
```
#Checking to see how many values in Animal Health Status do not match values in dictiona
Animals["Animal Health Status"].isin(DataDictionary["Data_Value"][DataDictionary["Data_F
```

Out[67]:
```
41477
```

In [68]:
```
#Shows how many are False, therefore how many Animal Health Status are not in the dictio
```

```
Animals.shape[0] - Animals["Animal Health Status"].isin(DataDictionary["Data_Value"][Dat
#There are lots that are not in dictionary. Will look at these deeper to determine if re
```

Out[68]:
```
32178
```

In [69]:
```
#Add column to dataframe that indicates which values match dictionary (True) and which d
Animals["Animal Health Status_Dict"] = Animals["Animal Health Status"].isin(DataDictiona
Animals.head()
```

Out[69]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Species Common Name	Sum of Number of Animals	Animal Health Status	Cause of Animal Health Status
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Coyote	2	Healthy	NaN
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Elk	1	Dead	Predation
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Wolf	3	Not Located	NaN
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Found Wildlife	White-tailed Deer	1	Dead	Collision
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Attractant	None	0	NaN	NaN

In [70]:
```
#Print values that do not match dictionary to see which need to be replaced.
Animals["Animal Health Status"][Animals["Animal Health Status_Dict"]== False].unique()
```

Out[70]:
```
array([nan, 'Not Applicable'], dtype=object)
```

In [71]:
```
#printing all activity types from dictionary to see which best match the errors listed a
DataDictionary["Data_Value"][DataDictionary["Data_Field"]== "Animal Health Status"].uniq
```

Out[71]:
```
array(['Dead', 'Healthy', 'Injured', 'Not Located', 'Orphaned', 'Other',
       'Sick', 'Unknown'], dtype=object)
```

In [72]:
```
#Not replacing NaN values with "Unknown".
#Will look at missing values closer later after after splitting and before modelling.
#I'm going to keep the "Not applicable" entries because those are realistic valid
#entries and not typos/errors.
#No replacements or changes to make.
```

In [73]:
```
#Checking to see how many values in Cause of Animal Health Status do not match values in
Animals["Cause of Animal Health Status"].isin(DataDictionary["Data_Value"][DataDictionar
```

Out[73]:
```
13080
```

In [74]:
```
#Shows how many are False, therefore how many Cause of Animal Health Status are not in t
Animals.shape[0] - Animals["Cause of Animal Health Status"].isin(DataDictionary["Data_Va
#There are plenty that are not in dictionary. No replacements needed.
```

```
60575
```

Out[74]:

In [75]:
```
#Add column to dataframe that indicates which values match dictionary (True) and which d
Animals["Cause of Animal Health Status_Dict"] = Animals["Cause of Animal Health Status"]
Animals.head()
```

Out[75]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Species Common Name	Sum of Number of Animals	Animal Health Status	Cause of Animal Health Status
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Coyote	2	Healthy	NaN
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Elk	1	Dead	Predation
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Wolf	3	Not Located	NaN
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Found Wildlife	White-tailed Deer	1	Dead	Collision
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Attractant	None	0	NaN	NaN

In [76]:
```
#Print values that do not match dictionary to see which need to be replaced.
Animals["Cause of Animal Health Status"][Animals["Cause of Animal Health Status_Dict"]==
```

Out[76]:
```
array([nan, 'Not Applicable'], dtype=object)
```

In [77]:
```
#printing all activity types from dictionary to see which best match the errors listed a
DataDictionary["Data_Value"][DataDictionary["Data_Field"]== "Cause of Animal Health Stat
```

Out[77]:
```
array(['Collision', 'Defence of Life/Property - public', 'Disease',
       'Drowned', 'Entangle-Entrapment', 'Hunting - Trapping',
       'Indigenous Harvest', 'Intraspecific Competition',
       'Management Destruction', 'Natural Mortality', 'Other', 'Poaching',
       'Poisoned', 'Predation', 'Starvation', 'Unknown'], dtype=object)
```

In [78]:
```
#Not replacing NaN values with "Unknown".
#Will look at missing values closer later after splitting and before modelling.
#I'm going to keep the "Not applicable" entries because those are realistic valid
#entries and not typos/errors.
#No replacements or changes to make.
```

In [79]:
```
#Checking to see how many values do not match values in dictionary.
Animals["Animal Behaviour"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Field
```

Out[79]:
```
45674
```

In [80]:
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Animals.shape[0] - Animals["Animal Behaviour"].isin(DataDictionary["Data_Value"][DataDic
#There are several entries that do not match values found in dictionary. Next lines of c
```

```
27981
```

Out[80]:

In [81]:
```
#Add column to dataframe that indicates which values match dictionary (True) and which d
Animals["Animal Behaviour_Dict"] = Animals["Animal Behaviour"].isin(DataDictionary["Data
Animals.head()
```

Out[81]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Species Common Name	Sum of Number of Animals	Animal Health Status	Cause of Animal Health Status
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Coyote	2	Healthy	NaN
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Elk	1	Dead	Predation
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Wolf	3	Not Located	NaN
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Found Wildlife	White-tailed Deer	1	Dead	Collision
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Attractant	None	0	NaN	NaN

In [82]:
```
#Print values that do not match dictionary to see which need to be replaced.
Animals["Animal Behaviour"][Animals["Animal Behaviour_Dict"]== False].unique()
```

Out[82]:
```
array([nan, 'Stress'], dtype=object)
```

In [83]:
```
#printing all activity types from dictionary to see which best match the errors listed a
DataDictionary["Data_Value"][DataDictionary["Data_Field"]== "Animal Behaviour"].unique()
```

Out[83]:
```
array(['Avoidance', 'Bluff Charge', 'Chase', 'Contact-People',
       'Contact-Pet', 'Contact-Property', 'Curious', 'Curious Approach',
       'Dive', 'Escort (Follow-Flank)', 'Indifferent to People/Vehicles',
       'Intense Staring', 'Not Applicable', 'Other',
       'Physical or Aggressive Display', 'Predatory Approach',
       'Presence - Wildlife Exclusion Zones', 'Secretive', 'Unaware',
       'Unknown', 'Unyielding (refuse to give ground)', 'Vocalization'],
      dtype=object)
```

In [84]:
```
#Replacing Stress with other as Stress is not a valid entry per the dictionary, there is
Animals["Animal Behaviour"] = Animals["Animal Behaviour"].replace({"Stress": "Other"})
```

In [85]:
```
#All other values that do not match dictionary are missing. No more replacements needed.
```

In [86]:
```
#Checking to see how many values do not match values in dictionary.
Animals["Reason for Animal Behaviour"].isin(DataDictionary["Data_Value"][DataDictionary[
```

Out[86]:
```
24849
```

In [87]:
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Animals.shape[0] - Animals["Reason for Animal Behaviour"].isin(DataDictionary["Data_Valu
```

```
#There are lots of entries that do not match values found in dictionary. Next lines of c
```

Out[87]:
```
48806
```

In [88]:
```
#Add column to dataframe that indicates which values match dictionary (True) and which d
Animals["Reason for Animal Behaviour_Dict"] = Animals["Reason for Animal Behaviour"].isi
Animals.head()
```

Out[88]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Species Common Name	Sum of Number of Animals	Animal Health Status	Cause of Animal Health Status
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Coyote	2	Healthy	NaN
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Elk	1	Dead	Predation
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Wolf	3	Not Located	NaN
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Found Wildlife	White-tailed Deer	1	Dead	Collision
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Attractant	None	0	NaN	NaN

In [89]:
```
#Print values that do not match dictionary to see which need to be replaced.
Animals["Reason for Animal Behaviour"][Animals["Reason for Animal Behaviour_Dict"]== Fal
```

Out[89]:
```
array([nan, 'Not applicable', 'Entangle-Entrapment'], dtype=object)
```

In [90]:
```
#printing all activity types from dictionary to see which best match the errors listed a
DataDictionary["Data_Value"][DataDictionary["Data_Field"]== "Reason for Animal Behaviour
```

Out[90]:
```
array(['Defence of Food', 'Defence of Mate', 'Defence of Space',
       'Defence of Young', 'Disease', 'Food Conditioned', 'Food Reward',
       'Habituation', 'Not Applicable', 'Predator Avoidance', 'Predatory',
       'Presence of Domestic Animal', 'Starvation', 'Stress', 'Surprise',
       'Unknown'], dtype=object)
```

In [91]:
```
Animals.loc[Animals["Reason for Animal Behaviour"]== "Entangle-Entrapment"]
```

Out[91]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Species Common Name	Sum of Number of Animals	Animal Health Status	Cause of Animal Health Status	Animal Behaviour
9220	YNP2012-0188	2012-07-27	Lake Louise, Yoho and Kootenay Field Unit	Yoho National Park of Canada	Human Wildlife Interaction	Black Bear	1	NaN	NaN	Presence - Wildlife Exclusion Zones

In [92]:
```
#There is only one occurence of "Entangle-Entrapment" in the "Reason for Animal Behaviou
#That value is not valid for that column but it is a valid entry for "Cause of Animal Be
#which was missing for this row/incident entry. Moving to "Cause" column and listing "Re

#Assigning Entangle-Entrapment to Cause column for one row that has that value in Reason
Animals["Cause of Animal Health Status"].loc[Animals["Reason for Animal Behaviour"]== "E

#Replacing Entangle-Entrapment with a missing value as that is not a valid entry for Rea
Animals["Reason for Animal Behaviour"] = Animals["Reason for Animal Behaviour"].replace(

#Replace Not applicable value with correction "Not Applicable"
Animals["Reason for Animal Behaviour"] = Animals["Reason for Animal Behaviour"].replace(

#All other values that are not in dictionary are nan (missing) and will be dealt with la
```

In [93]:
```
#Checking to see how many values do not match values in dictionary.
Animals["Animal Attractant"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Fiel
```

Out[93]:
```
23005
```

In [94]:
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Animals.shape[0] - Animals["Animal Attractant"].isin(DataDictionary["Data_Value"][DataDi
#There are lots of entries that do not match values found in dictionary. Next lines of c
```

Out[94]:
```
50650
```

In [95]:
```
#Add column to dataframe that indicates which values match dictionary (True) and which d
Animals["Animal Attractant_Dict"] = Animals["Animal Attractant"].isin(DataDictionary["Da
Animals.head()
```

Out[95]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Species Common Name	Sum of Number of Animals	Animal Health Status	Cause of Animal Health Status
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Coyote	2	Healthy	NaN
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Elk	1	Dead	Predation
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Wolf	3	Not Located	NaN
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Found Wildlife	White-tailed Deer	1	Dead	Collision
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Attractant	None	0	NaN	NaN

In [96]:
```
#Print values that do not match dictionary to see which need to be replaced.
Animals["Animal Attractant"][Animals["Animal Attractant_Dict"]== False].unique()
```

Out[96]:
```
array([nan, 'Domestic animal', 'Not applicable', 'None', 'Other'],
      dtype=object)
```

In [97]:
```
#printing all activity types from dictionary to see which best match the errors listed a
DataDictionary["Data_Value"][DataDictionary["Data_Field"]== "Animal Attractant"].unique(
```

Out[97]:
```
array(['Berries (natural)', 'Carrion', 'Compost', 'Domestic Animal',
       'Domestic grass', 'Fish', 'Fruit tree, shrub or garden', 'Garbage',
       'Grain', 'Human food', 'Mate', 'Mineral Lick', 'Not Applicable',
       'Petroleum products', 'Prey animal (natural)', 'Road salt',
       'Small animal feeders', 'Unknown', 'Vegetation (natural)'],
      dtype=object)
```

In [98]:
```
Animals["Animal Attractant"][Animals["Animal Attractant"]== "Other"].count()
```

Out[98]:
```
14
```

In [99]:
```
Animals["Animal Attractant"][Animals["Animal Attractant"]== "None"].count()
```

Out[99]:
```
12
```

In [100…
```
#Replacing "Domestic animal" with correct "Domestic Animal"; "Not applicable" to correct
#I will replace "Other" and "None" with missing values as those entries are not valid.

Animals["Animal Attractant"] = Animals["Animal Attractant"].replace({"Domestic animal":

#Only remaining values that are not in dictionary are the missing values.
```

In [101…
```
#Checking to see how many values do not match values in dictionary.
Animals["Deterrents Used"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Field"
```

Out[101]:
```
19540
```

In [102…
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Animals.shape[0] - Animals["Deterrents Used"].isin(DataDictionary["Data_Value"][DataDict
#There are lots of entries that do not match values found in dictionary. Next lines of c
```

Out[102]:
```
54115
```

In [103…
```
#Add column to dataframe that indicates which values match dictionary (True) and which d
Animals["Deterrents Used_Dict"] = Animals["Deterrents Used"].isin(DataDictionary["Data_V
Animals.head()
```

Out[103]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Species Common Name	Sum of Number of Animals	Animal Health Status	Cause of Animal Health Status
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Coyote	2	Healthy	NaN
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Elk	1	Dead	Predation
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Wolf	3	Not Located	NaN
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Found Wildlife	White-tailed Deer	1	Dead	Collision

4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada		Attractant	None	0	NaN	NaN

5 rows × 21 columns

In [104…
```
#Print values that do not match dictionary to see which need to be replaced.
Animals["Deterrents Used"][Animals["Deterrents Used_Dict"]== False].unique()
```

Out[104]:
```
array([nan, 'Impact - Electric Shock'], dtype=object)
```

In [105…
```
#printing all activity types from dictionary to see which best match the errors listed a
DataDictionary["Data_Value"][DataDictionary["Data_Field"]== "Deterrents/Projectiles used
```

Out[105]:
```
array(['Bear Spray', 'Impact - Beanbag', 'Impact - Chalkball',
       'Impact - Paintball', 'Impact - Pellet', 'Impact - Projectile',
       'Impact - Rubber', 'Lethal Round - Centrefire',
       'Lethal Round - Rimfire', 'Lethal Round - Shotgun',
       'Noise - Banger or Screamer', 'Noise - Blank', 'Noise - Clapping',
       'Noise - Horn', 'Noise - Siren', 'Noise - Voice', 'None',
       'Non-impact - Chalkball', 'Non-impact - Projectile',
       'Not Applicable', 'Other', 'Presence of Officer/Person',
       'Presence of Vehicle', 'Unknown', 'Visual - Flagging or stick'],
      dtype=object)
```

In [106…
```
Animals["Deterrents Used"][Animals["Deterrents Used"]== "Impact - Electric Shock"].count
```

Out[106]:
```
4
```

In [107…
```
#There are only 4 occurences of "Impact - Electric Shock" and it is not valid per the di

Animals["Deterrents Used"] = Animals["Deterrents Used"].replace({"Impact - Electric Shoc

#The only other values not in dictionary are missing and not to be replaced at this time
```

In [108…
```
#Checking to see how many values do not match values in dictionary.
Animals["Animal Response to Deterrents"].isin(DataDictionary["Data_Value"][DataDictionar
```

Out[108]:
```
10502
```

In [109…
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Animals.shape[0] - Animals["Animal Response to Deterrents"].isin(DataDictionary["Data_Va
#There are lots of entries that do not match values found in dictionary. Next lines of c
```

Out[109]:
```
63153
```

In [110…
```
#Add column to dataframe that indicates which values match dictionary (True) and which d
Animals["Animal Response to Deterrents_Dict"] = Animals["Animal Response to Deterrents"]
Animals.head()
```

Out[110]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Species Common Name	Sum of Number of Animals	Animal Health Status	Cause of Animal Health Status
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Coyote	2	Healthy	NaN
1	BAN2010-0003	2010-01-01	Banff Field	Banff National	Human Wildlife Interaction	Elk	1	Dead	Predation

			Unit	Park of Canada					
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Wolf	3	Not Located	NaN
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Found Wildlife	White-tailed Deer	1	Dead	Collision
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Attractant	None	0	NaN	NaN

5 rows × 22 columns

In [111…
```
#Print values that do not match dictionary to see which need to be replaced.
Animals["Animal Response to Deterrents"][Animals["Animal Response to Deterrents_Dict"]==

#all values that are not in the dictionary are missing values and not to be replaced at
```

Out[111]:
```
array([nan], dtype=object)
```

In [112…
```
#Data cleaning/validation complete for Animals dataset.

#Drop the columns I added during cleaning that are no longer needed
Animals = Animals.drop(["Species Common Name_Dict", "Animal Health Status_Dict", "Cause
Animals.head()
```

Out[112]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Species Common Name	Sum of Number of Animals	Animal Health Status	Cause of Animal Health Status
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Coyote	2	Healthy	NaN
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Elk	1	Dead	Predation
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	Wolf	3	Not Located	NaN
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Found Wildlife	White-tailed Deer	1	Dead	Collision
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Attractant	None	0	NaN	NaN

In [113…
```
###
#Next, we're looking for duplicate occurances of the incident number to ensure our final
```

In [114…
```
Animals_subset = Animals[["Incident Number", "Incident Date", "Field Unit", "Protected H
#Duplicate subset includes Incident Type attribute. This is what leads to me to dig deep
duplicate_Animals_subset = Animals_subset.duplicated(keep=False)
sum(duplicate_Animals_subset)
```

Out[114]:
```
18314
```

In [115…
```
duplicate_Animals_Inc_Num = Animals.duplicated(subset="Incident Number", keep=False)
sum(duplicate_Animals_Inc_Num)
```

Out[115]:
```
18314
```

In [116…
```
sum(duplicate_Animals_Inc_Num)==sum(duplicate_Animals_subset)
```

Out[116]:
```
True
```

In [117…
```
#There are several duplicates of incident numbers here. There are also several attribute
#I will add a new column to this dataset that combines the Incident Number with the "Spe
#I will join the other 3 datasets to this dataset using the Incident Number (and any oth
#So the each occurence of the incident number in the Animal dataset will have the same i
```

In [118…
```
Animals3 = Animals
```

In [119…
```
Animals3.insert(0, "Duplicate Inc_Num", Animals3.duplicated(subset="Incident Number", ke
```

In [120…
```
ValueCounts = Animals3["Incident Number"].value_counts()
```

In [121…
```
ValueCounts["BAN2013-1151"]
```

Out[121]:
```
11
```

In [122…
```
Counts = []
for i in Animals3["Incident Number"]:
                Counts.append(ValueCounts[i])

Animals3.insert(0, "Duplicate Counts", Counts)

UniqueCounts = []
for i in Animals3["Incident Number"]:
                if ValueCounts[i] >= 1:
                    UniqueCounts.append(ValueCounts[i])
                    ValueCounts[i] -= 1
```

In [123…
```
Animals3.insert(0, "Unique Counts", UniqueCounts)
```

In [124…
```
#Need to convert "Unique Counts" to string type (from integer type) before i'm able to j

Animals3["Unique Counts"]= Animals3["Unique Counts"].astype(str)
```

In [125…
```
Animals3.insert(0, "UniqueID", Animals3[["Incident Number", "Unique Counts"]].apply(".".
```

In [126…
```
#Checking to ensure there are no duplicates in the the UniqueID
duplicates_UniqueID = Animals3.duplicated(subset="UniqueID", keep=False)
sum(duplicates_UniqueID)
```

Out[126]:
```
0
```

In [127…
```
Animals3
```

Out[127]:

	UniqueID	Unique Counts	Duplicate Counts	Duplicate Inc_Num	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Inci
0	BAN2010-0003.3	3	3	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife
1	BAN2010-0003.2	2	3	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife
2	BAN2010-0003.1	1	3	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife
3	JNP2010-0011.1	1	1	False	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recove
4	JNP2010-0015.1	1	1	False	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	
...	...	...	...	...	...	...	...	...	
73650	2021-HWC-0574-JASFU-0016.2	2	2	True	2021-HWC-0574-JASFU-0016	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife
73651	2021-HWC-0574-JASFU-0016.1	1	2	True	2021-HWC-0574-JASFU-0016	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife
73652	2021-HWC-1114-YKLLFU-0033.1	1	1	False	2021-HWC-1114-YKLLFU-0033	2021-12-31	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	
73653	2022-HWC-0574-JASFU-0001.2	2	2	True	2022-HWC-0574-JASFU-0001	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife
73654	2022-HWC-0574-JASFU-0001.1	1	2	True	2022-HWC-0574-JASFU-0001	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife

73655 rows × 18 columns

In [ ]:

In [128…

```
Incidents = pd.read_csv("/Users/nerdbear/Downloads/5. pca-human-wildlife-coexistence-inc
```

In [129…
```
Incidents.head()
```

Out[129]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Latitude Public	Longitude Public	Within Park	Incident Type	T S Invol
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	51.161093	-115.593386	Yes	Human Wildlife Interaction	
1	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	53.139120	-117.964219	Yes	Rescued/Recovered/Found Wildlife	
2	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	53.050492	-118.073612	Yes	Attractant	
3	JNP2010-0023	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	52.858415	-118.102814	Yes	Rescued/Recovered/Found Wildlife	
4	JNP2010-0016	2010-01-02	Jasper Field Unit	Jasper National Park of Canada	52.857314	-118.103110	Yes	Rescued/Recovered/Found Wildlife	

In [130…
```
Incidents.shape
```

Out[130]:
```
(64290, 10)
```

In [131…
```
Incidents.dtypes
```

Out[131]:
```
Incident Number             object
Incident Date               object
Field Unit                  object
Protected Heritage Area     object
Latitude Public            float64
Longitude Public           float64
Within Park                 object
Incident Type               object
Total Staff Involved       float64
Total Staff Hours          float64
dtype: object
```

In [132…
```
#First we are going to clean the data to ensure valid entries on the string values by co
```

In [133…
```
#Checking to see how many values do not match values in dictionary.
Incidents["Field Unit"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Field"]==
```

Out[133]:
```
64290
```

In [134…
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Incidents.shape[0] - Incidents["Field Unit"].isin(DataDictionary["Data_Value"][DataDicti
#There are no entries that do not match values found in dictionary. No replacements need
```

Out[134]:
```
0
```

In [135…
```
#Checking to see how many values do not match values in dictionary.
Incidents["Protected Heritage Area"].isin(DataDictionary["Data_Value"][DataDictionary["D
```

Out[135]: 64290

In [136…
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Incidents.shape[0] - Incidents["Protected Heritage Area"].isin(DataDictionary["Data_Valu
#There are no entries that do not match values found in dictionary. No replacements need
```

Out[136]: 0

In [137…
```
#Checking to see how many values do not match values in dictionary.
Incidents["Incident Type"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Field"
```

Out[137]: 64258

In [138…
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Incidents.shape[0] - Incidents["Incident Type"].isin(DataDictionary["Data_Value"][DataDi
#There are no entries that do not match values found in dictionary. No replacements need
```

Out[138]: 32

In [139…
```
#Add column to dataframe that indicates which values match dictionary (True) and which d
Incidents["Incident Type_Dict"] = Incidents["Incident Type"].isin(DataDictionary["Data_V
Incidents.head()
```

Out[139]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Latitude Public	Longitude Public	Within Park	Incident Type	T S Invol
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	51.161093	-115.593386	Yes	Human Wildlife Interaction	
1	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	53.139120	-117.964219	Yes	Rescued/Recovered/Found Wildlife	
2	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	53.050492	-118.073612	Yes	Attractant	
3	JNP2010-0023	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	52.858415	-118.102814	Yes	Rescued/Recovered/Found Wildlife	
4	JNP2010-0016	2010-01-02	Jasper Field Unit	Jasper National Park of Canada	52.857314	-118.103110	Yes	Rescued/Recovered/Found Wildlife	

In [140…
```
#Print values that do not match dictionary to see which need to be replaced.
Incidents["Incident Type"][Incidents["Incident Type_Dict"]== False].unique()
#All values that are not in dictionary are missing values and will not be replaced now (
```

Out[140]: array([nan], dtype=object)

In [141…
```
#Drop the columns I added during cleaning that are no longer needed
Incidents = Incidents.drop(["Incident Type_Dict"], axis=1)
Incidents.head()
```

Out[141]:

	Incident Number	Incident Date	Field Unit	Protected Heritage	Latitude Public	Longitude Public	Within Park	Incident Type	T S

				Area				Invol
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	51.161093	-115.593386	Yes	Human Wildlife Interaction
1	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	53.139120	-117.964219	Yes	Rescued/Recovered/Found Wildlife
2	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	53.050492	-118.073612	Yes	Attractant
3	JNP2010-0023	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	52.858415	-118.102814	Yes	Rescued/Recovered/Found Wildlife
4	JNP2010-0016	2010-01-02	Jasper Field Unit	Jasper National Park of Canada	52.857314	-118.103110	Yes	Rescued/Recovered/Found Wildlife

In [ ]:

In [142…
```
###
#Next, we're looking for duplicate occurances of the incident number to ensure our final
```

In [143…
```
Inc_subset = Incidents[["Incident Number", "Incident Date", "Field Unit", "Protected Her
duplicate_Inc_subset = Inc_subset.duplicated(keep=False)
sum(duplicate_Inc_subset)
```

Out[143]:
```
64
```

In [144…
```
duplicate_Inc_Inc_Num = Incidents.duplicated(subset="Incident Number", keep=False)
sum(duplicate_Inc_Inc_Num)
```

Out[144]:
```
64
```

In [145…
```
sum(duplicate_Inc_Inc_Num)==sum(duplicate_Inc_subset)
##Conclusiong, The "Incident Type" and "Total Staff Involved" and "Total Staff Hours" at
#Will further investigate duplicates:
```

Out[145]:
```
True
```

In [146…
```
dup_bool = Incidents["Incident Number"].duplicated(keep=False)
print (dup_bool)
Dup_Incidents = Incidents[dup_bool]
Dup_Incidents
Dup_Incidents["Incident Type"].isna().sum()
```

```
0         False
1         False
2         False
3         False
4         False
          ...
64285     False
64286     False
64287     False
64288     False
```

```
64289    False
Name: Incident Number, Length: 64290, dtype: bool
```

Out[146]:
```
32
```

In [147…
```
Incidents[Incidents["Incident Type"].isna()]
Incidents["Incident Type"].isna().sum()
```

Out[147]:
```
32
```

In [148…
```
Incidents["Incident Type"].isna().sum() == Dup_Incidents["Incident Type"].isna().sum()
```

Out[148]:
```
True
```

In [149…
```
#Number of NA's in "Incident Type" column is the same in the entire dataset as it is in
#There are only 32 Incident Types that are NaN and they are the 32 Incidents Types that

#There are a total of 64 duplicate rows and of the 64 duplicate rows, there are 32 missi
#If "Incident Type" is Nan AND Incident type is duplicate, I will delete that row. The n

#There are only 32 Incident Types that are NaN and they are the 32 Incidents Types that
#Delete these rows
```

In [150…
```
Incidents2 = Incidents[Incidents["Incident Type"].notnull()]
```

In [151…
```
#Checking to confirm there are no duplicates remaining:
#Looking for duplicates in subset
Inc_subset = Incidents2[["Incident Number", "Incident Date", "Field Unit", "Protected He
```

In [152…
```
duplicate_Inc_subset = Inc_subset.duplicated(keep=False)
sum(duplicate_Inc_subset)
```

Out[152]:
```
0
```

In [153…
```
#Looking for duplicates in just Incident Number column.
duplicate_Inc_Inc_Num = Incidents2.duplicated(subset="Incident Number", keep=False)
sum(duplicate_Inc_Inc_Num)
```

Out[153]:
```
0
```

In [154…
```
#Comparing the two
sum(duplicate_Inc_Inc_Num)==sum(duplicate_Inc_subset)
#Conclusion, there are no duplicate Incident Numbers remaining.
```

Out[154]:
```
True
```

In [155…
```
#Confirming there are no NA values remaining in "Incident Type" column of new dataframe:
Incidents2["Incident Type"].isna().sum()
#Conclusion, no missing values remaining in new Incidents2 dataset. Will use this datase
```

Out[155]:
```
0
```

In [156…
```
Incidents2
```

Out[156]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Latitude Public	Longitude Public	Within Park	Incident Typ
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	51.161093	-115.593386	Yes	Human Wildlife Interactio

1	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	53.139120	-117.964219	Yes	Rescued/Recovered/Foun Wildlif
2	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	53.050492	-118.073612	Yes	Attractar
3	JNP2010-0023	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	52.858415	-118.102814	Yes	Rescued/Recovered/Foun Wildlif
4	JNP2010-0016	2010-01-02	Jasper Field Unit	Jasper National Park of Canada	52.857314	-118.103110	Yes	Rescued/Recovered/Foun Wildlif
...	...	...	...	...	...	...	...	.
64285	2021-HWC-0000-JASFU-2861	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	52.876739	-118.091588	Yes	Human Wildlife Interactio
64286	2021-HWC-0000-JASFU-2862	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	53.093617	-118.030592	Yes	Rescued/Recovered/Foun Wildlif
64287	2021-HWC-0574-JASFU-0016	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	52.860896	-118.087098	Yes	Human Wildlife Interactio
64288	2021-HWC-1114-YKLLFU-0033	2021-12-31	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	51.380551	-116.147884	Yes	Attractar
64289	2022-HWC-0574-JASFU-0001	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	53.162687	-117.964186	Yes	Human Wildlife Interactio

64258 rows × 10 columns

In [157…
```
#Cross checking to ensure correct number of rows remain.
#Number of rows in Original Dataset, minus number of NA values in duplicates (32) == Num
Incidents.shape[0] - 32 == Incidents2.shape[0]
```

Out[157]:
```
True
```

In [158…
```
Responses = pd.read_csv("/Users/nerdbear/Downloads/6. pca-human-wildlife-coexistence-res
```

In [159…
```
Responses.head()
```

Out[159]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Response Type
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Dispose Carcass

1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Investigate Incident
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Monitor - patrol
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Dispose Carcass
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Dispose Carcass

In [160…
```
Responses.shape
```

Out[160]:
```
(82109, 5)
```

In [161…
```
Responses.dtypes
```

Out[161]:
```
Incident Number            object
Incident Date              object
Field Unit                 object
Protected Heritage Area    object
Response Type              object
dtype: object
```

In [162…
```
#First we are going to clean the data to ensure valid entries on the string values by co
```

In [163…
```
#Checking to see how many values do not match values in dictionary.
Responses["Field Unit"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Field"]==
```

Out[163]:
```
82109
```

In [164…
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Responses.shape[0] - Responses["Field Unit"].isin(DataDictionary["Data_Value"][DataDicti
#There are no entries that do not match values found in dictionary. No replacements need
```

Out[164]:
```
0
```

In [165…
```
#Checking to see how many values do not match values in dictionary.
Responses["Protected Heritage Area"].isin(DataDictionary["Data_Value"][DataDictionary["D
```

Out[165]:
```
82109
```

In [166…
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Responses.shape[0] - Responses["Protected Heritage Area"].isin(DataDictionary["Data_Valu
#There are no entries that do not match values found in dictionary. No replacements need
```

Out[166]:
```
0
```

In [167…
```
#Checking to see how many values do not match values in dictionary.
Responses["Response Type"].isin(DataDictionary["Data_Value"][DataDictionary["Data_Field"
```

Out[167]:
```
79747
```

In [168…
```
#Shows how many are False, therefore how many activity types are not in the dictionary.
Responses.shape[0] - Responses["Response Type"].isin(DataDictionary["Data_Value"][DataDi
#There are no entries that do not match values found in dictionary. No replacements need
```

Out[168]:
```
2362
```

In [169…
```
#Add column to dataframe that indicates which values match dictionary (True) and which d
Responses["Response Type_Dict"] = Responses["Response Type"].isin(DataDictionary["Data_V
Responses.head()
```

Out[169]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Response Type	Response Type_Dict
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Dispose Carcass	True
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Investigate Incident	True
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Monitor - patrol	True
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Dispose Carcass	True
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Dispose Carcass	True

In [170…
```
#Print values that do not match dictionary to see which need to be replaced.
Responses["Response Type"][Responses["Response Type_Dict"]== False].unique()
```

Out[170]:
```
array([nan, 'Monitor'], dtype=object)
```

In [171…
```
#printing all activity types from dictionary to see which best match the errors listed a
DataDictionary["Data_Value"][DataDictionary["Data_Field"]== "Response Type"].unique()
```

Out[171]:
```
array(['Assist other Agency', 'Assist other Field Unit', 'Assist Visitor',
       'Attractant Management', 'Aversive Conditioning', 'Cancel Permit',
       'Capture and transport to captivity', 'Clean Up', 'Close Area',
       'Close Road', 'Collar', 'Collect Sample', 'Cull', 'Destroy Animal',
       'Disentangle', 'Dispatch other Agency', 'Disperse Wildlife Jam',
       'Dispose Carcass', 'Ear Tag', 'Euthanize', 'Evacuate Visitor',
       'Haze - Hard', 'Haze - Soft', 'Immobilize Animal',
       'Inform Visitor', 'Infrastructure modification',
       'Investigate Incident', 'Issue Prohibited Activity Order',
       'Issue Restricted Activity Order', 'Issue Stop Work Order',
       'Leave on Landscape', 'Mark - microchip', 'Mark - paint',
       'Monitor - Camera', 'Monitor - patrol',
       'Monitor - visitor and staff sighting', 'Necropsy',
       'No response required', 'Not Applicable',
       'Refer incident to other agency', 'Rehabilitate area',
       'Relocate animal (s)', 'Request assistance - other Agency',
       'Request assistance - police', 'Traffic control', 'Translocate',
       'Trap or snare', 'Unable to respond', 'Warning signs'],
      dtype=object)
```

In [172…
```
Responses["Response Type"][Responses["Response Type"]== "Monitor"].count()
#There are only 2 occurences of the invalid "monitor" value
```

Out[172]:
```
2
```

In [173…
```
Responses.loc[Responses["Response Type"]== "Monitor"]
```

Out[173]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Response Type	Response Type_Dict
51090	2018-HWC-0177-ENFU-0004	2018-09-10	Newfoundland East Field Unit	Terra Nova National Park of Canada	Monitor	False
52549	2019-HWC-0144-NPRFU-0001	2019-01-08	Northern Prairies Field Unit	Prince Albert National Park of Canada	Monitor	False

In [174…
```
#No way to know which of the 3 valid "Monitor" options value was intended here so replac
#Replacing values that were mis-entered with their proper type, if none was obvious from
```

```
Responses["Response Type"] = Responses["Response Type"].replace({"Monitor": ""})
#All other values that don't match dictionary are missing values, will not replace at th
```

In [175…
```
#Drop the columns I added during cleaning that are no longer needed
Responses = Responses.drop(["Response Type_Dict"], axis=1)
Responses.head()
```

Out[175]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Response Type
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Dispose Carcass
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Investigate Incident
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Monitor - patrol
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Dispose Carcass
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Dispose Carcass

In [ ]:
```
```

In [176…
```
###
#Next, we're looking for duplicate occurances of the incident number to ensure our final
```

In [177…
```
Resp_subset = Responses[["Incident Number", "Incident Date", "Field Unit", "Protected He
duplicate_Resp_subset = Resp_subset.duplicated(keep=False)
sum(duplicate_Resp_subset)
```

Out[177]:
```
32243
```

In [178…
```
duplicate_Resp_Inc_Num = Responses.duplicated(subset="Incident Number", keep=False)
sum(duplicate_Resp_Inc_Num)
```

Out[178]:
```
32243
```

In [179…
```
sum(duplicate_Resp_Inc_Num)==sum(duplicate_Resp_subset)
#Where the Incident Number is duplicated, all column values are duplicated except for th
```

Out[179]:
```
True
```

In [180…
```
#Finding unique Response Types. *** Emailed David Gummer about whether there is a refere
Responses["Response Type"].unique()
```

Out[180]:
```
array(['Dispose Carcass', 'Investigate Incident', 'Monitor - patrol', nan,
       'Inform Visitor', 'Destroy Animal', 'Request assistance - police',
       'Relocate animal (s)', 'Trap or snare', 'Necropsy',
       'Refer incident to other agency', 'Haze - Soft', 'Clean Up',
       'Traffic control', 'Dispatch other Agency',
       'Issue Restricted Activity Order', 'Close Area', 'Not Applicable',
       'Request assistance - other Agency', 'Immobilize Animal',
       'Leave on Landscape', 'Warning signs', 'Assist other Agency',
       'Collect Sample', 'Assist Visitor', 'No response required',
       'Haze - Hard', 'Capture and transport to captivity', 'Ear Tag',
       'Disperse Wildlife Jam', 'Evacuate Visitor',
       'Aversive Conditioning', 'Close Road',
       'Issue Prohibited Activity Order', 'Euthanize',
       'Infrastructure modification', 'Disentangle',
       'Monitor - visitor and staff sighting', 'Assist other Field Unit',
       'Cull', 'Monitor - Camera', 'Attractant Management', 'Collar',
       'Unable to respond', 'Issue Stop Work Order', 'Translocate',
       'Mark - paint', 'Rehabilitate area', ''], dtype=object)
```

In [181…
```
#Checking how many of the duplicates have NA values in "Response Type"
```

```
dup_Responses = Responses[duplicate_Resp_subset]
dup_Responses
```

Out[181]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Response Type
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Dispose Carcass
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Investigate Incident
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Monitor - patrol
8	PRN2010-0001	2010-01-02	Coastal British Columbia Field Unit	Pacific Rim National Park Reserve of Canada	NaN
9	PRN2010-0001	2010-01-02	Coastal British Columbia Field Unit	Pacific Rim National Park Reserve of Canada	Investigate Incident
...	...	...	...	...	...
82075	2021-HWC-1075-CBCFU-0051	2021-12-20	Coastal British Columbia Field Unit	Pacific Rim National Park Reserve of Canada	Clean Up
82076	2021-HWC-1075-CBCFU-0051	2021-12-20	Coastal British Columbia Field Unit	Pacific Rim National Park Reserve of Canada	Monitor - patrol
82077	2021-HWC-1075-CBCFU-0051	2021-12-20	Coastal British Columbia Field Unit	Pacific Rim National Park Reserve of Canada	Monitor - visitor and staff sighting
82083	2021-HWC-1075-CBCFU-0052	2021-12-21	Coastal British Columbia Field Unit	Pacific Rim National Park Reserve of Canada	Monitor - patrol
82084	2021-HWC-1075-CBCFU-0052	2021-12-21	Coastal British Columbia Field Unit	Pacific Rim National Park Reserve of Canada	Monitor - visitor and staff sighting

32243 rows × 5 columns

In [182…

```
#Checking how many of the duplicates have NA values in "Activity Type"
dup_Responses["Response Type"].isna().sum()
```

Out[182]:

```
583
```

In [183…

```
#Count number of unique Incident Numbers in duplicates.
dup_Responses["Incident Number"].nunique()
```

Out[183]:

```
12930
```

In [184…

```
#I would like to encode Response Type so each distinct Response type is it's own column

#Count distinct values in Response Type
Responses["Response Type"].nunique()

encoder = OneHotEncoder(handle_unknown='ignore')
encoder_df = pd.DataFrame(encoder.fit_transform(Responses[["Response Type"]]).toarray())
encoder_df.columns = encoder.get_feature_names_out(["Response Type"])
encoder_df.head()
Responses_encoded = Responses.join(encoder_df)
Responses_encoded.head()
Responses_encoded.drop('Response Type', axis = 1, inplace=True)
Responses_encoded.head()
```

Out[184]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Response Type_	Response Type_Assist Visitor	Response Type_Assist	Response Type_Assist	Resp Type_Attrac Manage

							other Agency	other Field Unit
0	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0
1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0
2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0
3	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	0.0	0.0	0.0	0.0
4	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	0.0	0.0	0.0	0.0

5 rows × 53 columns

In [ ]:

In [217…
```
#Viewing sums of each response type column.
print(Responses_encoded[Responses_encoded.columns[4:53]].sum())
```

```
Response Type_                                    2.0
Response Type_Assist Visitor                    238.0
Response Type_Assist other Agency               291.0
Response Type_Assist other Field Unit             9.0
Response Type_Attractant Management             373.0
Response Type_Aversive Conditioning             134.0
Response Type_Capture and transport to captivity  66.0
Response Type_Clean Up                         1006.0
Response Type_Close Area                        860.0
Response Type_Close Road                        197.0
Response Type_Collar                             33.0
Response Type_Collect Sample                    578.0
Response Type_Cull                              179.0
Response Type_Destroy Animal                    720.0
Response Type_Disentangle                       128.0
Response Type_Dispatch other Agency             138.0
Response Type_Disperse Wildlife Jam            2880.0
Response Type_Dispose Carcass                  4550.0
Response Type_Ear Tag                           141.0
Response Type_Euthanize                         312.0
Response Type_Evacuate Visitor                  111.0
Response Type_Haze - Hard                      2032.0
Response Type_Haze - Soft                     18957.0
Response Type_Immobilize Animal                 103.0
Response Type_Inform Visitor                   2796.0
Response Type_Infrastructure modification       272.0
Response Type_Investigate Incident             18545.0
Response Type_Issue Prohibited Activity Order     31.0
Response Type_Issue Restricted Activity Order     82.0
Response Type_Issue Stop Work Order                3.0
Response Type_Leave on Landscape                 880.0
Response Type_Mark - paint                        76.0
```

```
Response Type_Monitor - Camera                          324.0
Response Type_Monitor - patrol                         9740.0
Response Type_Monitor - visitor and staff sighting     3420.0
Response Type_Necropsy                                  400.0
Response Type_No response required                      266.0
Response Type_Not Applicable                            313.0
Response Type_Refer incident to other agency            556.0
Response Type_Rehabilitate area                          26.0
Response Type_Relocate animal (s)                      2293.0
Response Type_Request assistance - other Agency         256.0
Response Type_Request assistance - police               194.0
Response Type_Traffic control                          1909.0
Response Type_Translocate                                34.0
Response Type_Trap or snare                            1482.0
Response Type_Unable to respond                         501.0
Response Type_Warning signs                            1312.0
Response Type_nan                                      2360.0
dtype: float64
```

In [186…

```
##** I would like to merge all columns relating to Reponse Types (columns 4-53) across d
Responses2 = Responses_encoded[Responses_encoded.columns[4:53]].groupby([Responses['Inci
Responses2
```

Out[186]:

	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Response Type_	Response Type_Assist Visitor	Response Type_Assist other Agency	Response Type_Assist other Field Unit	Type M
0	2017-HWC-0005-YKLLFU-0001	2017-08-01	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0	
1	2017-HWC-0005-YKLLFU-0002	2017-09-07	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0	
2	2017-HWC-0005-YKLLFU-0003	2017-07-08	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0	
3	2017-HWC-0005-YKLLFU-0004	2017-06-23	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0	
4	2017-HWC-0005-YKLLFU-0006	2017-06-28	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	0.0	0.0	0.0	0.0	
…	…	…	…	…	…	…	…	…	
62791	YNP2016-0146	2016-10-28	Lake Louise, Yoho and Kootenay	Yoho National Park of Canada	0.0	0.0	0.0	0.0	

			Field Unit					
62792	YNP2016-0147	2016-10-30	Lake Louise, Yoho and Kootenay Field Unit	Yoho National Park of Canada	0.0	0.0	0.0	0.0
62793	YNP2016-0148	2016-11-22	Lake Louise, Yoho and Kootenay Field Unit	Yoho National Park of Canada	0.0	0.0	0.0	0.0
62794	YNP2016-0151	2016-12-27	Lake Louise, Yoho and Kootenay Field Unit	Yoho National Park of Canada	0.0	0.0	0.0	0.0
62795	ynp2014-0137	2014-06-23	Lake Louise, Yoho and Kootenay Field Unit	Yoho National Park of Canada	0.0	0.0	0.0	0.0

62796 rows × 53 columns

In [218…
```
Responses2.isna().sum()
```

Out[218]:
```
Incident Number                                    0
Incident Date                                      0
Field Unit                                         0
Protected Heritage Area                            0
Response Type_                                     0
Response Type_Assist Visitor                       0
Response Type_Assist other Agency                  0
Response Type_Assist other Field Unit              0
Response Type_Attractant Management                0
Response Type_Aversive Conditioning                0
Response Type_Capture and transport to captivity   0
Response Type_Clean Up                             0
Response Type_Close Area                           0
Response Type_Close Road                           0
Response Type_Collar                               0
Response Type_Collect Sample                       0
Response Type_Cull                                 0
Response Type_Destroy Animal                       0
Response Type_Disentangle                          0
Response Type_Dispatch other Agency                0
Response Type_Disperse Wildlife Jam                0
Response Type_Dispose Carcass                      0
Response Type_Ear Tag                              0
Response Type_Euthanize                            0
Response Type_Evacuate Visitor                     0
Response Type_Haze - Hard                          0
Response Type_Haze - Soft                          0
Response Type_Immobilize Animal                    0
Response Type_Inform Visitor                       0
Response Type_Infrastructure modification          0
Response Type_Investigate Incident                 0
Response Type_Issue Prohibited Activity Order      0
```

```
Response Type_Issue Restricted Activity Order         0
Response Type_Issue Stop Work Order                   0
Response Type_Leave on Landscape                      0
Response Type_Mark - paint                            0
Response Type_Monitor - Camera                        0
Response Type_Monitor - patrol                        0
Response Type_Monitor - visitor and staff sighting    0
Response Type_Necropsy                                0
Response Type_No response required                    0
Response Type_Not Applicable                          0
Response Type_Refer incident to other agency          0
Response Type_Rehabilitate area                       0
Response Type_Relocate animal (s)                     0
Response Type_Request assistance - other Agency       0
Response Type_Request assistance - police             0
Response Type_Traffic control                         0
Response Type_Translocate                             0
Response Type_Trap or snare                           0
Response Type_Unable to respond                       0
Response Type_Warning signs                           0
Response Type_nan                                     0
dtype: int64
```

In [187…
```
#Confirming whether the new dataset has any duplicate incident numbers
duplicate_Resp2_Inc_Num = Responses2.duplicated(subset="Incident Number", keep=False)
sum(duplicate_Resp2_Inc_Num)
```

Out[187]:
```
0
```

In [188…
```
#Cross checking to ensure correct number of rows remain.
#Number of rows in Original Dataset, minus (number of rows in duplicates subset minus nu
Responses.shape[0] - (dup_Responses.shape[0] - dup_Responses["Incident Number"].nunique(
```

Out[188]:
```
True
```

In [189…
```
#(In other words, I want to ensure that our new dataset has the same number of Unique in
Responses["Incident Number"].nunique() == Responses2["Incident Number"].nunique()

#Conclusion, correct number of rows are remaining in our new dataset.
```

Out[189]:
```
True
```

In [190…
```
#Joining datasets without losing any rows from any dataset.

#Checking all 4 datasets and comparing Incident Numbers. Because we'll be using Animals
#I Want to see if there are any incident numbers included in the other 3 datasets that a
#Conclusion based on results below, there are three (3) incident numbers included in oth

AnimalIDs = Animals3["Incident Number"].unique()
AnimalIDs
AnimalIDs = np.sort(AnimalIDs)
AnimalIDs
AnimalIDs.size
ActivityIDs = Activities2["Incident Number"]
ActivityIDs
ActivityIDs = np.sort(ActivityIDs)
ActivityIDs
ActivityIDs.size
dif1 = list(set(ActivityIDs)-set(AnimalIDs))
dif1
IncidentIDs = Incidents2["Incident Number"]
IncidentIDs.size
IncidentIDs = np.sort(IncidentIDs)
IncidentIDs
```

```
dif2 = list(set(IncidentIDs)-set(AnimalIDs))
dif2
ResponseIDs = Responses2["Incident Number"]
ResponseIDs.size
ResponseIDs = np.sort(ResponseIDs)
ResponseIDs
dif3 = list(set(ResponseIDs)-set(AnimalIDs))
dif3
print(dif1, dif2, dif3)
```

```
['PEINP2011-0131', '2021-VS-0748-YKLLFU-0001'] ['PEINP2011-0131', '2019-HWC-0000-BANFU-1
457', '2021-VS-0748-YKLLFU-0001'] ['PEINP2011-0131']
```

In [191…

```
Animals.head()
```

Out[191]:

	UniqueID	Unique Counts	Duplicate Counts	Duplicate Inc_Num	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Ty
0	BAN2010-0003.3	3	3	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interacti
1	BAN2010-0003.2	2	3	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interacti
2	BAN2010-0003.1	1	3	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interacti
3	JNP2010-0011.1	1	1	False	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Fou Wildl
4	JNP2010-0015.1	1	1	False	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Attracta

In [ ]:

In [192…

```
#Now joining datasets together.
#Doing Outer Joins to ensure no loss of data at this stage for Incident Numbers that exi

JoinedData1 = pd.merge(Animals3, Activities2, how="outer", on = ["Incident Number", "Inc
JoinedData1
```

Out[192]:

	UniqueID	Unique Counts	Duplicate Counts	Duplicate Inc_Num	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Inc
0	BAN2010-0003.3	3	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife
1	BAN2010-0003.2	2	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife

2	BAN2010-0003.1	1	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife
3	JNP2010-0011.1	1	1.0	False	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recov
4	JNP2010-0015.1	1	1.0	False	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	
...	...	...	...	...	...	...	...	...	
73652	2021-HWC-1114-YKLLFU-0033.1	1	1.0	False	2021-HWC-1114-YKLLFU-0033	2021-12-31	Lake Louise, Yoho and Kootenay Field Unit	Banff National Park of Canada	
73653	2022-HWC-0574-JASFU-0001.2	2	2.0	True	2022-HWC-0574-JASFU-0001	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife
73654	2022-HWC-0574-JASFU-0001.1	1	2.0	True	2022-HWC-0574-JASFU-0001	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife
73655	NaN	NaN	NaN	NaN	2021-VS-0748-YKLLFU-0001	2021-06-19	Banff Field Unit	Banff National Park of Canada	
73656	NaN	NaN	NaN	NaN	PEINP2011-0131	2011-07-08	Prince Edward Island Field Unit	Prince Edward Island National Park of Canada	

73657 rows × 107 columns

In [193…

```
#Confirming that Incident Numbers contained in Activities but not in Animals dataset wer
JoinedData1.loc[JoinedData1["Incident Number"].isin(dif1)]
```

Out[193]:

	UniqueID	Unique Counts	Duplicate Counts	Duplicate Inc_Num	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Spec Comm Na
73655	NaN	NaN	NaN	NaN	2021-VS-0748-YKLLFU-0001	2021-06-19	Banff Field Unit	Banff National Park of Canada	NaN	N
73656	NaN	NaN	NaN	NaN	PEINP2011-0131	2011-07-08	Prince Edward Island Field Unit	Prince Edward Island National Park of Canada	NaN	N

2 rows × 107 columns

In [194…
```
JoinedData2 = pd.merge(JoinedData1, Incidents2, how="outer", on = ["Incident Number", "I
JoinedData2
```

Out[194]:

	UniqueID	Unique Counts	Duplicate Counts	Duplicate Inc_Num	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incide
0	BAN2010-0003.3	3	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife I
1	BAN2010-0003.2	2	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife I
2	BAN2010-0003.1	1	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife I
3	JNP2010-0011.1	1	1.0	False	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recove
4	JNP2010-0015.1	1	1.0	False	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	
...	...	...	...	...	...	...	...	...	
73653	2022-HWC-0574-JASFU-0001.2	2	2.0	True	2022-HWC-0574-JASFU-0001	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife I
73654	2022-HWC-0574-JASFU-0001.1	1	2.0	True	2022-HWC-0574-JASFU-0001	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife I
73655	NaN	NaN	NaN	NaN	2021-VS-0748-YKLLFU-0001	2021-06-19	Banff Field Unit	Banff National Park of Canada	
73656	NaN	NaN	NaN	NaN	PEINP2011-0131	2011-07-08	Prince Edward Island Field Unit	Prince Edward Island National Park of Canada	
73657	NaN	NaN	NaN	NaN	2019-HWC-0000-BANFU-1457	2019-08-20	Banff Field Unit	Banff National Park of Canada	

73658 rows × 113 columns

In [195…
```
JoinedData2["Incident Type_x"] == JoinedData2["Incident Type_y"]
```

Out[195]:
```
0          True
1          True
2          True
3          True
4          True
          ...
73653      True
73654      True
73655     False
73656     False
73657     False
Length: 73658, dtype: bool
```

In [196…
```
#Both Animals3 and Incidents2 contained a column for "Incident Type" so joining the two
#Looking for differences between the two columns.
```

In [197…
```
difference = list(set(JoinedData2["Incident Type_x"]) - set(JoinedData2["Incident Type_y
```

In [198…
```
difference
```

Out[198]:
```
[nan]
```

In [199…
```
JoinedData2["Incident Type_x"].isna().sum()
```

Out[199]:
```
3
```

In [200…
```
JoinedData2["Incident Type_y"].isna().sum()
```

Out[200]:
```
0
```

In [201…
```
#Conclusion, Incident Type_x column contains 3 na values, whereas Incident Type_y contai
#Will drop "Incident Type_x".

JoinedData2.drop('Incident Type_x', axis = 1, inplace=True)
JoinedData2.head()
```

Out[201]:

	UniqueID	Unique Counts	Duplicate Counts	Duplicate Inc_Num	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Species Common Name	Sum of Number of Animals	..
0	BAN2010-0003.3	3	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Coyote	2.0	..
1	BAN2010-0003.2	2	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Elk	1.0	..
2	BAN2010-0003.1	1	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Wolf	3.0	..
3	JNP2010-0011.1	1	1.0	False	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	White-tailed Deer	1.0	..
4	JNP2010-0015.1	1	1.0	False	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National	None	0.0	..

Park of Canada

5 rows × 112 columns

In [202…
```
#Moving columns around so key information is closer to start of dataframe and all the ac
eight = JoinedData2.pop('Incident Type_y')
JoinedData2.insert(8, 'Incident Type', eight)
```

In [203…
```
nine = JoinedData2.pop('Latitude Public')
JoinedData2.insert(9, 'Latitude Public', nine)
```

In [204…
```
ten = JoinedData2.pop('Longitude Public')
JoinedData2.insert(10, 'Longitude Public', ten)
```

In [205…
```
eleven = JoinedData2.pop('Within Park')
JoinedData2.insert(11, 'Within Park', eleven)
```

In [206…
```
twelve = JoinedData2.pop('Total Staff Involved')
JoinedData2.insert(12, 'Total Staff Involved', twelve)
```

In [207…
```
thirteen = JoinedData2.pop('Total Staff Hours')
JoinedData2.insert(13, 'Total Staff Hours', thirteen)
```

In [208…
```
JoinedData2.head()
```

Out[208]:

	UniqueID	Unique Counts	Duplicate Counts	Duplicate Inc_Num	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Ty
0	BAN2010-0003.3	3	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interacti
1	BAN2010-0003.2	2	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interacti
2	BAN2010-0003.1	1	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interacti
3	JNP2010-0011.1	1	1.0	False	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Fou Wildl
4	JNP2010-0015.1	1	1.0	False	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Attracta

5 rows × 112 columns

In [209…
```
#Confirming that Incident Numbers contained in Incidents but not in Animals dataset were
JoinedData2.loc[JoinedData2["Incident Number"].isin(dif2)]
```

Out[209]:

	UniqueID	Unique Counts	Duplicate Counts	Duplicate Inc_Num	Incident Number	Incident Date	Field Unit	Protected Heritage	Incic

								Area	
73655	NaN	NaN	NaN	NaN	2021-VS-0748-YKLLFU-0001	2021-06-19	Banff Field Unit	Banff National Park of Canada	Highway Fence
73656	NaN	NaN	NaN	NaN	PEINP2011-0131	2011-07-08	Prince Edward Island Field Unit	Prince Edward Island National Park of Canada	Rescued/Recover
73657	NaN	NaN	NaN	NaN	2019-HWC-0000-BANFU-1457	2019-08-20	Banff Field Unit	Banff National Park of Canada	Human Wildlife In

3 rows × 112 columns

In [210…
```
JoinedData3 = pd.merge(JoinedData2, Responses2, how="outer", on = ["Incident Number", "I
JoinedData3
```

Out[210]:

	UniqueID	Unique Counts	Duplicate Counts	Duplicate Inc_Num	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Inci
0	BAN2010-0003.3	3	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife I
1	BAN2010-0003.2	2	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife I
2	BAN2010-0003.1	1	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife I
3	JNP2010-0011.1	1	1.0	False	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recove
4	JNP2010-0015.1	1	1.0	False	JNP2010-0015	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	
...	...	...	...	...	...	...	...	...	
73653	2022-HWC-0574-JASFU-0001.2	2	2.0	True	2022-HWC-0574-JASFU-0001	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife I
73654	2022-HWC-0574-JASFU-0001.1	1	2.0	True	2022-HWC-0574-JASFU-0001	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife I

73655	NaN	NaN	NaN	NaN	2021-VS-0748-YKLLFU-0001	2021-06-19	Banff Field Unit	Banff National Park of Canada	Highway Fenc
73656	NaN	NaN	NaN	NaN	PEINP2011-0131	2011-07-08	Prince Edward Island Field Unit	Prince Edward Island National Park of Canada	Rescued/Recove
73657	NaN	NaN	NaN	NaN	2019-HWC-0000-BANFU-1457	2019-08-20	Banff Field Unit	Banff National Park of Canada	Human Wildlife I

73658 rows × 161 columns

In [211…

```
#Confirming that Incident Numbers contained in Responses but not in Animals dataset were
JoinedData3.loc[JoinedData3["Incident Number"].isin(dif3)]
```

Out[211]:

	UniqueID	Unique Counts	Duplicate Counts	Duplicate Inc_Num	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incic
73656	NaN	NaN	NaN	NaN	PEINP2011-0131	2011-07-08	Prince Edward Island Field Unit	Prince Edward Island National Park of Canada	Rescued/Recover

1 rows × 161 columns

In [212…

```
#Renaming our final complete Dataset.
CompleteData = JoinedData3
CompleteData
```

Out[212]:

	UniqueID	Unique Counts	Duplicate Counts	Duplicate Inc_Num	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Inci
0	BAN2010-0003.3	3	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife I
1	BAN2010-0003.2	2	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife I
2	BAN2010-0003.1	1	3.0	True	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife I
3	JNP2010-0011.1	1	1.0	False	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recove
4	JNP2010-	1	1.0	False	JNP2010-	2010-	Jasper	Jasper	

	0015.1				0015	01-01	Field Unit	National Park of Canada	
...	...	...	...	...	...	...	...	...	
73653	2022-HWC-0574-JASFU-0001.2	2	2.0	True	2022-HWC-0574-JASFU-0001	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife I
73654	2022-HWC-0574-JASFU-0001.1	1	2.0	True	2022-HWC-0574-JASFU-0001	2021-12-31	Jasper Field Unit	Jasper National Park of Canada	Human Wildlife I
73655	NaN	NaN	NaN	NaN	2021-VS-0748-YKLLFU-0001	2021-06-19	Banff Field Unit	Banff National Park of Canada	Highway Fenc
73656	NaN	NaN	NaN	NaN	PEINP2011-0131	2011-07-08	Prince Edward Island Field Unit	Prince Edward Island National Park of Canada	Rescued/Recove
73657	NaN	NaN	NaN	NaN	2019-HWC-0000-BANFU-1457	2019-08-20	Banff Field Unit	Banff National Park of Canada	Human Wildlife I

73658 rows × 161 columns

In [213…

```
CompleteData.drop('Unique Counts', axis = 1, inplace=True)
CompleteData.drop('Duplicate Counts', axis = 1, inplace=True)
CompleteData.drop('Duplicate Inc_Num', axis = 1, inplace=True)
CompleteData.head()
```

Out[213]:

	UniqueID	Incident Number	Incident Date	Field Unit	Protected Heritage Area	Incident Type	Latitude Public	Longitude Public	W
0	BAN2010-0003.3	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	51.161093	-115.593386	
1	BAN2010-0003.2	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	51.161093	-115.593386	
2	BAN2010-0003.1	BAN2010-0003	2010-01-01	Banff Field Unit	Banff National Park of Canada	Human Wildlife Interaction	51.161093	-115.593386	
3	JNP2010-0011.1	JNP2010-0011	2010-01-01	Jasper Field Unit	Jasper National Park of Canada	Rescued/Recovered/Found Wildlife	53.139120	-117.964219	
4	JNP2010-0015.1	JNP2010-0015	2010-01-01	Jasper Field	Jasper National	Attractant	53.050492	-118.073612	

Unit Park of Canada

5 rows × 158 columns

In [214…
```
#assigning "UniqueID" to the rows that didn't already have one (i.e. the 3 rows that did
CompleteData['UniqueID'] = CompleteData['UniqueID'].fillna(CompleteData['Incident Number
```

In [224…
```
#assigning 0 values to the response type columns for all response types that were missin
CompleteData[CompleteData.columns[109:158]] = CompleteData[CompleteData.columns[109:158]
```

In [230…
```
CompleteData.to_csv("/Users/nerdbear/Downloads/Complete_HWC_Data.csv")
```