# This code file looks into Research Questions 1 & 2:

## Here we explore and analyze patterns in the data by plotting frequency histograms and time series plots looking at incident types by human activity, animals involved, location, and time of year.

*NOTE: The data variable named "Protected Heritage Area" refers to the 35 different **Canadian National Parks** included in the data. In all plot titles and labels, I refer to them as "Canadian National Parks" to assist in clarity as "Protected Heritage Area" can be a bit vague/unclear as to what it refers to.

**First, I import all the packages I'll be using throughout this file:**

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import *
from matplotlib.pyplot import figure
import numpy as np
import matplotlib.colors as mcolors
import calendar
from datetime import datetime
```

In [2]:
```
#Used to view long lists of output if needed
#pd.set_option('display.height', 1000)
#pd.set_option('display.max_rows', 500)
#pd.set_option('display.max_columns', 500)
#pd.set_option('display.width', 1000)
```

**Next, I import my dataset and set the datatypes as appropriate:**

In [3]:
```
Complete_HWC_Data = pd.read_csv("/Users/nerdbear/Downloads/Complete_HWC_Data.csv", index
Complete_HWC_Data[Complete_HWC_Data.columns[0:20]] = Complete_HWC_Data[Complete_HWC_Data

Complete_HWC_Data["Sum of Number of Animals"] = Complete_HWC_Data["Sum of Number of Anim
Complete_HWC_Data["Total Staff Hours"] = Complete_HWC_Data["Total Staff Hours"].astype("
Complete_HWC_Data["Total Staff Involved"] = Complete_HWC_Data["Total Staff Involved"].as
Complete_HWC_Data["Latitude Public"] = Complete_HWC_Data["Latitude Public"].astype("floa
Complete_HWC_Data["Longitude Public"] = Complete_HWC_Data["Longitude Public"].astype("fl
#Complete_HWC_Data["Species Common Name"] = Complete_HWC_Data["Species Common Name"].ast
Complete_HWC_Data[Complete_HWC_Data.columns[20:158]] = Complete_HWC_Data[Complete_HWC_Da
#Complete_HWC_Data["Animal Health Status"] = Complete_HWC_Data["Animal Health Status"].a
#Complete_HWC_Data["Cause of Animal Health Status"] = Complete_HWC_Data["Cause of Animal
Complete_HWC_Data.head()
```

Out[3]:

| | UniqueID | Incident Number | Incident Date | Field Unit | Protected Heritage Area | Incident Type | Latitude Public | Longitude Public | Wi F |
|---|---|---|---|---|---|---|---|---|---|
| 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 | Human Wildlife Interaction | 51.161093 | -115.593386 | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Park of Canada | | | |
| **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 Unit | Jasper National Park of Canada | Attractant | 53.050492 | -118.073612 |

5 rows × 158 columns

**Setting up variables here that are used in plotting below:**

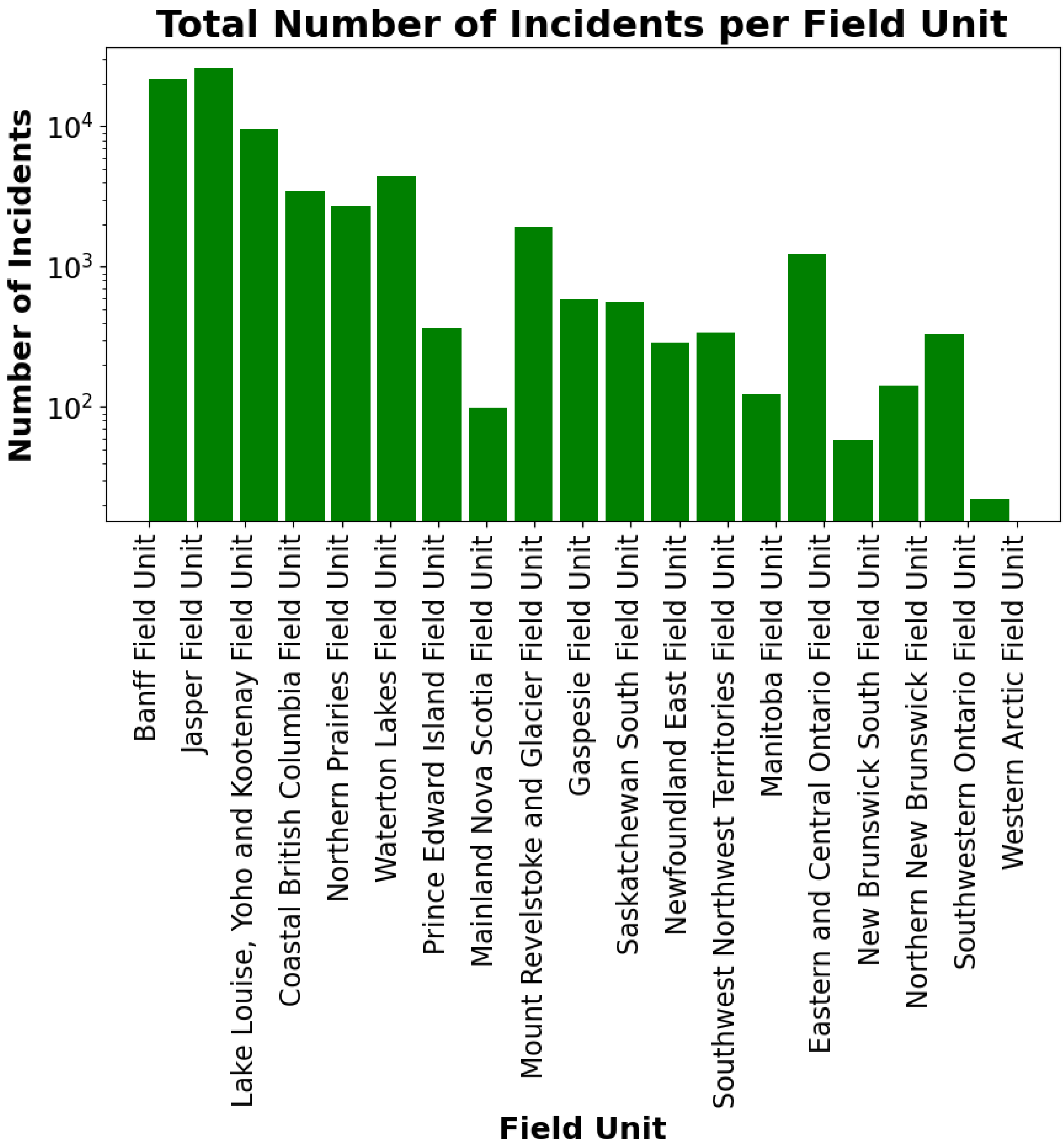
In [4]:
```
Complete_HWC_Data_Month = Complete_HWC_Data.loc[:, ("Incident Type", "Field Unit", "Inci
Complete_HWC_Data_Month["Incident Month"] = pd.to_datetime(Complete_HWC_Data_Month["Inci
Complete_HWC_Data_Month["Incident Month"] = Complete_HWC_Data_Month["Incident Month"].as
Complete_HWC_Data_Month["Incident Month"] = pd.to_datetime(Complete_HWC_Data_Month["Inci
#Complete_HWC_Data_Month
Complete_HWC_Data_Month = Complete_HWC_Data_Month.set_index("Incident Date")
Complete_HWC_Data_Month["Incident Month"] = pd.DatetimeIndex(Complete_HWC_Data_Month.ind
#Complete_HWC_Data_Month
Complete_HWC_Data_Month = Complete_HWC_Data_Month.replace({'Incident Month' : { 1 : "Jan
Complete_HWC_Data_Month["Incident Month"] = Complete_HWC_Data_Month["Incident Month"].as
Complete_HWC_Data_Month["Incident Month"] = Complete_HWC_Data_Month["Incident Month"].ca
Complete_HWC_Data_Month["Incident Month"].cat.categories

IncidentsByMonth = Complete_HWC_Data_Month["Incident Type"].groupby([Complete_HWC_Data_M
IncidentsByMonth = IncidentsByMonth.pivot_table("count", "Incident Month", "Incident Typ
#IncidentsByMonth

Complete_HWC_Data_Year = Complete_HWC_Data.loc[:, ("Incident Type", "Field Unit", "Incid
Complete_HWC_Data_Year["Incident Year"] = pd.to_datetime(Complete_HWC_Data_Year["Inciden
Complete_HWC_Data_Year["Incident Year"] = Complete_HWC_Data_Year["Incident Year"].astype
Complete_HWC_Data_Year["Incident Year"] = pd.to_datetime(Complete_HWC_Data_Year["Inciden
#Complete_HWC_Data_Year

IncidentsByYear = Complete_HWC_Data_Year["Incident Type"].groupby([Complete_HWC_Data_Yea
#IncidentsByYear
IncidentsByYear = IncidentsByYear.pivot_table("count", "Incident Year", "Incident Type")
#IncidentsByYear

Parks = Complete_HWC_Data_Year["Protected Heritage Area"].unique()
#Parks

IncidentsByPark = Complete_HWC_Data_Year["Protected Heritage Area"].groupby([Complete_HW
IncidentsByPark = IncidentsByPark.pivot_table("count", "Protected Heritage Area")
#IncidentsByPark.index
IncidentsByPark=IncidentsByPark.sort_values(by=['count'])
#IncidentsByPark

HighIncParks = IncidentsByPark.loc[IncidentsByPark["count"] >1000]
#HighIncParks

IncidentsByYearByPark = Complete_HWC_Data_Year["Protected Heritage Area"].groupby([Compl
IncidentsByYearByPark = IncidentsByYearByPark.pivot_table("count", "Incident Year", "Pro
#IncidentsByYearByPark
```

```
IncidentsByTypeByPark = Complete_HWC_Data_Year["Incident Type"].groupby([Complete_HWC_Da
IncidentsByTypeByPark = IncidentsByTypeByPark.sort_values(by="count")
IncidentsByTypeByPark = IncidentsByTypeByPark.sort_values(by="Protected Heritage Area")
IncidentsByPark=IncidentsByPark.sort_values(by=['count',])
#IncidentsByTypeByPark

IncidentsByTypeByPark = IncidentsByTypeByPark.pivot_table("count", "Protected Heritage A
#IncidentsByTypeByPark

##
#The for loop I use below doesn't function how I wanted - and I ran out of time to get i
#to do exactly what I wanted it to do (count frequency of Incident Types by Species).
#I'm still using the output of the loop as it does provide counts by species, and
#can be used to plot only the species that are included in more than 100 incidents.
#If I can get the for loop working properly later on, I'll be able to plot the Species
#and Incident Types better. ValueCounts = Complete_HWC_Data["Species Common Name"].value_


ValueCounts = Complete_HWC_Data["Species Common Name"].value_counts()

SpeciesData = Complete_HWC_Data.loc[:, ("Incident Type", "Species Common Name")]

Counts = []
for i in Complete_HWC_Data["Species Common Name"]:
                Counts.append(ValueCounts[i])
SpeciesData.insert(0, "Species_Counts", Counts)
#SpeciesData

HighSpeciesData = SpeciesData.loc[SpeciesData["Species_Counts"] > 100]
HighSpeciesData=HighSpeciesData.sort_values(by=['Species_Counts'])
HighSpeciesCount = HighSpeciesData["Species Common Name"].unique()
#HighSpeciesCount

###

IncBySpecies = Complete_HWC_Data.loc[:, ("Incident Type", "Species Common Name")]
IncBySpecies = IncBySpecies["Incident Type"].groupby([IncBySpecies["Species Common Name"
IncBySpecies = IncBySpecies.set_index("Incident Type")
#Complete_HWC_Data["Incident Type"].loc[Complete_HWC_Data["Species Common Name"]=="Black
IncBySpecies = IncBySpecies.pivot_table("count", "Incident Type", "Species Common Name")


#IncBySpecies

HealthBySpecies = Complete_HWC_Data.loc[:, ("Animal Health Status", "Species Common Name
HealthBySpecies = HealthBySpecies["Animal Health Status"].groupby([HealthBySpecies["Spec
HealthBySpecies = HealthBySpecies.loc[HealthBySpecies["Animal Health Status"] != "Not Ap
HealthBySpecies = HealthBySpecies.set_index("Animal Health Status")
#Complete_HWC_Data["Incident Type"].loc[Complete_HWC_Data["Species Common Name"]=="Black
HealthBySpecies = HealthBySpecies.pivot_table("count", "Animal Health Status", "Species 
#HealthBySpecies

SpeciesByHealth = Complete_HWC_Data.loc[:, ("Animal Health Status", "Species Common Name
SpeciesByHealth = SpeciesByHealth["Animal Health Status"].groupby([SpeciesByHealth["Spec
SpeciesByHealth = SpeciesByHealth.loc[SpeciesByHealth["Animal Health Status"] != "Not Ap
SpeciesByHealth = SpeciesByHealth.set_index("Species Common Name")
SpeciesByHealth = SpeciesByHealth.pivot_table("count","Species Common Name", "Animal Hea
#SpeciesByHealth
```

## The following section generates basic frequency histograms to view the trends in "Incident Types" over various other variables such as human activities, animals involved, cause of animal behaviour, location, etc.

In [26]:

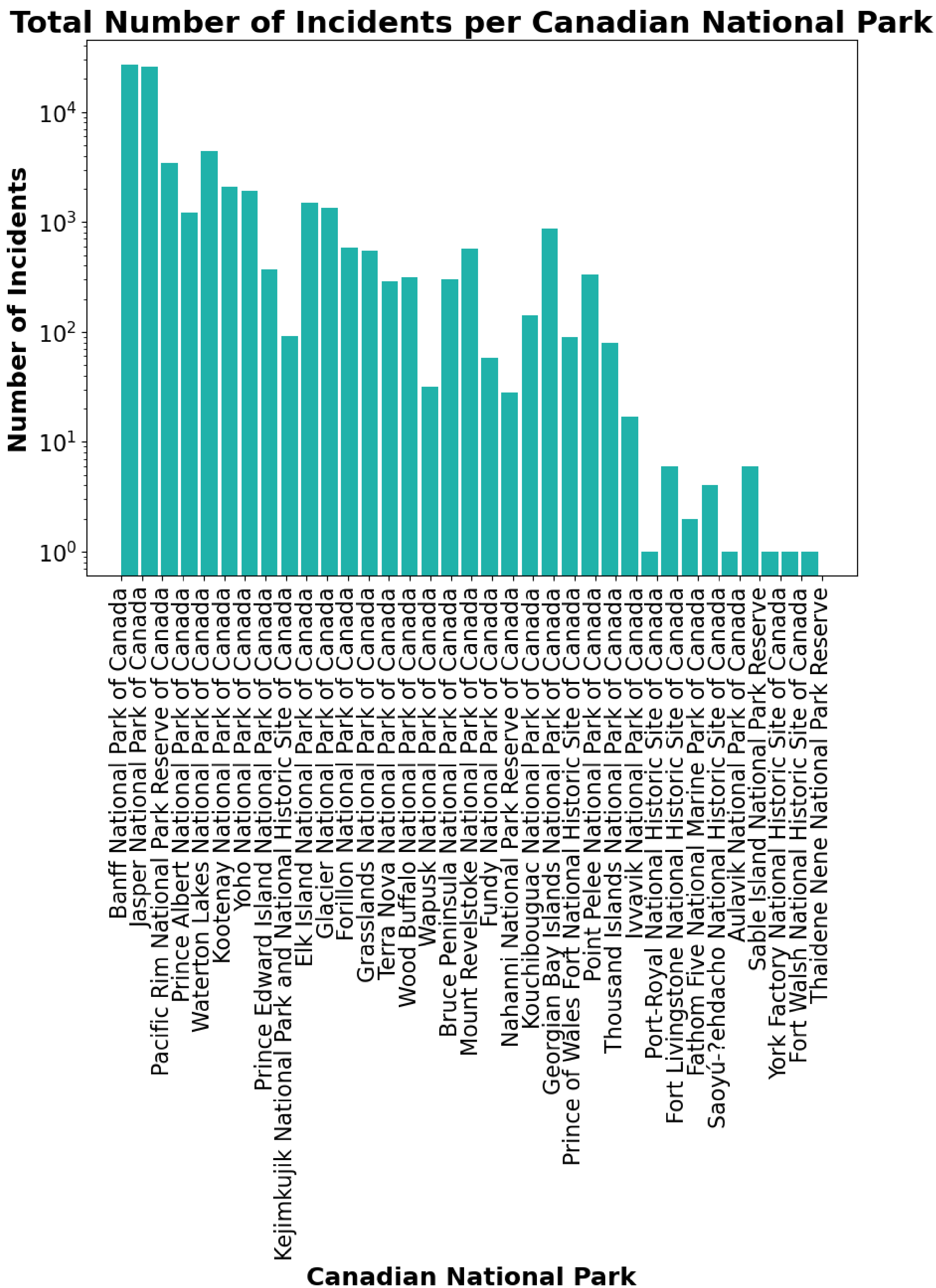
```python
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(10,5));
plt.hist(Complete_HWC_Data["Field Unit"], width = 0.8, bins = 19, color = "green")
plt.ylabel('Number of Incidents', fontweight="bold", size = 18)
plt.xlabel('Field Unit', fontweight="bold", size = 18);
plt.title('Total Number of Incidents per Field Unit', fontweight="bold", size = 22)
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```



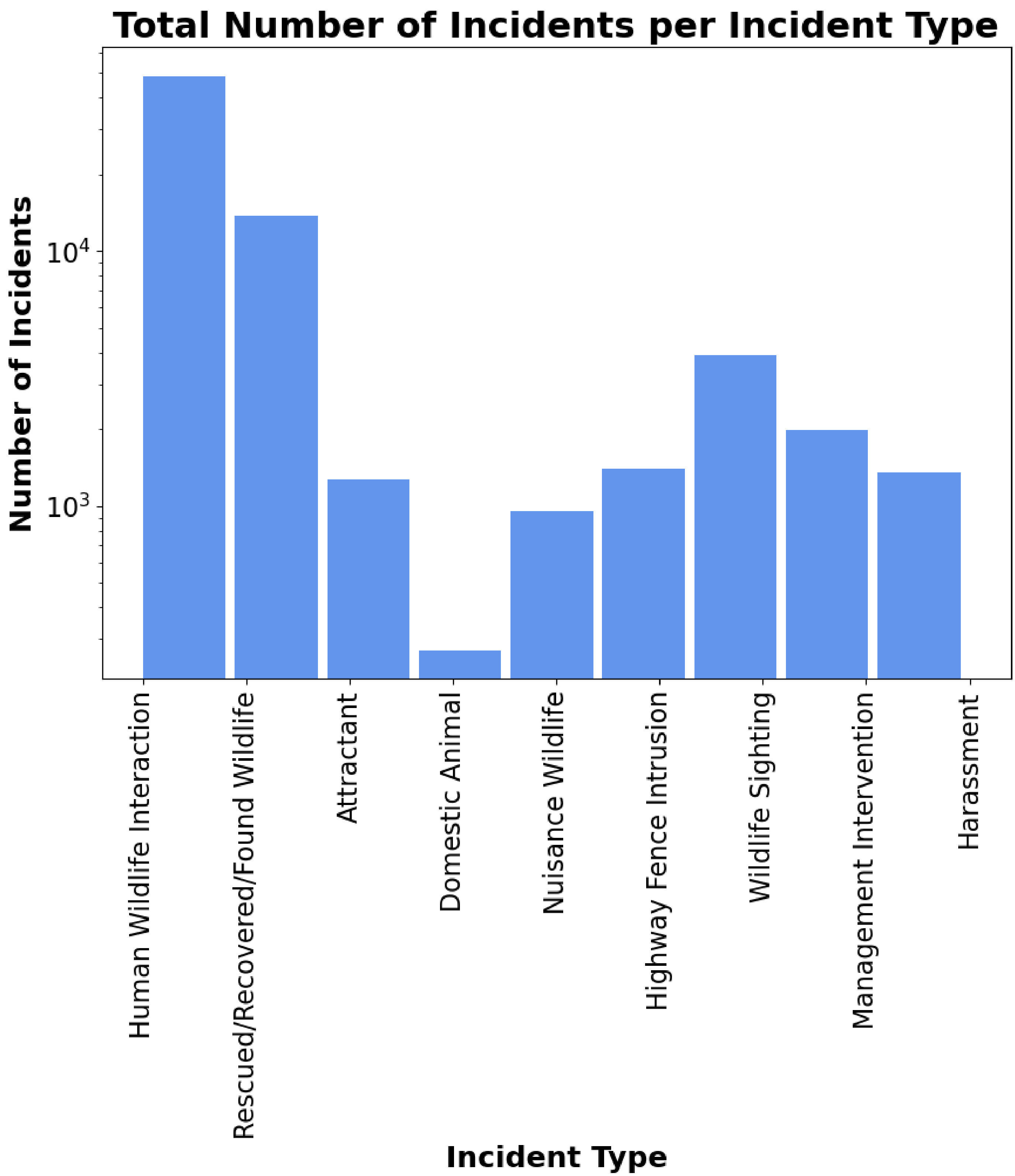

In [6]:

```python
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(10,7));
plt.hist(Complete_HWC_Data["Protected Heritage Area"], width = 0.8, bins = 35, color = "
plt.ylabel('Number of Incidents', fontweight="bold", size = 18)
plt.xlabel('Canadian National Park', fontweight="bold", size = 18);
```

```python
plt.title('Total Number of Incidents per Canadian National Park', fontweight="bold", siz
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```

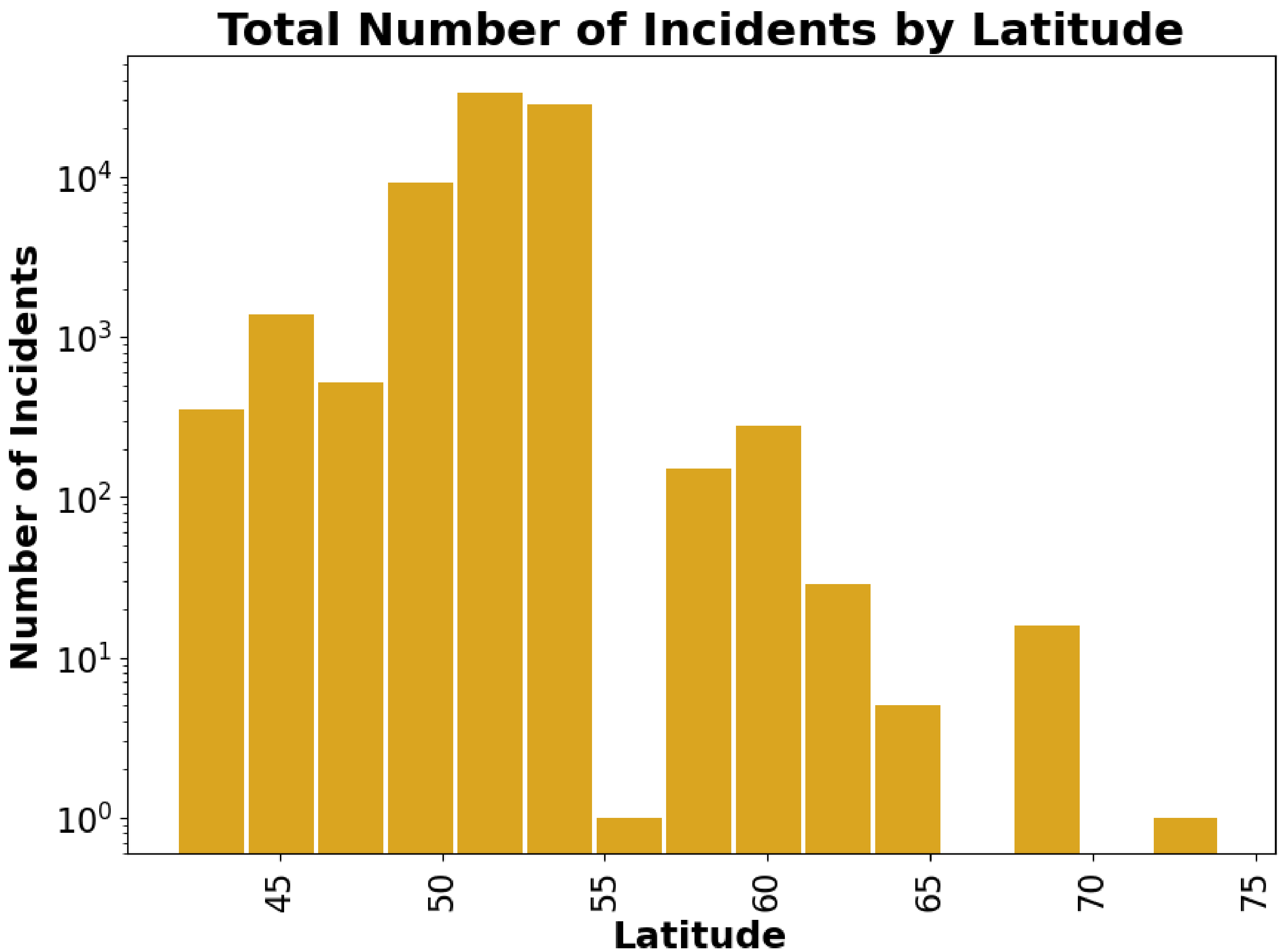


In [27]:
```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(10,7));
plt.hist(Complete_HWC_Data["Incident Type"], width = 0.8, bins = 9, color = "cornflowerb
plt.ylabel('Number of Incidents', fontweight="bold", size = 18)
plt.xlabel('Incident Type', fontweight="bold", size = 18);
plt.title('Total Number of Incidents per Incident Type', fontweight="bold", size = 22)
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```



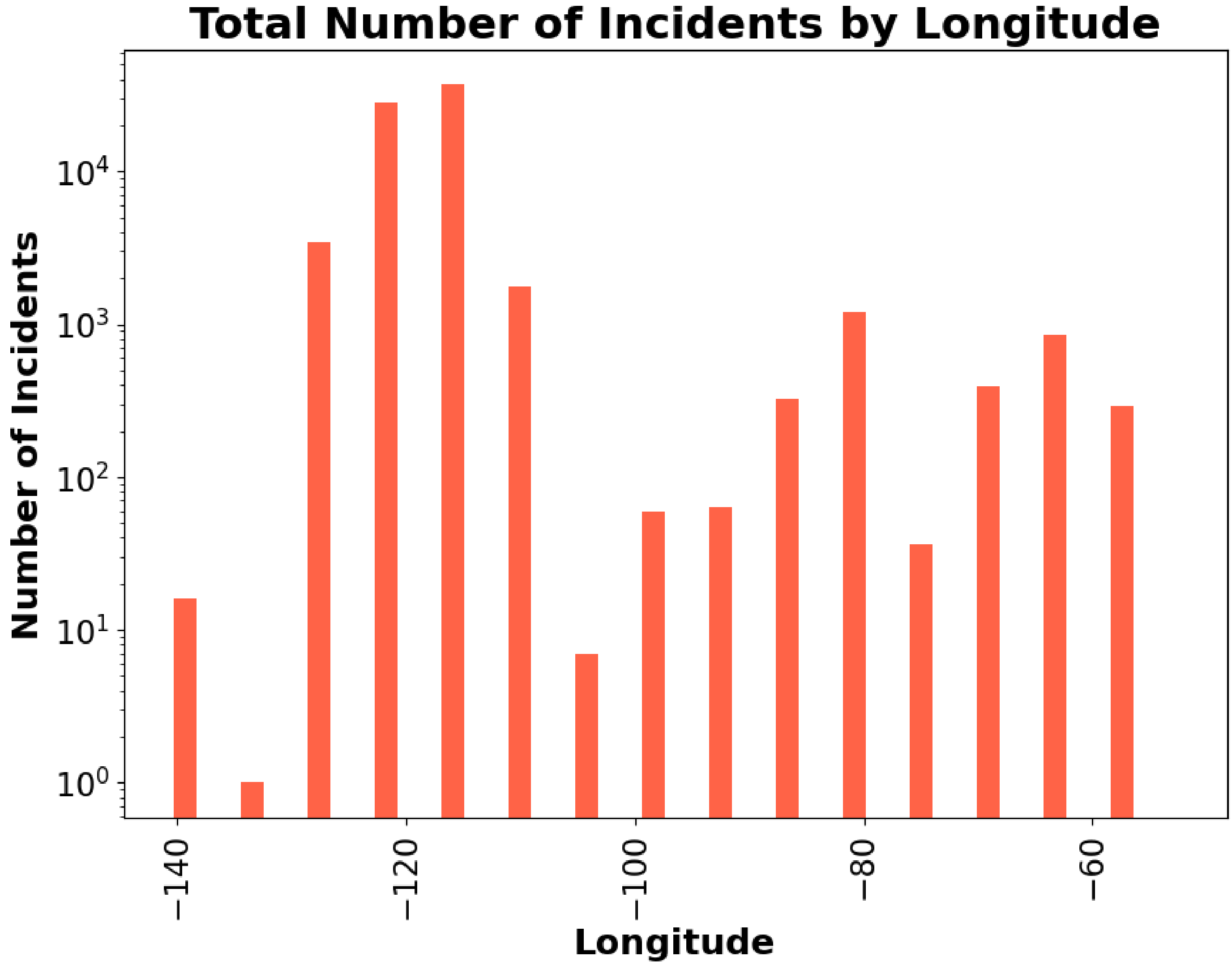

In [28]:
```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h
```

```
plt.figure(figsize=(10,7));
plt.hist(Complete_HWC_Data["Latitude Public"],width=2, bins=15, color = "goldenrod")
plt.ylabel('Number of Incidents', fontweight="bold", size = 18)
plt.xlabel('Latitude', fontweight="bold", size = 18);
plt.title('Total Number of Incidents by Latitude', fontweight="bold", size = 22)
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```



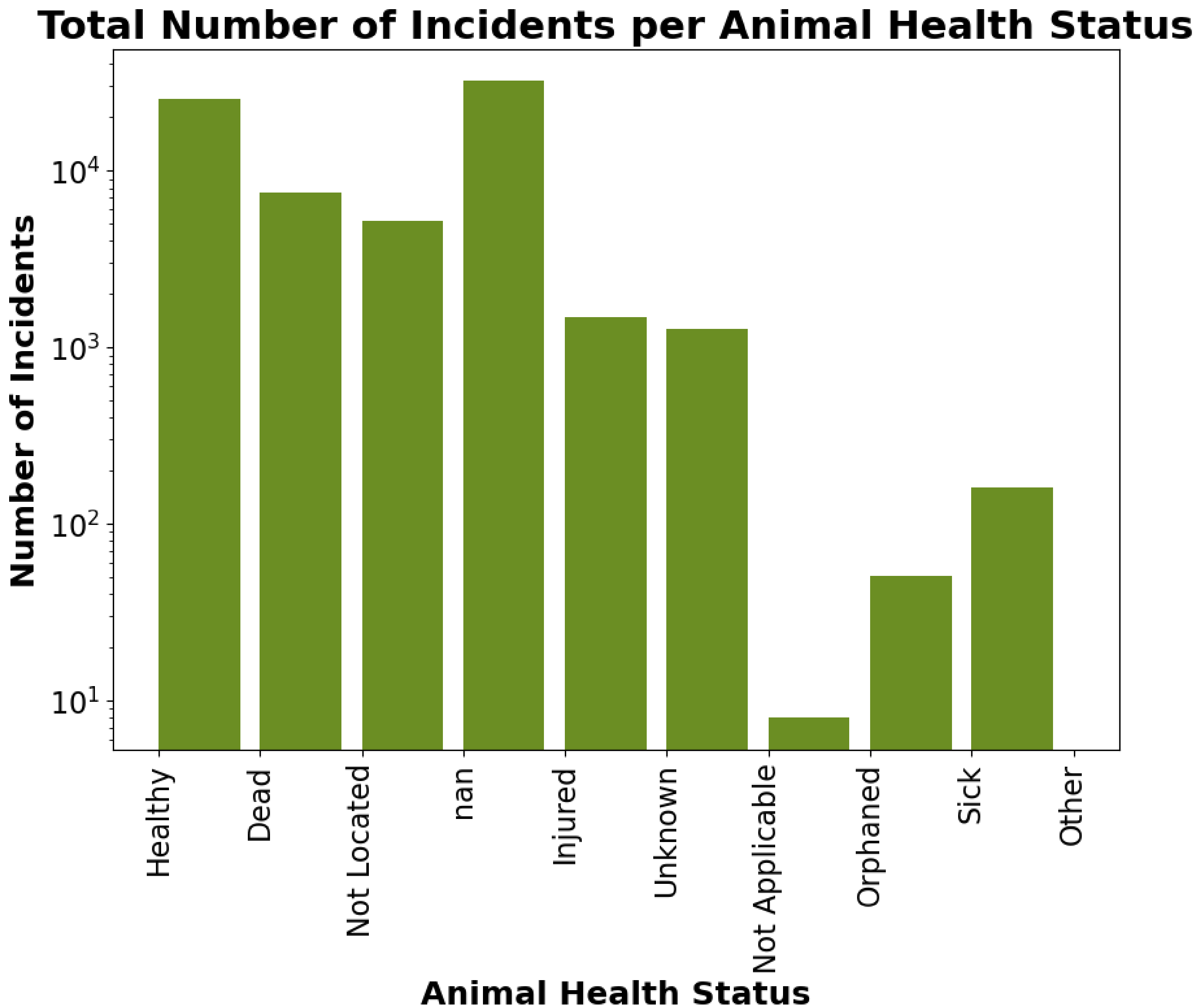

In [29]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(10,7));
plt.hist(Complete_HWC_Data["Longitude Public"], width=2, bins=15, color = "tomato")
plt.ylabel('Number of Incidents', fontweight="bold", size = 18)
plt.xlabel('Longitude', fontweight="bold", size = 18);
plt.title('Total Number of Incidents by Longitude', fontweight="bold", size = 22)
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```



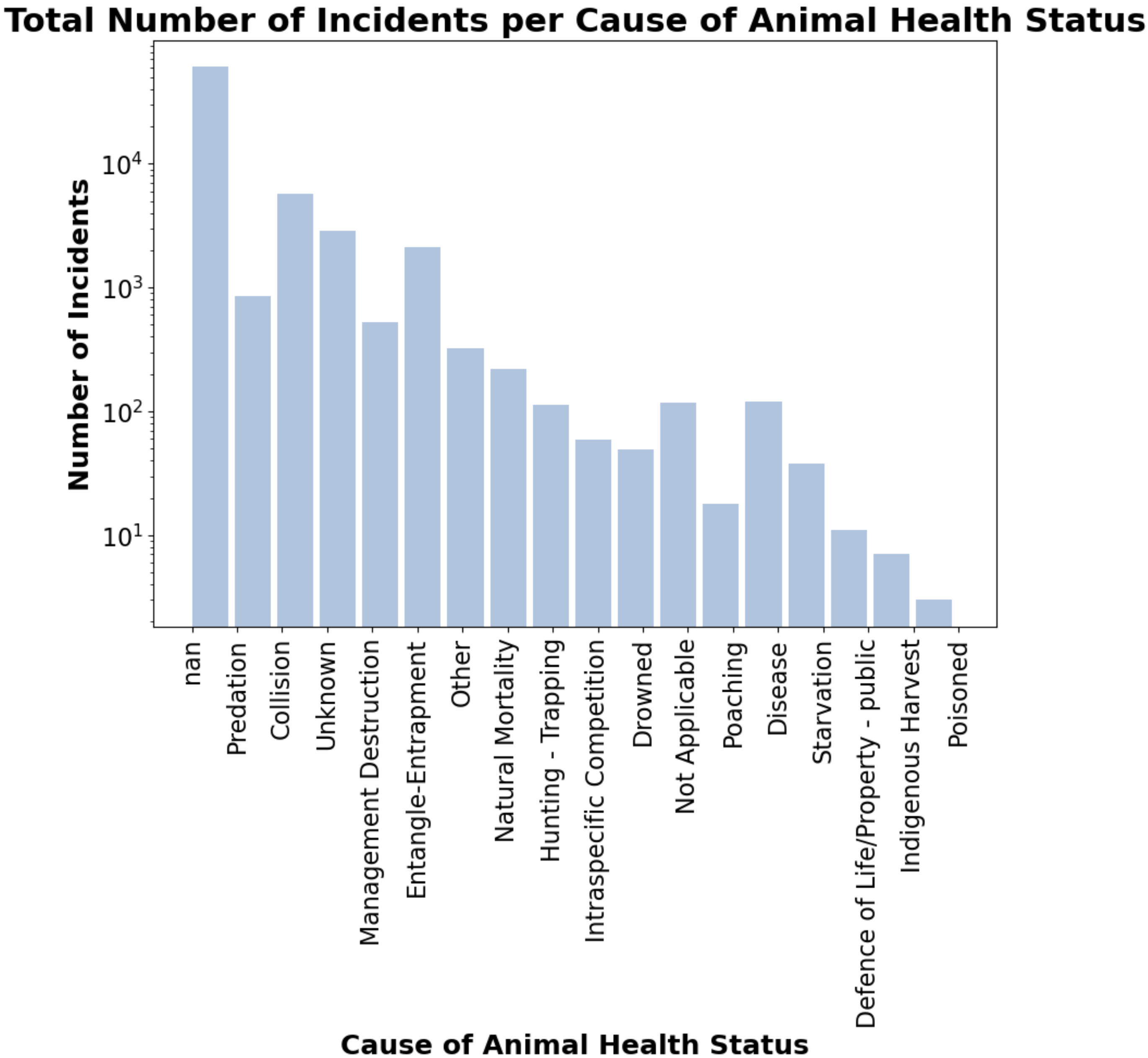

In [30]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(10,7));
plt.hist(Complete_HWC_Data["Animal Health Status"], width = 0.8, bins = 9, color="olived
plt.ylabel('Number of Incidents', fontweight="bold", size = 18);
plt.xlabel('Animal Health Status', fontweight="bold", size = 18);
plt.title('Total Number of Incidents per Animal Health Status', fontweight="bold", size 
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```



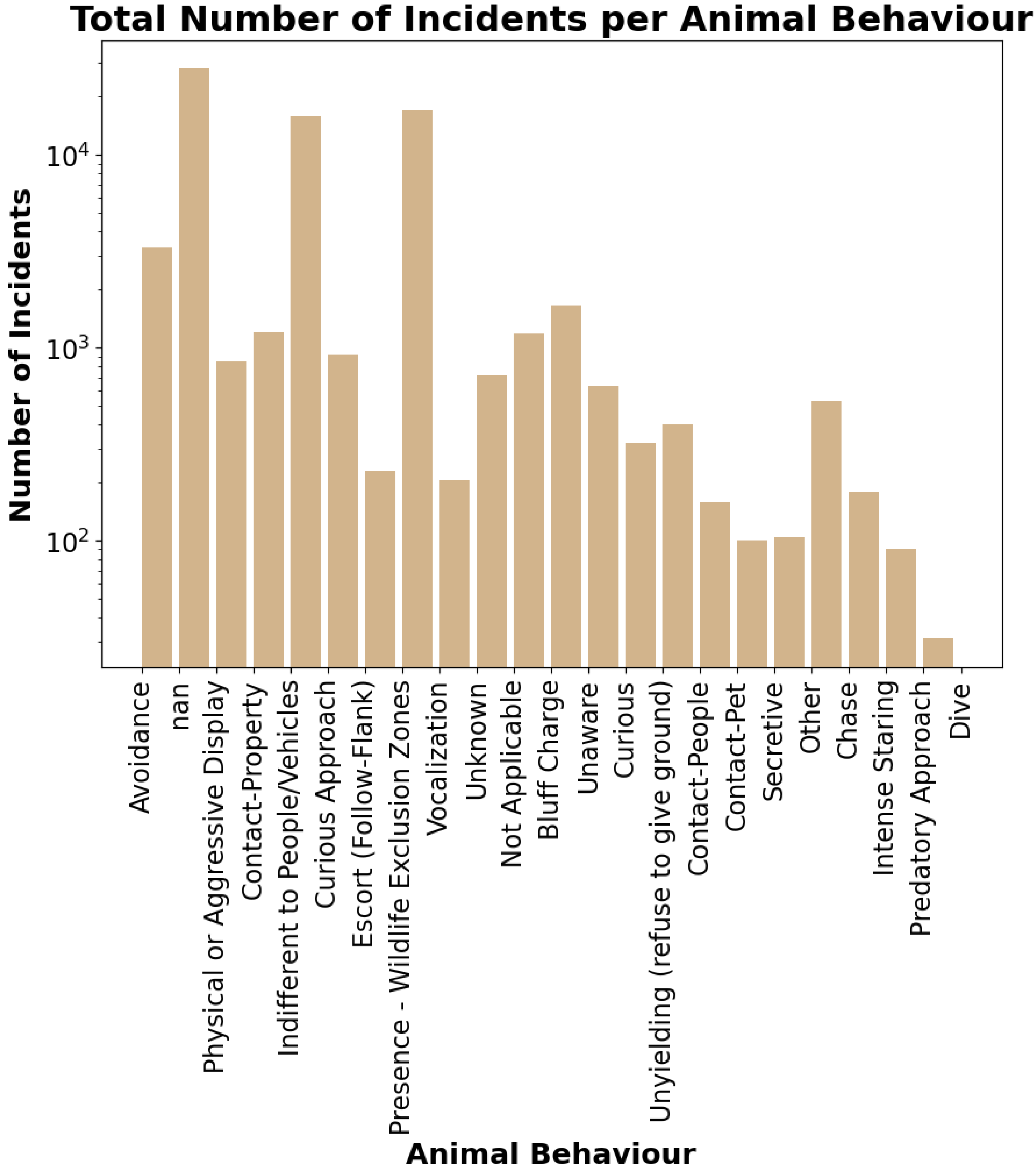

In [36]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(10,7));
plt.hist(Complete_HWC_Data["Cause of Animal Health Status"], width = 0.8, bins = 18, col
plt.ylabel('Number of Incidents', fontweight="bold", size = 18);
plt.xlabel('Cause of Animal Health Status', fontweight="bold", size = 18);
plt.title('Total Number of Incidents per Cause of Animal Health Status', fontweight="bol
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```



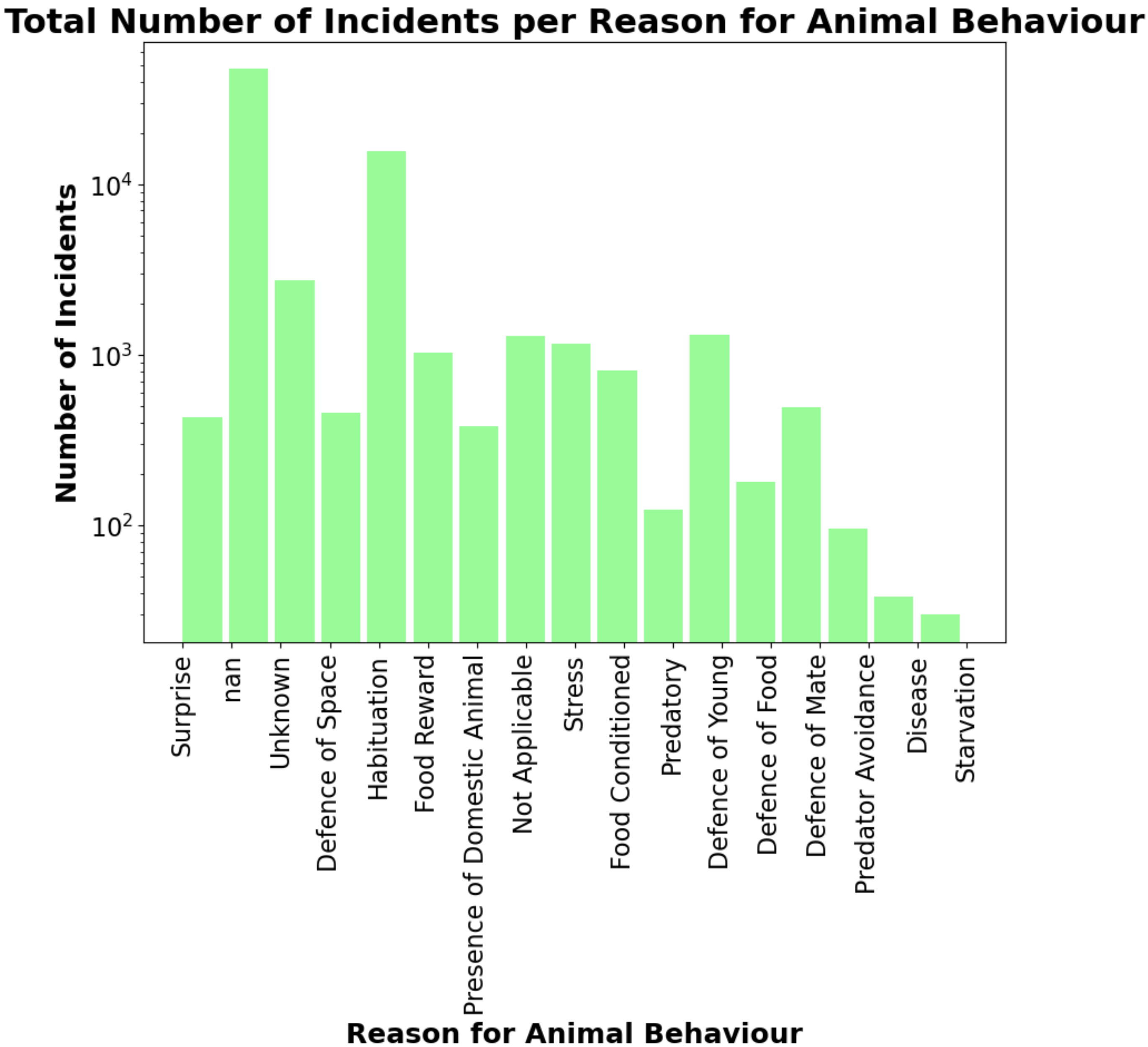

In [37]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(10,7));
plt.hist(Complete_HWC_Data["Animal Behaviour"], width = 0.8, bins = 22, color="tan")
plt.ylabel('Number of Incidents', fontweight="bold", size = 18);
plt.xlabel('Animal Behaviour', fontweight="bold", size = 18);
plt.title('Total Number of Incidents per Animal Behaviour', fontweight="bold", size = 22
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```



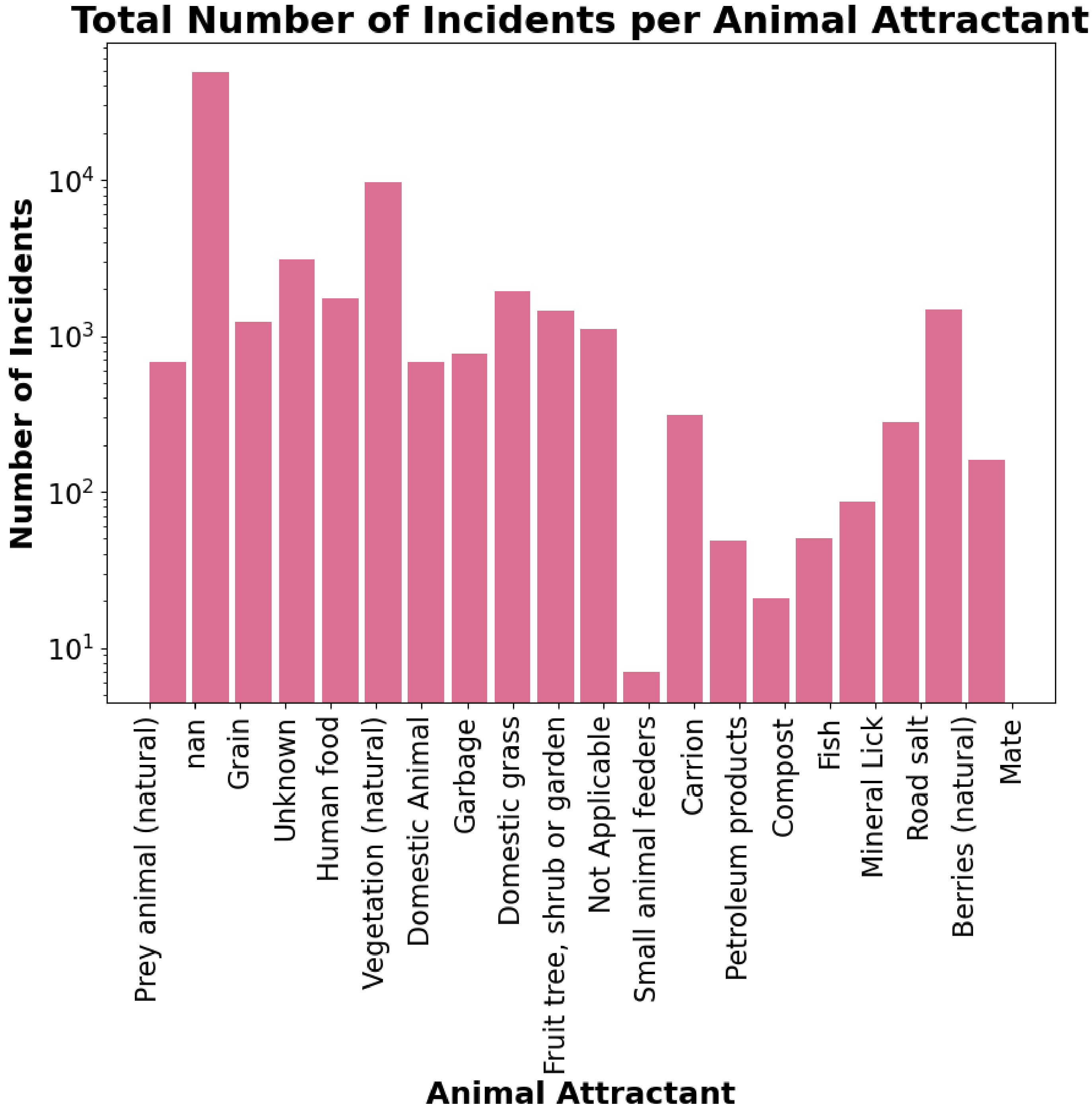

In [39]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(10,7));
plt.hist(Complete_HWC_Data["Reason for Animal Behaviour"], width = 0.8, bins = 17, color
plt.ylabel('Number of Incidents', fontweight="bold", size = 18);
plt.xlabel('Reason for Animal Behaviour', fontweight="bold", size = 18);
plt.title('Total Number of Incidents per Reason for Animal Behaviour', fontweight="bold"
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```



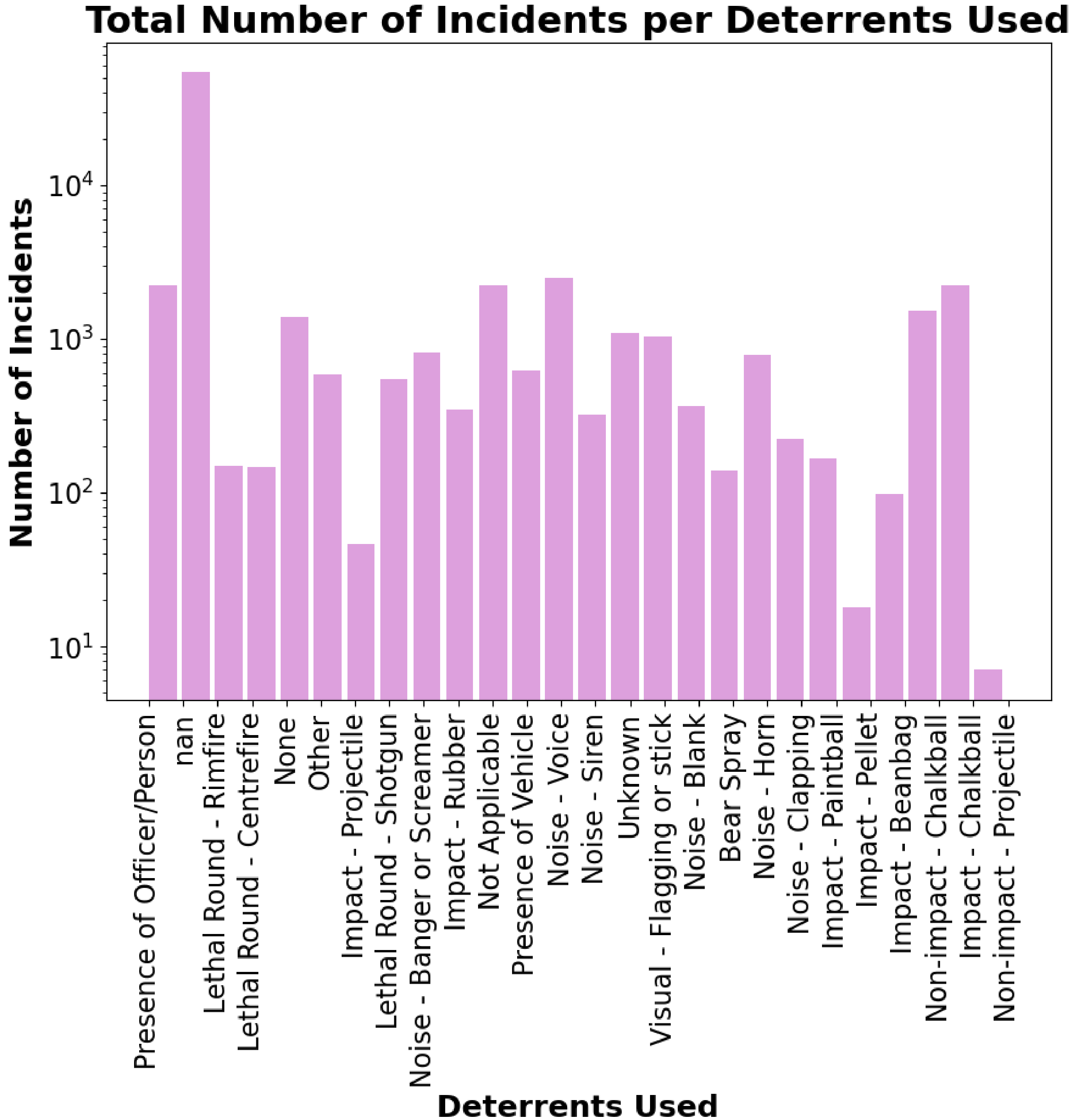

In [40]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(10,7));
plt.hist(Complete_HWC_Data["Animal Attractant"], width = 0.8, bins = 20, color="paleviol
plt.ylabel('Number of Incidents', fontweight="bold", size = 18);
plt.xlabel('Animal Attractant', fontweight="bold", size = 18);
plt.title('Total Number of Incidents per Animal Attractant', fontweight="bold", size = 2
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```



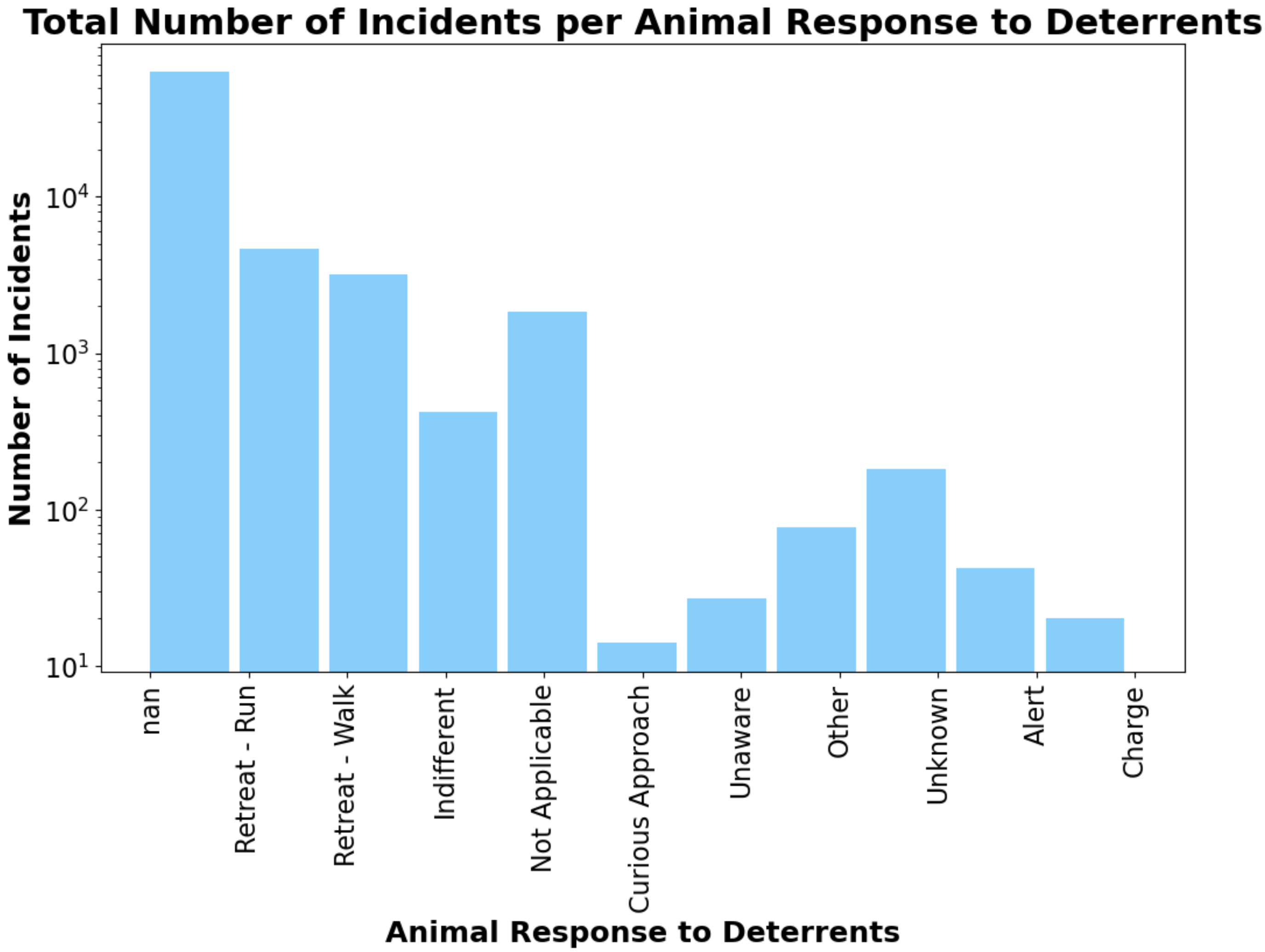

In [41]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(10,7));
plt.hist(Complete_HWC_Data["Deterrents Used"], width = 0.8, bins = 26, color="plum")
plt.ylabel('Number of Incidents', fontweight="bold", size = 18);
plt.xlabel('Deterrents Used', fontweight="bold", size = 18);
plt.title('Total Number of Incidents per Deterrents Used', fontweight="bold", size = 22)
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```




In [42]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(12,7));
plt.hist(Complete_HWC_Data["Animal Response to Deterrents"], width = 0.8, bins = 11, col
plt.ylabel('Number of Incidents', fontweight="bold", size = 18);
plt.xlabel('Animal Response to Deterrents', fontweight="bold", size = 18);
plt.title('Total Number of Incidents per Animal Response to Deterrents', fontweight="bol
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```

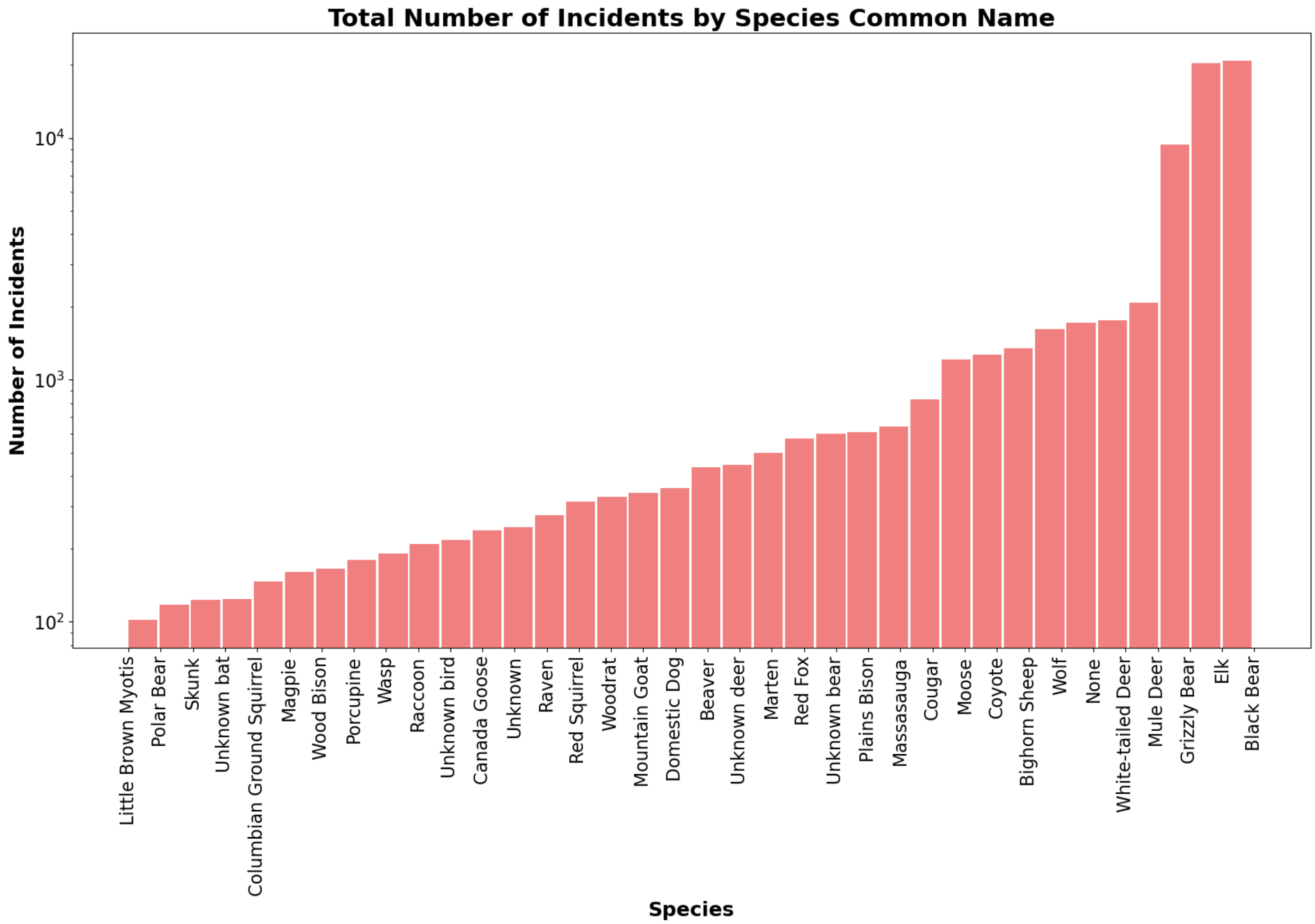



**Notes:**

- The "Species Common Variable" proved difficult to plot. There were too many different species types that the plot was not visually readable. I decided to plot out just the species that had frequency counts higher than 100.

In [43]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(20,10));
plt.hist(HighSpeciesData["Species Common Name"], width = 0.9, bins = 36, color="lightcor
plt.ylabel('Number of Incidents', fontweight="bold", size = 18)
plt.xlabel('Species', fontweight="bold", size = 18);
plt.title('Total Number of Incidents by Species Common Name', fontweight="bold", size =
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.show()
```




**Observations from histograms:**

While it appears as though the “Field Unit” and “Protected Heritage Area” (i.e. Canadian National Park plot) variables overlap, both are interesting to look at. “Protected Heritage Area” directly reflects the 35 Canadian National Parks and “Field Unit” reflects the “name of the administrative unit of Parks Canada Agency that is responsible for management of the incident based on its location” (based on the description provided in the “2. pca-national-human-wildlife-coexistence-header-descriptions.csv” of this data.

The “Field Unit” histogram shows that the majority of the incidents included in the dataset occurred in the “Banff”, “Jasper”, Field Units with over 20000 incidents, with the “Lake Louise, Yoho, and Kootenay Field Unit” being next highest but with significantly less incidents at just under 10,000.

Looking at the Canadian National Parks (i.e. “Protected Heritage Area”), “Banff” and “Jasper” National Parks of Canada have the most incidents with over 25000, and the next highest at “Waterton Lakes” with just under 5000.

Looking at Incident Types, “Human Wildlife Interactions” is the most frequent at near 50,000 and the next highest is “Rescued/Recovered/Found Wildlife” at over 10,000. These will likely be the two Incident Types I focus on in the prediction model and which are of the highest importance in investigating what causes them.

Looking at both Longitude and Latitude, you can see most incidents are occurring between 50 to 55 Latitude and -135 to -125 Longitude. Referencing those latitude and longitude values on a map, I can see that this mainly indicates the incidents are occurring around British Columbia. I am more interested in the location as indicated by the Park name rather than longitude/latitude values.

For Animal Health Status, it is obvious that there are several missing values, with the most frequent occurrence of over 30,000 incidents being nan. Next is "Healthy" with around 25,000. I will be interested in looking at the Healthy animals, but also the Dead and Injured health statuses and what factors affect that status.

Cause of Animal Health Status is not very informative at all with 60,000 missing values of nan. Ignoring the missing values, Collision, and Entrapment are the most prevalent, but with around or under 5000 incidents of each. It would be interesting to look more at the factors involved with those two causes.

Animal Behaviour has over 25000 nan (missing) values. Ignoring the missing values, there are two values that are most significantly frequent: "Presence – Wildlife Exclusion Zones" and "Indifferent to People/Vehicles" with over 15000.

For the Reason for Animal Behaviour, again the most frequent occurrence is the missing values (nan) at over 40,000. Ignoring the missing values, the known value that is significantly the most frequent is "Habituation" with over 15,000.

Looking at Animal Attractants, missing values (nan) are the most prevalent at nearly 50,000. Ignoring the missing values, the known value that is highest is "Vegetation (natural)".

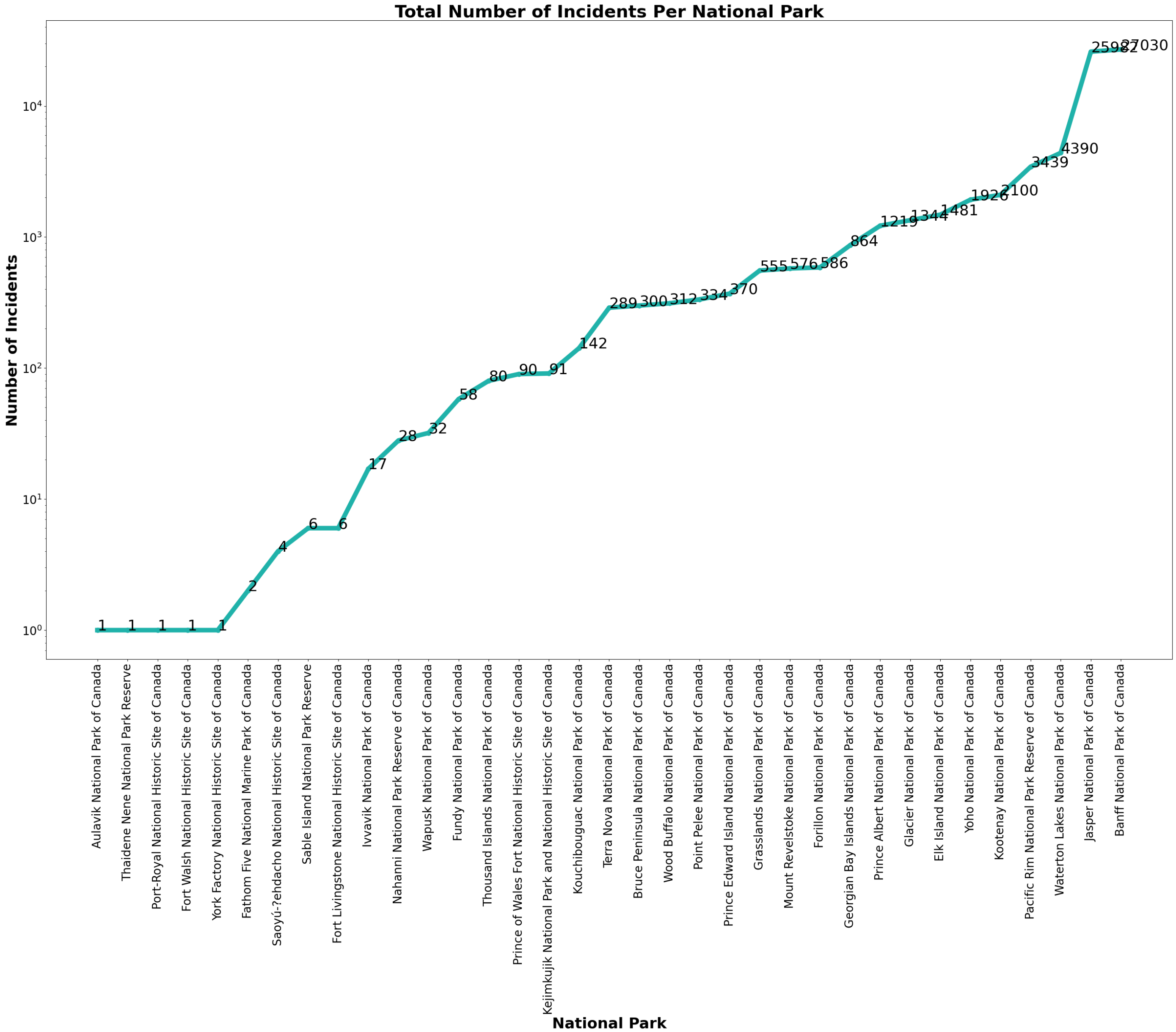
Looking at Deterrents Used, missing values (nan) are the most prevalent. Ignoring the missing values, we can see that Noise – Voice, is the most prevalent at nearly 2500, with Presence of Officer/Person, Not Applicable, and Impact – Chalkball being the next most frequent at over 2000.

Looking at Species Common Name, the 4 most frequent are Black Bear, Elk, Grizzly Bear, and Mule Deer. Interested to look further at the Incident Type distribution across these 4 species.

## The following section generates various plots that dig deeper into features found interesting in the histograms

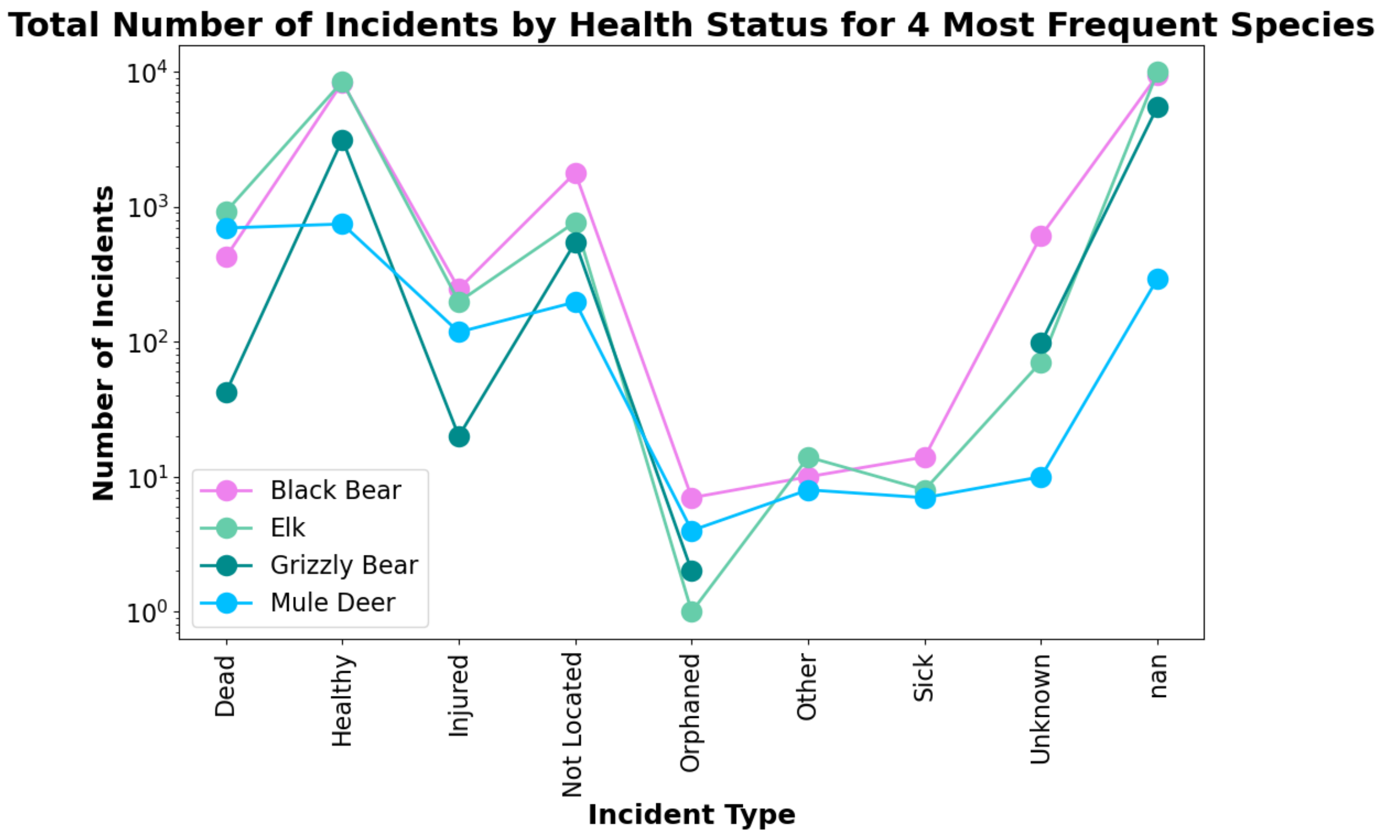
In [44]:

```
#After the basic histogram above and seeing how few incidents several parks had,
#I wanted to view the exact number of incidents that occured in each park.

#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

x = IncidentsByPark.index
y = IncidentsByPark["count"]

fig,ax = plt.subplots(figsize=(35,20))
plt.plot(x, y, label=IncidentsByPark.columns, marker="o", mew=3, linewidth=8, color = "l
plt.xlabel("National Park", fontweight="bold", size = 26)
plt.ylabel("Number of Incidents",  fontweight="bold", size = 26)
plt.xticks(size=20, rotation="vertical", label=IncidentsByPark["count"])
plt.yticks(size=20)
plt.yscale('log')
plt.title("Total Number of Incidents Per National Park", fontweight="bold", size = 30)
for index in range(len(x)):
  ax.text(x[index], y[index], y[index], size=26)
plt.show()
```



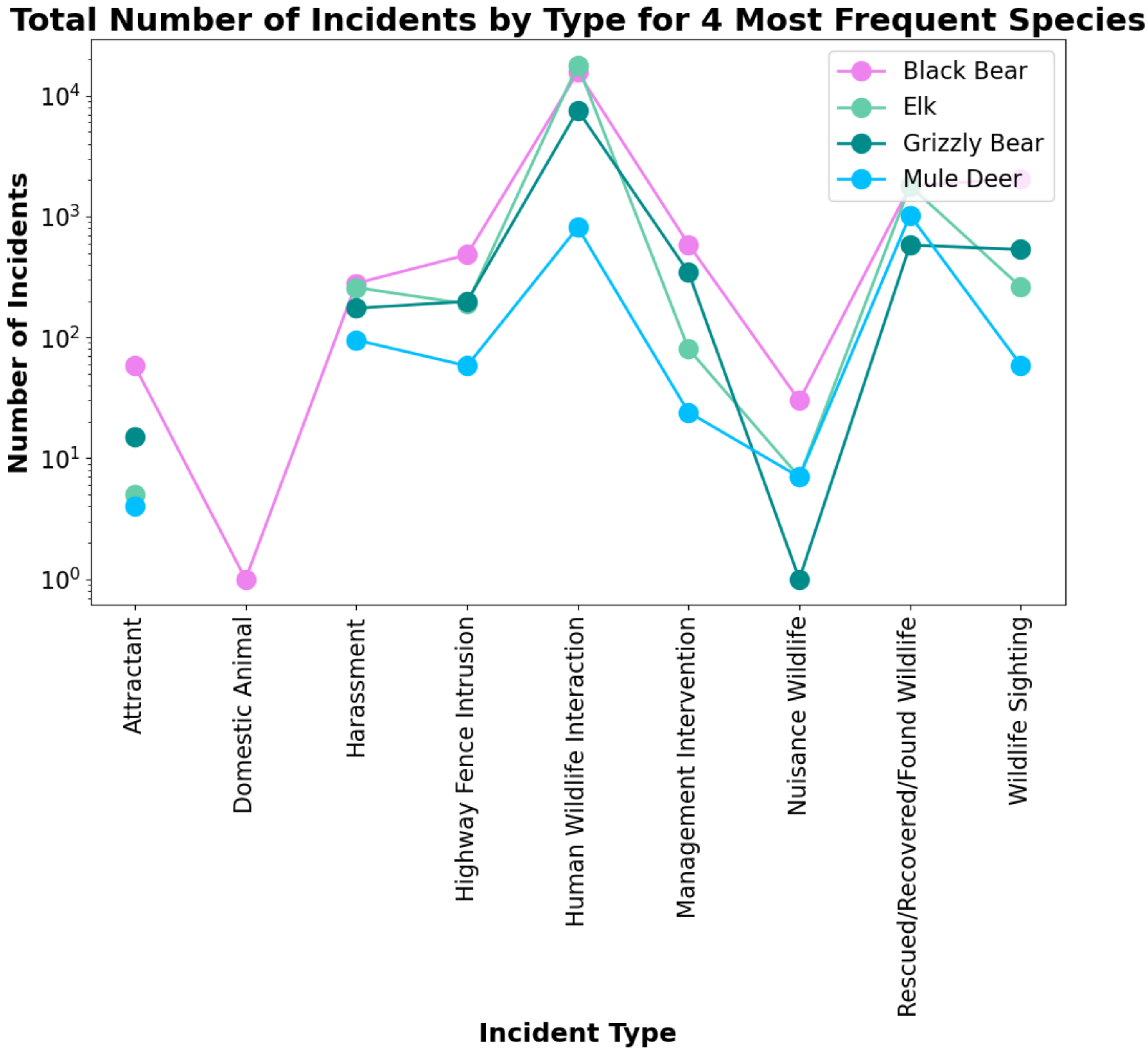

In [47]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(12,7));
plt.plot(HealthBySpecies["Black Bear"], label="Black Bear", marker="o", mew=8, linewidth
plt.plot(HealthBySpecies["Elk"], label="Elk", marker="o", mew=8, linewidth=2)
plt.plot(HealthBySpecies["Grizzly Bear"], label="Grizzly Bear", marker="o", mew=8, linew
plt.plot(HealthBySpecies["Mule Deer"], label="Mule Deer", marker="o", mew=8, linewidth=2
plt.ylabel('Number of Incidents', fontweight="bold", size = 18)
plt.xlabel('Incident Type', fontweight="bold", size = 18);
plt.title('Total Number of Incidents by Health Status for 4 Most Frequent Species', font
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.legend(prop={"size":16});
plt.show()
```



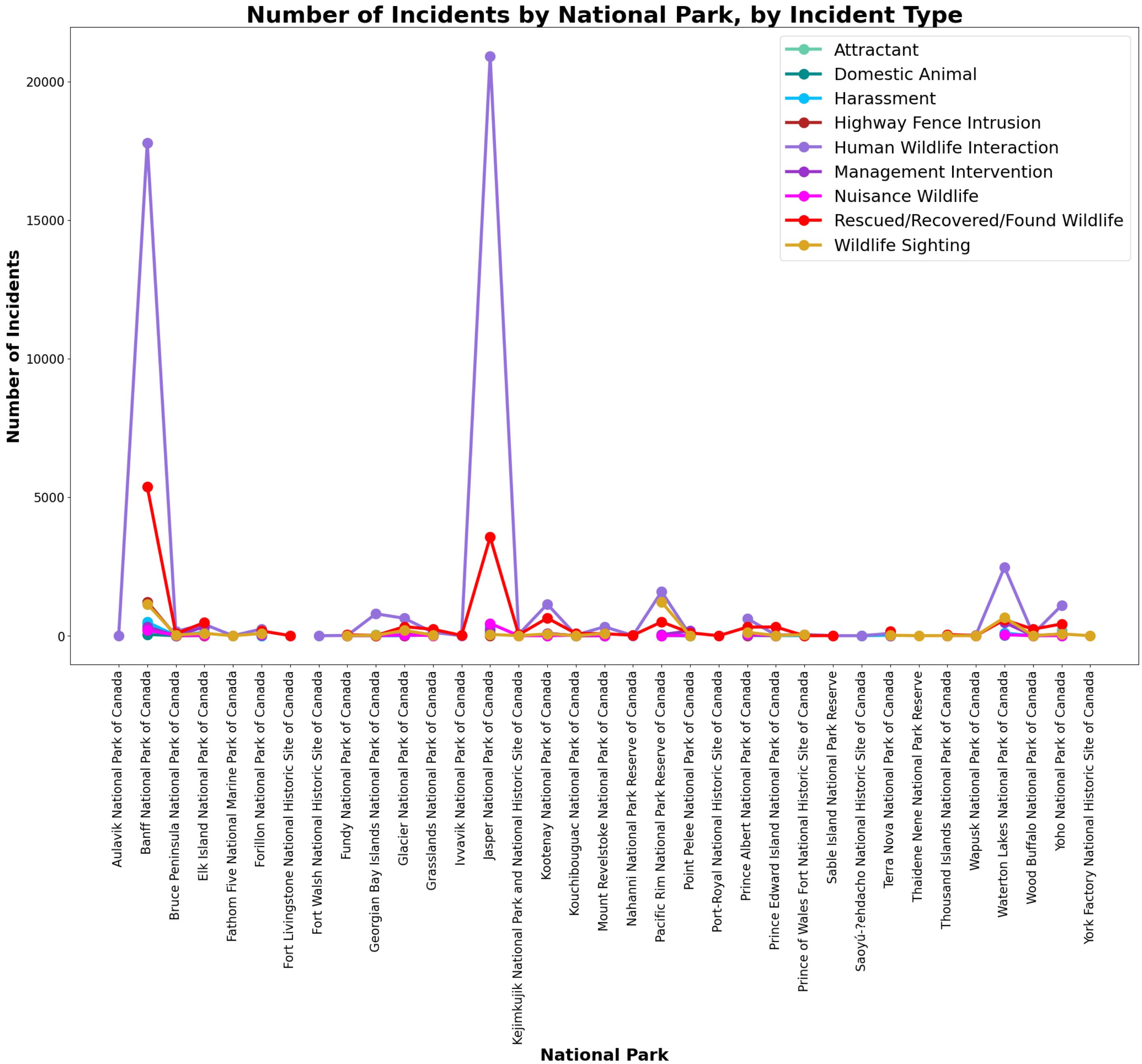

In [48]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plt.figure(figsize=(12,7));
plt.plot(IncBySpecies["Black Bear"], label="Black Bear", marker="o", mew=8, linewidth=2,
plt.plot(IncBySpecies["Elk"], label="Elk", marker="o", mew=8, linewidth=2)
plt.plot(IncBySpecies["Grizzly Bear"], label="Grizzly Bear", marker="o", mew=8, linewidt
plt.plot(IncBySpecies["Mule Deer"], label="Mule Deer", marker="o", mew=8, linewidth=2)
plt.ylabel('Number of Incidents', fontweight="bold", size = 18)
plt.xlabel('Incident Type', fontweight="bold", size = 18);
plt.title('Total Number of Incidents by Type for 4 Most Frequent Species', fontweight="b
plt.xticks(size=16, rotation='vertical')
plt.yticks(size=16)
plt.yscale('log')
plt.legend(loc="upper right", prop={"size":16});
plt.show()
```




In [58]:

```
#Modifying color cycler so that the colors are not repeating across different parks.
plot_colors = ["mediumaquamarine", "darkcyan", "deepskyblue", "firebrick", "mediumpurple
plt.rcParams['axes.prop_cycle'] = plt.cycler(color=plot_colors)


plt.figure(figsize=(25,15));
plt.plot(IncidentsByTypeByPark, label=IncidentsByTypeByPark.columns, marker="o", mew=8,
plt.xlabel("National Park", fontweight="bold", size = 22);
plt.ylabel("Number of Incidents", fontweight="bold", size = 22);
plt.xticks(size=16, rotation="vertical")
plt.yticks(size=16)
plt.legend(loc="upper right", prop={"size":22});
plt.title("Number of Incidents by National Park, by Incident Type", fontweight="bold", s
plt.figure().set_figheight(20);
plt.show();
```

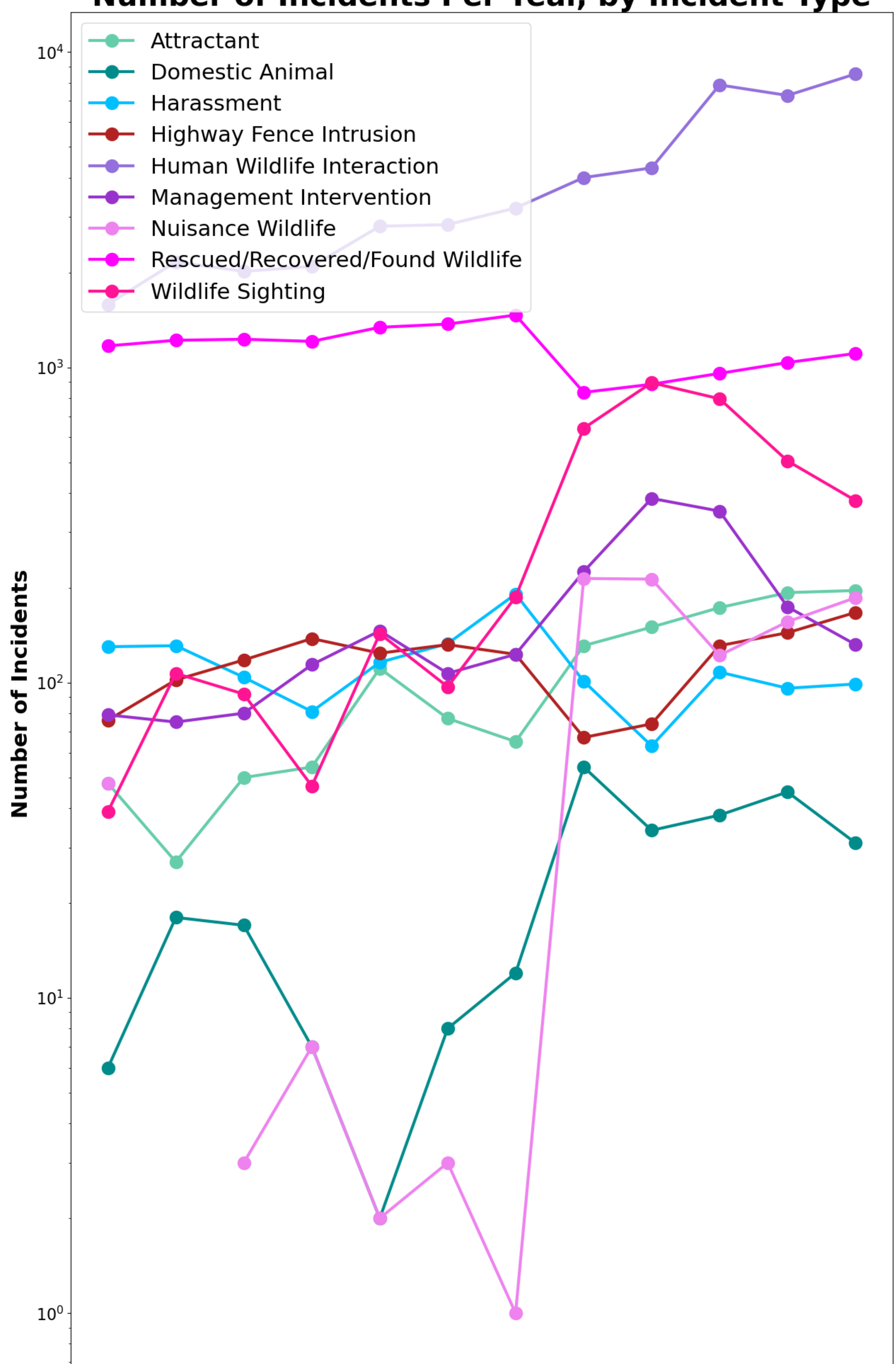



```
<Figure size 640x2000 with 0 Axes>
```

## The following section generates time series plots looking at patterns in Incident Type and/or Frequency and location and time of year.

In [59]:

```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plot_colors = ["mediumaquamarine", "darkcyan", "deepskyblue", "firebrick", "mediumpurple
plt.rcParams['axes.prop_cycle'] = plt.cycler(color=plot_colors)

#Plot elongated to be able to better view the distinct lines with lower values.
plt.figure(figsize=(15,25));
plt.plot(IncidentsByYear, label=IncidentsByYear.columns, marker="o", mew=8, linewidth=3)
plt.xlabel("Incident Date", fontweight="bold", size = 22);
plt.ylabel("Number of Incidents", fontweight="bold", size = 22);
plt.xticks(size=16)
plt.yticks(size=16)
plt.yscale('log')
plt.legend(loc="upper left", prop={"size":22});
plt.title("Number of Incidents Per Year, by Incident Type", fontweight="bold", size = 30
plt.figure().set_figheight(20);
plt.show();
```

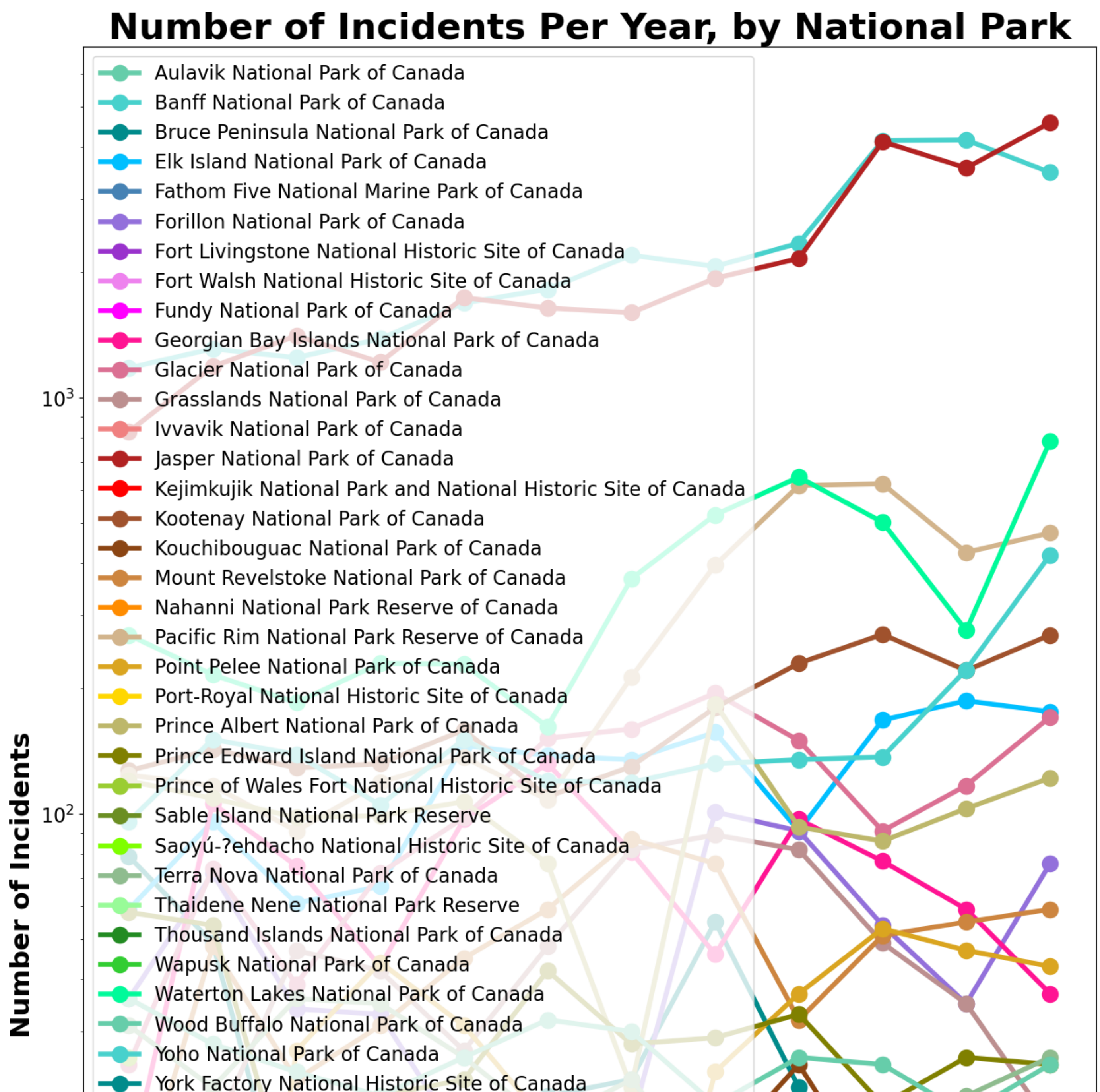


Number of Incidents Per Year, by Incident Type
Attractant
Domestic Animal
Harassment
Highway Fence Intrusion
Human Wildlife Interaction
Management Intervention
Nuisance Wildlife
Rescued/Recovered/Found Wildlife
Wildlife Sighting
Number of Incidents
$10^4$
$10^3$
$10^2$
$10^1$
$10^0$


<Figure size 640x2000 with 0 Axes>

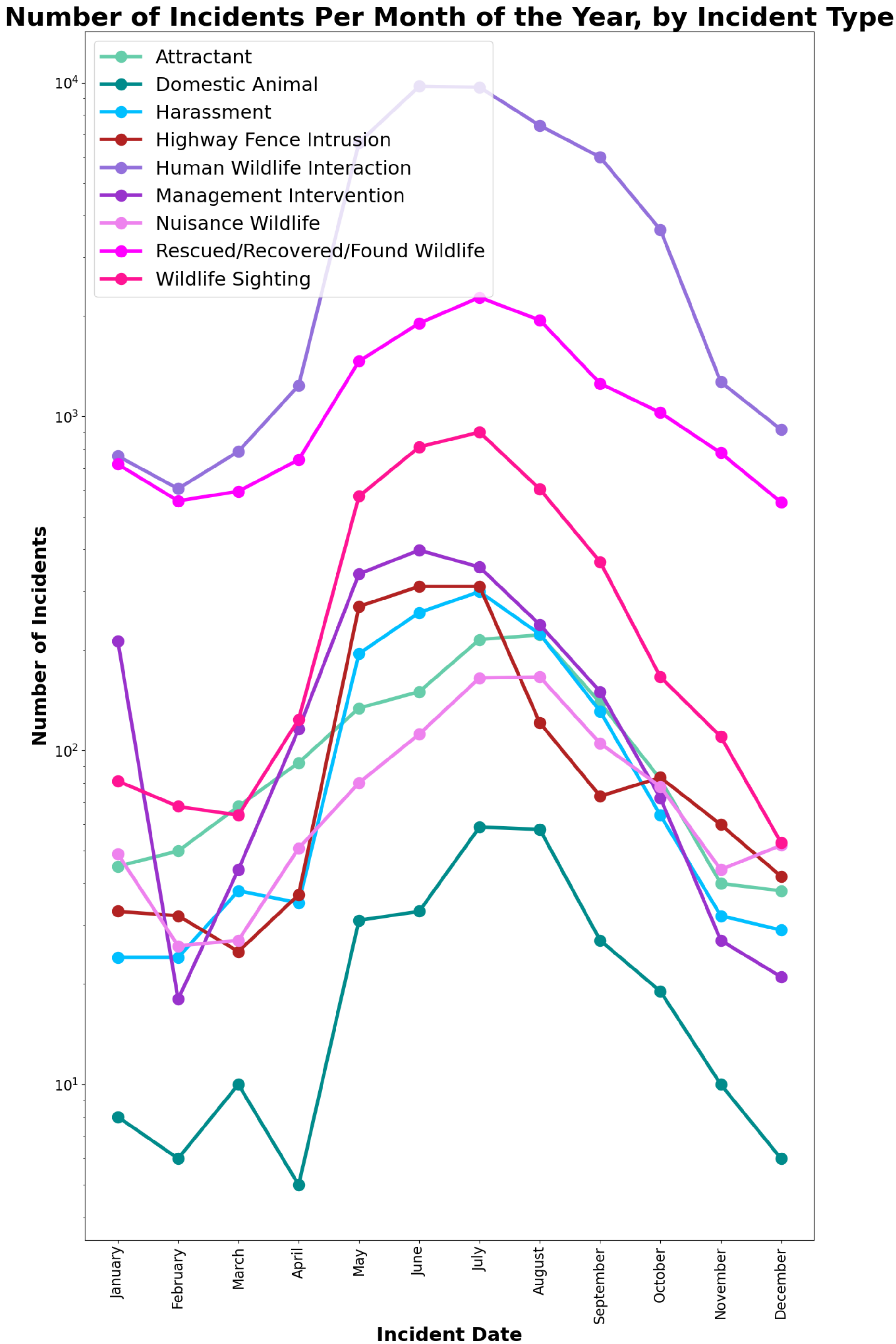
In [60]:

```python
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

#Modifying color cycler so that the colors are not repeating across different parks.
plot_colors = ["mediumaquamarine", "mediumturquoise", "darkcyan", "deepskyblue", "steelb
plt.rcParams['axes.prop_cycle'] = plt.cycler(color=plot_colors)

plt.figure(figsize=(15,25));
plt.plot(IncidentsByYearByPark, label=IncidentsByYearByPark.columns, marker="o", mew=8,
plt.xlabel("Incident Date", fontweight="bold", size = 22);
plt.ylabel("Number of Incidents", fontweight="bold", size = 22);
plt.xticks(size=16)
plt.yticks(size=16)
plt.yscale('log')
plt.legend(loc="upper left", prop={"size":16});
plt.title("Number of Incidents Per Year, by National Park", fontweight="bold", size = 30
plt.figure().set_figheight(20);
plt.show();
```




<Figure size 640x2000 with 0 Axes>

In [61]:
```
#Note, I've applied a logarithmic y-scale to be able to better view the x values which h

plot_colors = ["mediumaquamarine", "darkcyan", "deepskyblue", "firebrick", "mediumpurple
plt.rcParams['axes.prop_cycle'] = plt.cycler(color=plot_colors)

plt.figure(figsize=(15,25));
plt.plot(IncidentsByMonth, label=IncidentsByYear.columns, marker="o", mew=8, linewidth=4
plt.xlabel("Incident Date", fontweight="bold", size = 22);
plt.ylabel("Number of Incidents", fontweight="bold", size = 22);
plt.xticks(rotation="vertical", size=16)
plt.yticks(size=16)
plt.yscale('log')
plt.legend(loc="upper left", prop={"size":22});
plt.title("Number of Incidents Per Month of the Year, by Incident Type", fontweight="bol
plt.figure().set_figheight(20);
plt.show();
```



Number of Incidents Per Month of the Year, by Incident Type
Attractant
Domestic Animal
Harassment
Highway Fence Intrusion
Human Wildlife Interaction
Management Intervention
Nuisance Wildlife
Rescued/Recovered/Found Wildlife
Wildlife Sighting
$10^4$
$10^3$
$10^2$
$10^1$
Number of Incidents
January
February
March
April
May
June
July
August
September
October
November
December
Incident Date

```
<Figure size 640x2000 with 0 Axes>
```

**ANALYSIS AFTER PLOTTING**

**Research Question 1:** What patterns can be found in location and time of year for each of the following variables: human activities, animals involved, cause, and incident type. How do these patterns differ year over year?

- The Parks with the most incidents are Banff and Jasper.
- Almost half the number of Parks have fewer than 100 Incidents
- The majority of Animal Health Status's are "Healthy".
- The majority of Incident Types are Human Wildlife Interaction, and the next highest is Rescued/Recovered/Found Wildlife.
- The most frequent animals involved in incidents are Black bears, Elk, Grizzly bears, and Mule deer.
- Year Trend: Incident Types: the Human Wildlife Interaction Incident Type has Increased significantly, Rescued/Recovered/Found Wildlife has slightly decreased over time. Wildlife Sighting and Management Intervention both seemed to spike around 2018.
- Year Trend: Parks: Jasper and Banff both increased in number of incident types over the years, with Jasper going up in 2021 and Banff going down in 2021. All other parks seems to have mainly remained at a similar level each year, with Waterton and Pacific Rim having some increase over the years.
- Month trend: All Incidents increase during the months between May to October.

**Answer to Research Question #2:** "What incidents are the most concerning (i.e. where there is potential risk for humans or animals)?" Looking at Incident Types, "Human Wildlife Interactions" is the most frequent at near 50,000 and the next highest is "Rescued/Recovered/Found Wildlife" at over 10,000. These incident types are the most common. Rescued/Recovered/Found Wildlife will have the biggest implication/impact on animal health. Human Wildlife Interaction also has implications on humans and animals as it introduces potential risk.

In [25]:
```
#If needed to reset color cycler back to default
#import matplotlib as mpl
#from cycler import cycler
#mpl.rcParams['axes.prop_cycle'] = cycler(color='bgrcmyk')
```