1. The data set need cleaning. Decide what to do with missing values and extra attributes.

In [96]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [97]:
```
# Reading the csv file and storing it in df
df = pd.read_csv('/Users/aayushmakharia/Desktop/P3/Traffic_Crashes_-_Crashe
df.head(10)
```

Out[97]:

	CRASH_RECORD_ID	CRASH_DATE	POSTED_SPEED_LIMIT	TRAF
0	4fd0a3e0897b3335b94cd8d5b2d2b350eb691add56c62d...	7/10/19 17:56	35	
1	009e9e67203442370272e1a13d6ee51a4155dac65e583d...	6/30/17 16:00	35	
2	ee9283eff3a55ac50ee58f3d9528ce1d689b1c4180b4c4...	7/10/20 10:25	30	
3	f8960f698e870ebdc60b521b2a141a5395556bc3704191...	7/11/20 1:00	30	
4	8eaa2678d1a127804ee9b8c35ddf7d63d913c14eda61d6...	7/8/20 14:00	20	
5	00e47f189660cd8ba1e85fc63061bf1d8465184393f134...	3/21/19 22:50	30	
6	0126747fc9ffc0edc9a38abb83d80034f897db0f739eef...	3/26/18 14:23	35	
7	f636d4a51a88015ac89031159b1f1952b8d92e49d11aeb...	7/10/20 22:20	30	
8	76aabcf7c2219a5c90259c96fe94b33834ddb53e0dbcd3...	7/9/20 17:06	10	
9	9c974548026c1b962569040bd8fa08ae643ffc28c15ebd...	6/29/20 17:55	10	

10 rows × 27 columns

In [98]:
```
# finding the percentage of missing data in each column
for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    print('{} - {}%'.format(col, (pct_missing*100)))
```

```
CRASH_RECORD_ID - 0.0%
CRASH_DATE - 0.0%
POSTED_SPEED_LIMIT - 0.0%
TRAFFIC_CONTROL_DEVICE - 0.0%
DEVICE_CONDITION - 0.0%
WEATHER_CONDITION - 0.0%
LIGHTING_CONDITION - 0.0%
FIRST_CRASH_TYPE - 0.0%
TRAFFICWAY_TYPE - 0.0%
ROADWAY_SURFACE_COND - 0.0%
ROAD_DEFECT - 0.0%
CRASH_TYPE - 0.0%
INTERSECTION_RELATED_I - 77.43774695145373%
NOT_RIGHT_OF_WAY - 95.29673624390863%
HIT_AND_RUN_I - 70.62681807139609%
DAMAGE - 0.0%
DATE_POLICE_NOTIFIED - 0.0%
PRIM_CONTRIBUTORY_CAUSE - 0.0%
NUM_UNITS - 0.0%
INJURIES_TOTAL - 0.20015655398517096%
INJURIES_FATAL - 0.20015655398517096%
INJURIES_INCAPACITATING - 0.20015655398517096%
INJURIES_NON_INCAPACITATING - 0.20015655398517096%
INJURIES_REPORTED_NOT_EVIDENT - 0.20015655398517096%
CRASH_HOUR - 0.0%
CRASH_DAY_OF_WEEK - 0.0%
CRASH_MONTH - 0.0%
```

In [99]:
```
# Replacing Missing value with NaN
df.replace('',np.nan)
```

Out[99]:

	CRASH_RECORD_ID	CRASH_DATE	POSTED_SPEED_LIMIT
0	4fd0a3e0897b3335b94cd8d5b2d2b350eb691add56c62d...	7/10/19 17:56	35
1	009e9e67203442370272e1a13d6ee51a4155dac65e583d...	6/30/17 16:00	35
2	ee9283eff3a55ac50ee58f3d9528ce1d689b1c4180b4c4...	7/10/20 10:25	30
3	f8960f698e870ebdc60b521b2a141a5395556bc3704191...	7/11/20 1:00	30
4	8eaa2678d1a127804ee9b8c35ddf7d63d913c14eda61d6...	7/8/20 14:00	20
...	...	...	...
481618	276f2a5ce36d9aa08f7473daaf6b0061615475fc862f0d...	1/18/21 9:00	30
481619	71a084086041d7adc696d9fd71f3cb2d552b9ce93787ba...	1/19/21 21:23	25
481620	28c5281550170efd701934c4cd5f8896b96d43c20c152d...	1/20/21 20:20	30
481621	4983c1d0944603c5e93b599df3ad9c33b4863a6250691b...	1/20/21 17:00	30
481622	f1dba052d8fc8c80d3d693296ff8e0d7cc71d5929677b0...	1/20/21 17:50	30

481623 rows × 27 columns

In [100]:
```
# Changing values to make it more compatible to generate appropriate visual
print("Before Replace")
print(df['INTERSECTION_RELATED_I'].head(10))
df['INTERSECTION_RELATED_I'] = df['INTERSECTION_RELATED_I'].apply(lambda x
print("After Replace")
print(df['INTERSECTION_RELATED_I'].head(10))
```

```
Before Replace
0    NaN
1      Y
2    NaN
3    NaN
4    NaN
5      Y
6    NaN
7    NaN
8    NaN
9    NaN
Name: INTERSECTION_RELATED_I, dtype: object
After Replace
0    NaN
1      1
2    NaN
3    NaN
4    NaN
5      1
6    NaN
7    NaN
8    NaN
9    NaN
Name: INTERSECTION_RELATED_I, dtype: object
```

In [101]:
```
# Changing values to make it more compatible to generate appropriate visual
print("Before Replace")
print(df['NOT_RIGHT_OF_WAY'].head(10))
df['NOT_RIGHT_OF_WAY'] = df['NOT_RIGHT_OF_WAY'].apply(lambda x : '1' if x==
print("After Replace")
print(df['NOT_RIGHT_OF_WAY'].head(10))
```

```
Before Replace
0    NaN
1    NaN
2    NaN
3    NaN
4    NaN
5    NaN
6    NaN
7    NaN
8    NaN
9      Y
Name: NOT_RIGHT_OF_WAY, dtype: object
After Replace
0    NaN
1    NaN
2    NaN
3    NaN
4    NaN
5    NaN
6    NaN
7    NaN
8    NaN
9      1
Name: NOT_RIGHT_OF_WAY, dtype: object
```

In [102]:
```
# Changing values to make it more compatible to generate appropriate visual
print("Before Replace")
print(df['HIT_AND_RUN_I'].head(10))
df['HIT_AND_RUN_I'] = df['HIT_AND_RUN_I'].apply(lambda x : '1' if x=='Y'els
print("After Replace")
print(df['HIT_AND_RUN_I'].head(10))
```

```
Before Replace
0    NaN
1    NaN
2    NaN
3      Y
4    NaN
5    NaN
6    NaN
7    NaN
8      Y
9    NaN
Name: HIT_AND_RUN_I, dtype: object
After Replace
0    NaN
1    NaN
2    NaN
3      1
4    NaN
5    NaN
6    NaN
7    NaN
8      1
9    NaN
Name: HIT_AND_RUN_I, dtype: object
```

In [103]:
```
# Changing values to make it more compatible to generate appropriate visual

print("Before Replace")
print(df['CRASH_DAY_OF_WEEK'].head(10))
df['CRASH_DAY_OF_WEEK'] = df['CRASH_DAY_OF_WEEK'].apply(lambda x : 'Monday'
                                                        else('Tuesday' if x
                                                             else('Wednesda
                                                                  else('Thu
                                                                       else



print("After Replace")
print(df['CRASH_DAY_OF_WEEK'].head(10))
```

```
Before Replace
0    4
1    6
2    6
3    7
4    4
5    5
6    2
7    6
8    5
9    2
Name: CRASH_DAY_OF_WEEK, dtype: int64
After Replace
0    Wednesday
1       Friday
2       Friday
3     Saturday
4    Wednesday
5     Thursday
6       Monday
7       Friday
8     Thursday
9       Monday
Name: CRASH_DAY_OF_WEEK, dtype: object
```

In [104]:
```
# Changing values to make it more compatible to generate appropriate visual
print("Before Replace")
print(df['CRASH_MONTH'].head(10))
df['CRASH_MONTH'] = df['CRASH_MONTH'].apply(lambda x : 'January' if x==1
                                                    else('February' if x==2
                                                    else('March' if x==3
                                                    else('April' if x==4
                                                    else('May' if x==5
                                                    else('June' if x==6
                                                    else('July' if x==7
                                                    else('August' if x==8
                                                    else('September' if x==9
                                                    else('October' if x==10
                                                    else('November' if x==11
                                                    else('December' if x==12
print("After Replace")
print(df['CRASH_MONTH'].head(15))
```

```
Before Replace
0    7
1    6
2    7
3    7
4    7
5    3
6    3
7    7
8    7
9    6
Name: CRASH_MONTH, dtype: int64
After Replace
0       July
1       June
2       July
3       July
4       July
5      March
6      March
7       July
8       July
9       June
10    August
11      June
12      June
13      July
14      July
Name: CRASH_MONTH, dtype: object
```

In [105]:
```
# Changing values to make it more compatible to generate appropriate visual
df['TRAFFIC_CONTROL_DEVICE'] = df['TRAFFIC_CONTROL_DEVICE'].apply(lambda x
df['DEVICE_CONDITION'] = df['DEVICE_CONDITION'].apply(lambda x : 'OTHER' if
df['WEATHER_CONDITION'] = df['WEATHER_CONDITION'].apply(lambda x : 'OTHER'
df['LIGHTING_CONDITION'] = df['LIGHTING_CONDITION'].apply(lambda x : 'OTHER
df['TRAFFICWAY_TYPE'] = df['TRAFFICWAY_TYPE'].apply(lambda x : 'OTHER' if x
df['ROADWAY_SURFACE_COND'] = df['ROADWAY_SURFACE_COND'].apply(lambda x : 'O
df['ROAD_DEFECT'] = df['ROAD_DEFECT'].apply(lambda x : 'OTHER' if x=='UNKNO
df['DAMAGE'] = df['DAMAGE'].apply(lambda x : '1' if x=='OVER $1,500'
                                                 else('2' if x=='$501 - $1
                                                 else('3' if x== '$500 OR
```

In [106]:
```
# Displaying the cleaned datafram df
df
```

Out[106]:

	CRASH_RECORD_ID	CRASH_DATE	POSTED_SPEED_LIMIT
0	4fd0a3e0897b3335b94cd8d5b2d2b350eb691add56c62d...	7/10/19 17:56	35
1	009e9e67203442370272e1a13d6ee51a4155dac65e583d...	6/30/17 16:00	35
2	ee9283eff3a55ac50ee58f3d9528ce1d689b1c4180b4c4...	7/10/20 10:25	30
3	f8960f698e870ebdc60b521b2a141a5395556bc3704191...	7/11/20 1:00	30
4	8eaa2678d1a127804ee9b8c35ddf7d63d913c14eda61d6...	7/8/20 14:00	20
...	...	...	...
481618	276f2a5ce36d9aa08f7473daaf6b0061615475fc862f0d...	1/18/21 9:00	30
481619	71a084086041d7adc696d9fd71f3cb2d552b9ce93787ba...	1/19/21 21:23	25
481620	28c5281550170efd701934c4cd5f8896b96d43c20c152d...	1/20/21 20:20	30
481621	4983c1d0944603c5e93b599df3ad9c33b4863a6250691b...	1/20/21 17:00	30
481622	f1dba052d8fc8c80d3d693296ff8e0d7cc71d5929677b0...	1/20/21 17:50	30

481623 rows × 27 columns

2. Some attributes are more useful if you break them into several attributes. An example of this is already included in the data set where the time, day, and month of the crash are given as separate attributes. These attributes allow you to compare

crashes based on the day of the week, time, or month (season). Are there other attributes that you can break down into smaller attributes to gain more information from?

In [107]:
```
# Dividing the attribute into smaller attributes
df['DATE_POLICE_NOTIFIED']  = pd.to_datetime(df['DATE_POLICE_NOTIFIED'])

df['DATE_POLICE'] = df['DATE_POLICE_NOTIFIED'].dt.date
df['TIME_POLICE'] = df['DATE_POLICE_NOTIFIED'].dt.time
df['HOUR_POLICE'] = df['DATE_POLICE_NOTIFIED'].dt.hour
df['MINUTE_POLICE'] = df['DATE_POLICE_NOTIFIED'].dt.minute
df['CRASH_DATE']  = pd.to_datetime(df['CRASH_DATE'])
df['CRASH_YEAR'] = df['CRASH_DATE'].dt.year
```

In [108]:
```
# Displaying the altered datafram df
df
```

Out[108]:

	CRASH_RECORD_ID	CRASH_DATE	POSTED_SPEED_LIMIT
0	4fd0a3e0897b3335b94cd8d5b2d2b350eb691add56c62d...	2019-07-10 17:56:00	35
1	009e9e67203442370272e1a13d6ee51a4155dac65e583d...	2017-06-30 16:00:00	35
2	ee9283eff3a55ac50ee58f3d9528ce1d689b1c4180b4c4...	2020-07-10 10:25:00	30
3	f8960f698e870ebdc60b521b2a141a5395556bc3704191...	2020-07-11 01:00:00	30
4	8eaa2678d1a127804ee9b8c35ddf7d63d913c14eda61d6...	2020-07-08 14:00:00	20
...	...	...	...
481618	276f2a5ce36d9aa08f7473daaf6b0061615475fc862f0d...	2021-01-18 09:00:00	30
481619	71a084086041d7adc696d9fd71f3cb2d552b9ce93787ba...	2021-01-19 21:23:00	25
481620	28c5281550170efd701934c4cd5f8896b96d43c20c152d...	2021-01-20 20:20:00	30
481621	4983c1d0944603c5e93b599df3ad9c33b4863a6250691b...	2021-01-20 17:00:00	30
481622	f1dba052d8fc8c80d3d693296ff8e0d7cc71d5929677b0...	2021-01-20 17:50:00	30

481623 rows × 32 columns

In [109]:
```
# Data Preparation
df['CRASH_SEASON'] = df['CRASH_MONTH'].apply(lambda x : 'Winter' if x=='Jan
                                                    else('Winter' if x=='Febr
                                                    else('Spring' if x=='Marc
                                                    else('Spring' if x=='Apri
                                                    else('Spring' if x=='May'
                                                    else('Summer' if x=='June
                                                    else('Summer' if x=='July
                                                    else('Summer' if x=='Augu
                                                    else('Fall' if x=='Septem
                                                    else('Fall' if x=='Octobe
                                                    else('Fall' if x=='Novemb
                                                    else('Winter' if x=='Dece
```

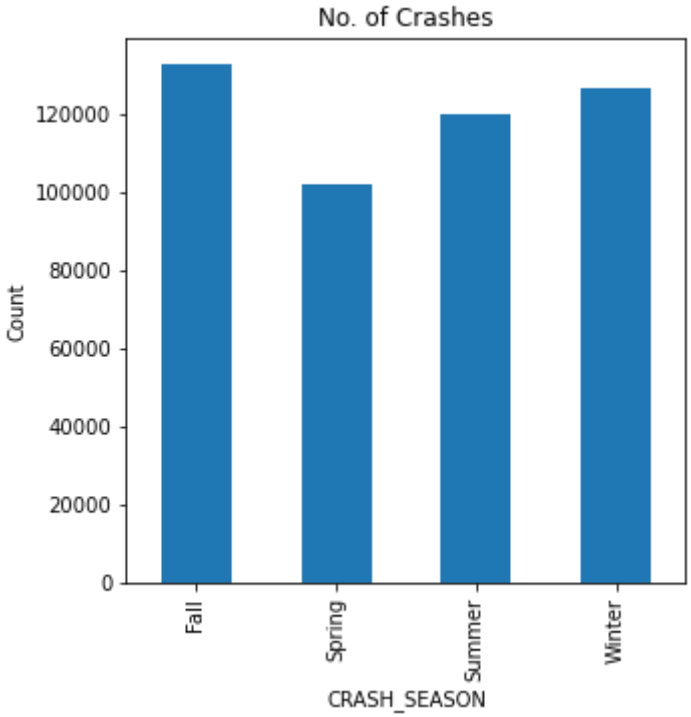
In [110]:
```
# Data Preparation
df['POSTED_SPEED_LIMIT'] = df['POSTED_SPEED_LIMIT'].apply(lambda x : '0-10'
                                                    else('11-20' if x> 10 and
                                                    else('21-30' if x> 20 and
                                                    else('31-40' if x> 30 and
                                                    else('41-50' if x> 40 and
                                                    else('51-60' if x> 50 and
                                                    else('61-70' if x> 60 and
                                                    else('71-80' if x> 70 and
                                                    else('81-90' if x> 80 and
                                                    else('91-100' if x> 90 an
```

In [111]:
```
# Data Preparation
df['DAY_OR_NIGHT'] = df['CRASH_HOUR'].apply(lambda x : 'Night' if (x > 19 o
```

3. What are some insights about the crashes and date/time? You can look into season, day of the week, day/night, lightning, weather, etc

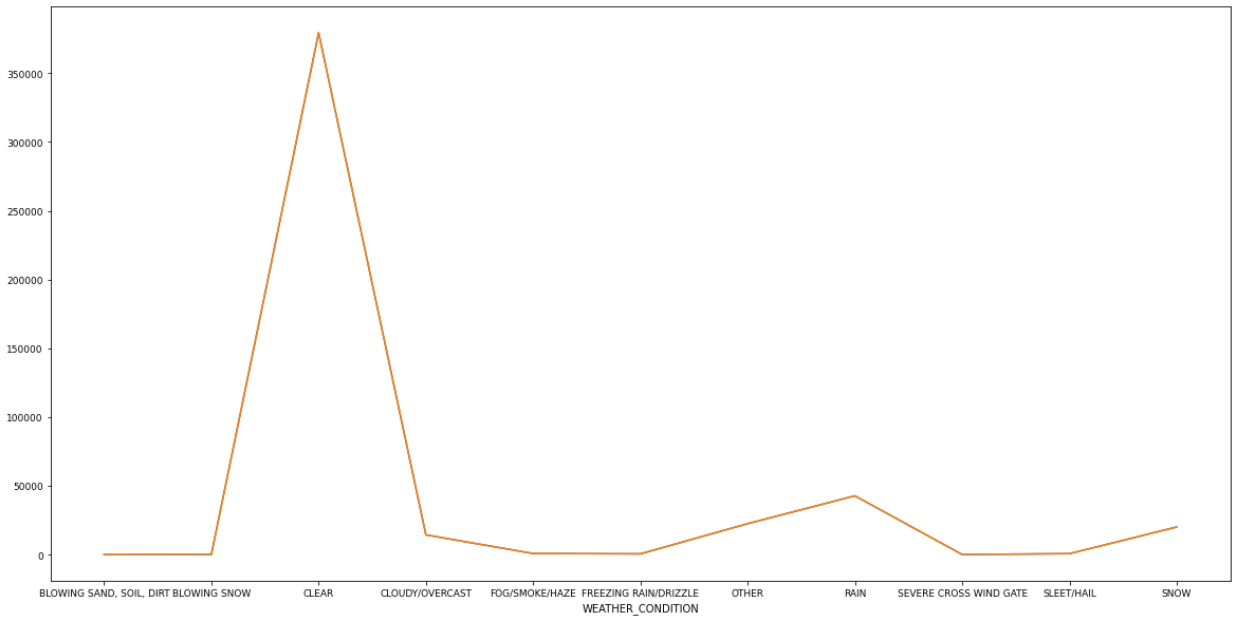
In [112]:
```
# 3.1
from matplotlib import pyplot as plt
from pylab import rcParams
rcParams['figure.figsize'] = 5, 5
x = df.groupby("CRASH_SEASON").size()
plt.ylabel("Count")
plt.title("No. of Crashes")
x.plot.bar()
x
```

Out[112]:
```
CRASH_SEASON
Fall      132732
Spring    102120
Summer    120098
Winter    126673
dtype: int64
```



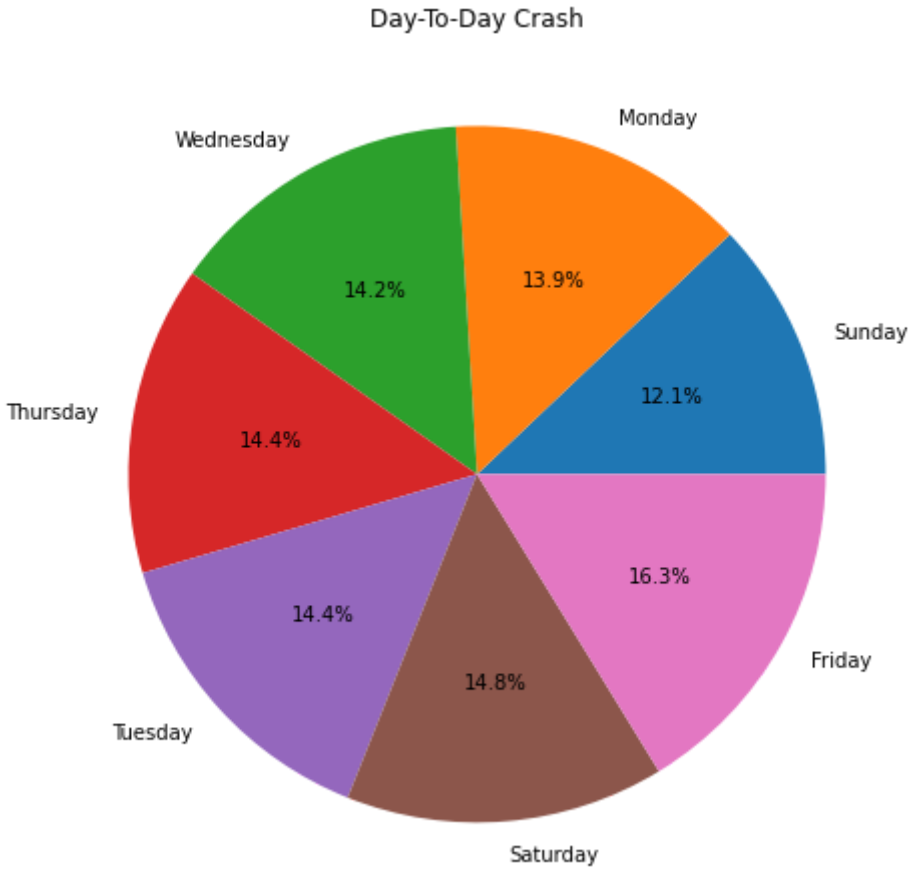
In [113]:
```
# 3.2
x1 = df.groupby("WEATHER_CONDITION").size()
from pylab import rcParams
rcParams['figure.figsize'] = 20, 10
plot = x1.plot(x='Weather', y='Count', xticks=range(0,12),fontsize=9)
x1.plot.line()
x1
```

Out[113]:
```
WEATHER_CONDITION
BLOWING SAND, SOIL, DIRT         2
BLOWING SNOW                   161
CLEAR                       379417
CLOUDY/OVERCAST              14410
FOG/SMOKE/HAZE                 824
FREEZING RAIN/DRIZZLE          574
OTHER                        22537
RAIN                         42770
SEVERE CROSS WIND GATE          98
SLEET/HAIL                     716
SNOW                         20114
dtype: int64
```



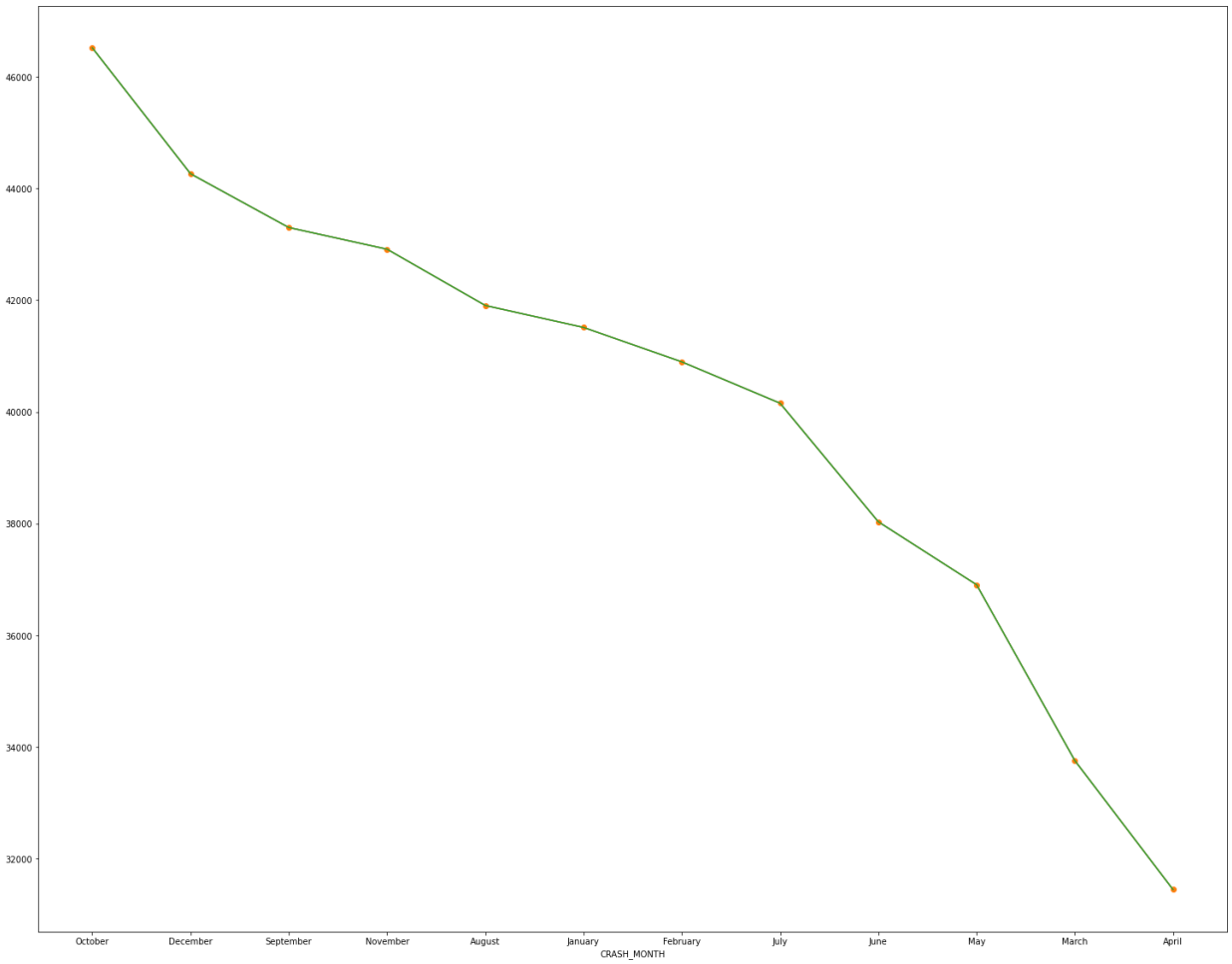
In [114]:
```
# 3.3
x2 = df.groupby("CRASH_DAY_OF_WEEK").size().sort_values(ascending=True)
from pylab import rcParams
rcParams['figure.figsize'] = 8, 8
plt.title("Day-To-Day Crash")
x2.plot.pie(autopct='%1.1f%%')
x2
plt.ylabel('')
```

Out[114]:
```
Text(0, 0.5, '')
```



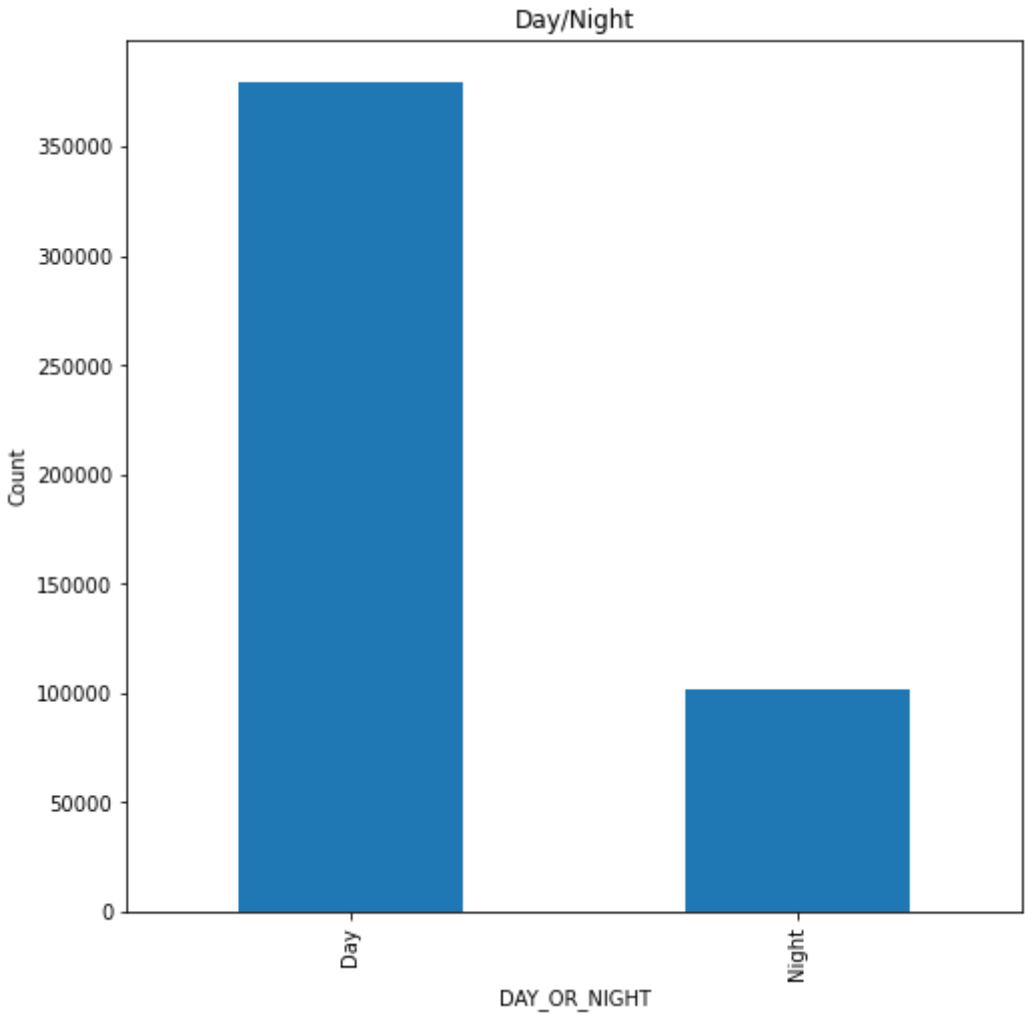
In [124]:
```
# 3.4
x3 = df.groupby("CRASH_MONTH").size().sort_values(ascending=False)
plot = x3.plot(x='Month', y='Count', xticks=range(0,12))
rcParams['figure.figsize'] = 25,20
x3.plot.line(marker="o")
x3.plot()
x3
```

Out[124]:
```
CRASH_MONTH
October      46518
December     44263
September    43301
November     42913
August       41906
January      41513
February     40897
July         40157
June         38035
May          36908
March        33761
April        31451
dtype: int64
```



In [128]:
```
# 3.5
x4 = df.groupby("DAY_OR_NIGHT").size().sort_values(ascending=False)
plt.ylabel("Count")
plt.title("Day/Night")
from pylab import rcParams
rcParams['figure.figsize'] = 8, 8
x4.plot.bar()
x4
```

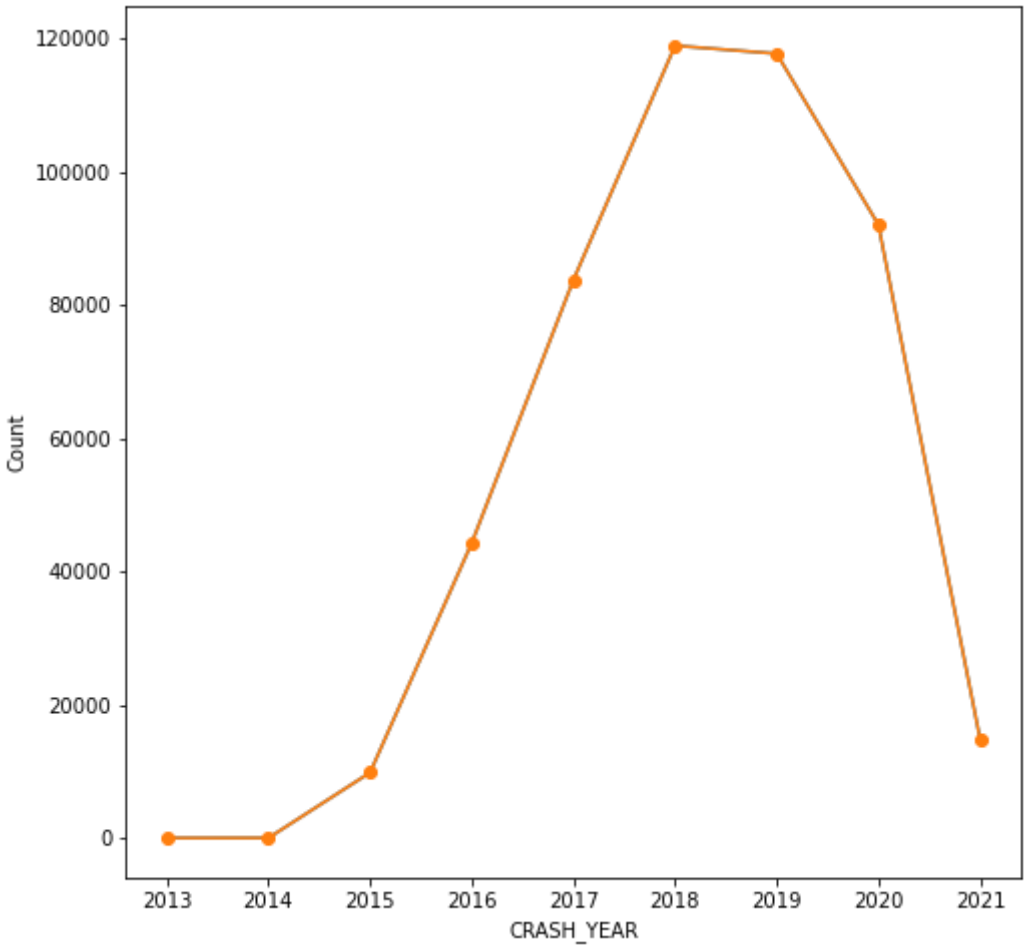
Out[128]:
```
DAY_OR_NIGHT
Day      379868
Night    101755
dtype: int64
```



4. Has number of deadly crashes increased recently? Look at the data over the years. Can you identify any significant increase/decrease?

In [129]:
```
y = df.groupby("CRASH_YEAR").size()
plot = y.plot(x='Year', y='Count')
from pylab import rcParams
rcParams['figure.figsize'] = 8, 8
plt.ylabel("Count")
y.plot.line(marker="o")
print("From 2013 to 2018 there was a significant growth reaching the peak")
print("From 2018 to 2019 there was a slight dip")
print("From 2019 to 2021 the there was a significant dip reaching as low as
```

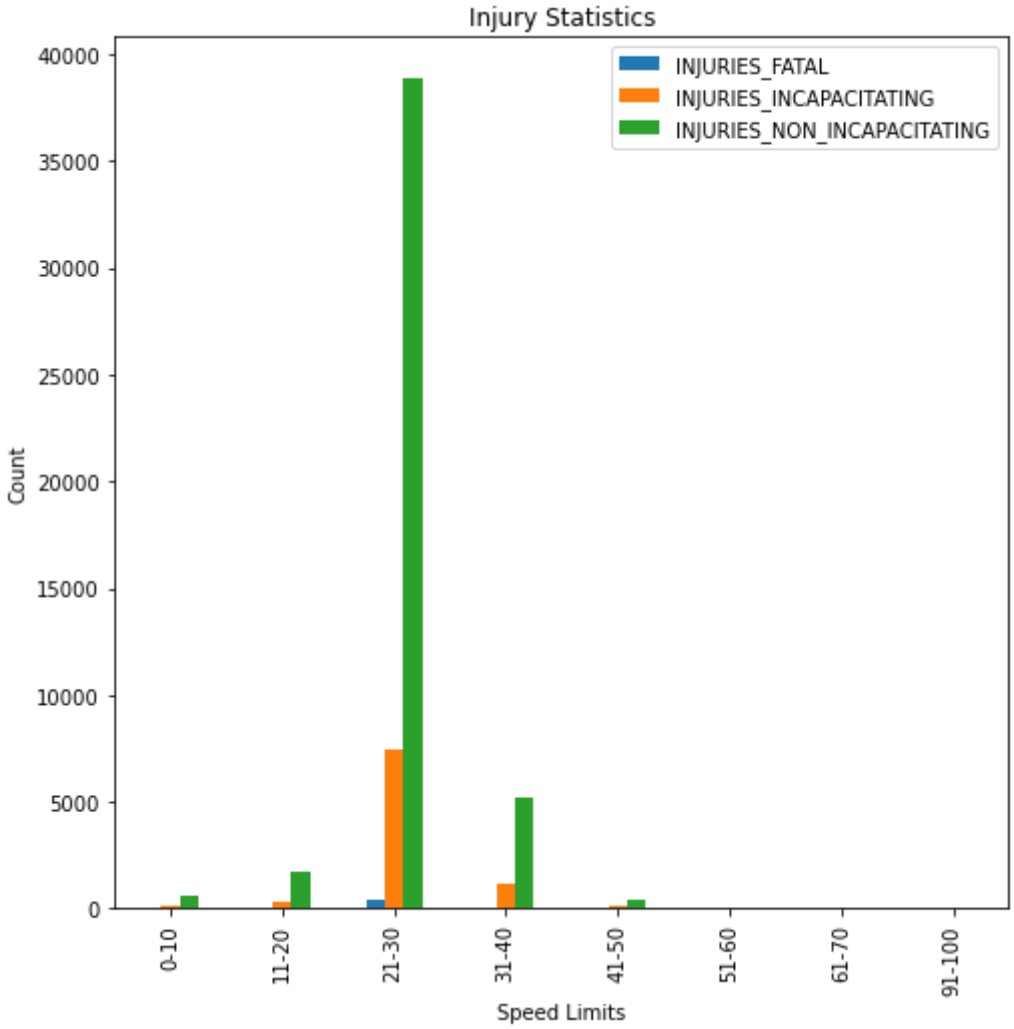
```
From 2013 to 2018 there was a significant growth reaching the peak
From 2018 to 2019 there was a slight dip
From 2019 to 2021 the there was a significant dip reaching as low as of s
omething similar to 2015
```



5. Investigate number and type of injuries based on the speed limit.

In [130]:
```
a = df.groupby('POSTED_SPEED_LIMIT')[['INJURIES_FATAL', 'INJURIES_INCAPACIT
plt.title('Injury Statistics')
plt.xlabel('Speed Limits')
plt.ylabel('Count')
```

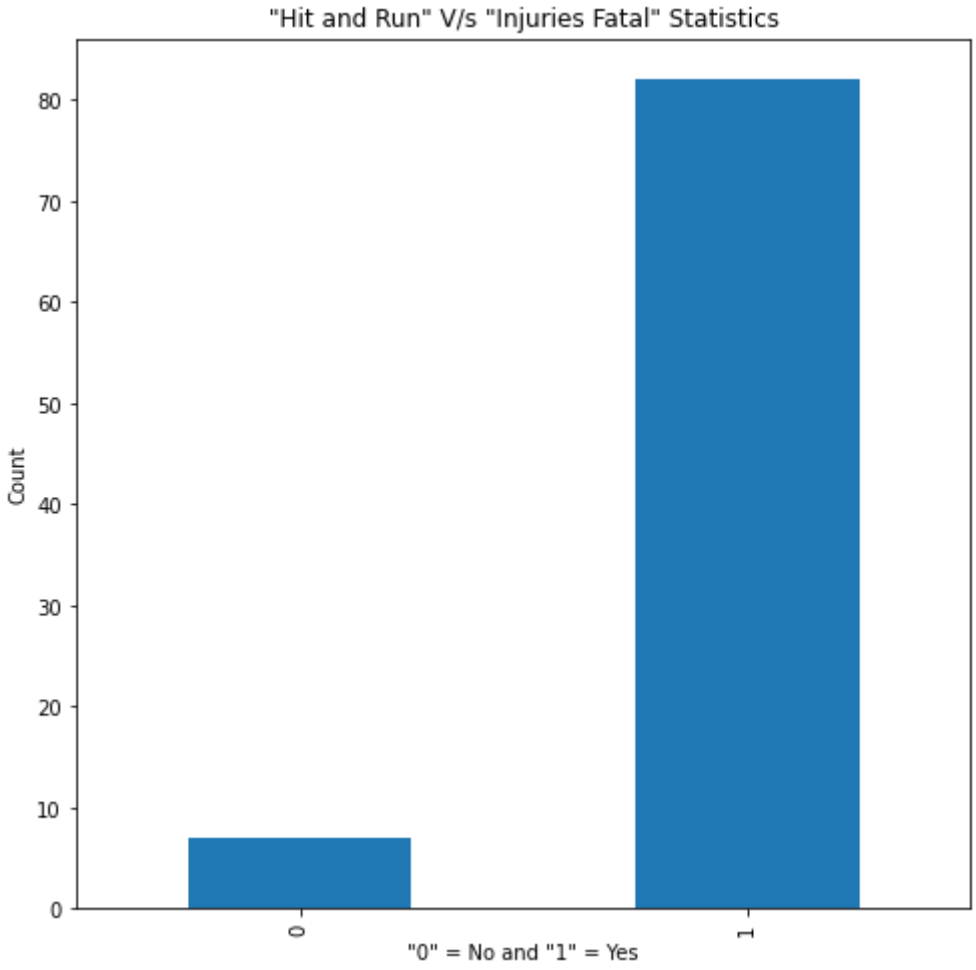
Out[130]:
```
Text(0, 0.5, 'Count')
```



6. Is there a relationship between hit and run crashes and number of fatal injuries?

In [131]:
```
b = df.groupby('HIT_AND_RUN_I')['INJURIES_FATAL'].sum()
b.plot(kind='bar')
plt.title('"Hit and Run" V/s "Injuries Fatal" Statistics')
plt.xlabel('"0" = No and "1" = Yes')
plt.ylabel('Count')
b
```

Out[131]:
```
HIT_AND_RUN_I
0     7.0
1    82.0
Name: INJURIES_FATAL, dtype: float64
```

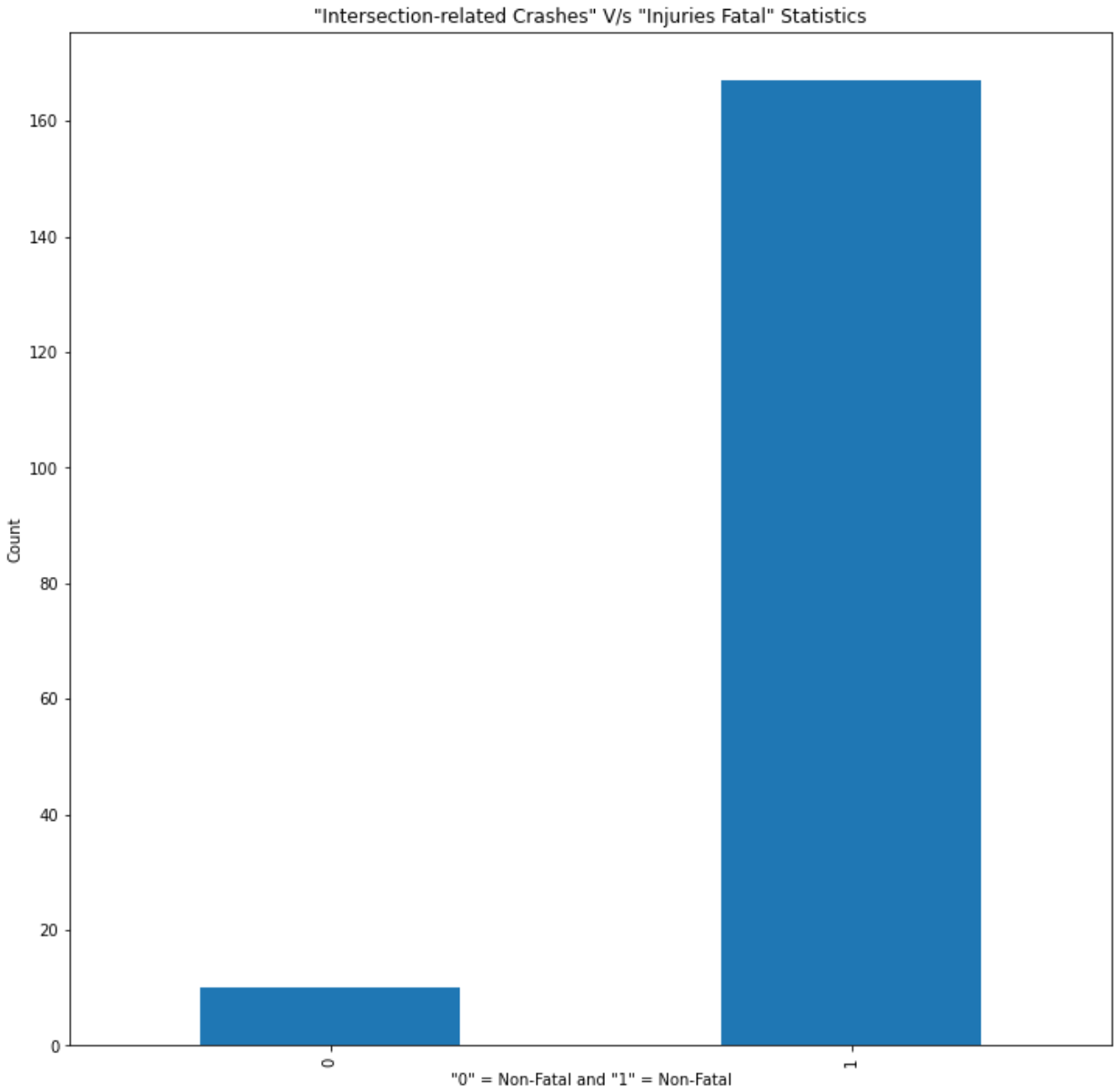


The Number of injuries in Hit and Run is more than as compared to that in non Hit and Run as we can visualize from above

7. Do intersection-related crashes result in more fatal injuries?

In [135]:
```
c = df.groupby('INTERSECTION_RELATED_I')['INJURIES_FATAL'].sum()
c.plot(kind='bar')
plt.title('"Intersection-related Crashes" V/s "Injuries Fatal" Statistics')
plt.xlabel('"0" = Non-Fatal and "1" = Non-Fatal')
plt.ylabel('Count')
c
```

Out[135]:
```
INTERSECTION_RELATED_I
0     10.0
1    167.0
Name: INJURIES_FATAL, dtype: float64
```



Intersection related crashes have more fatal deaths than non fatal deaths which we cn visualize from the graph above

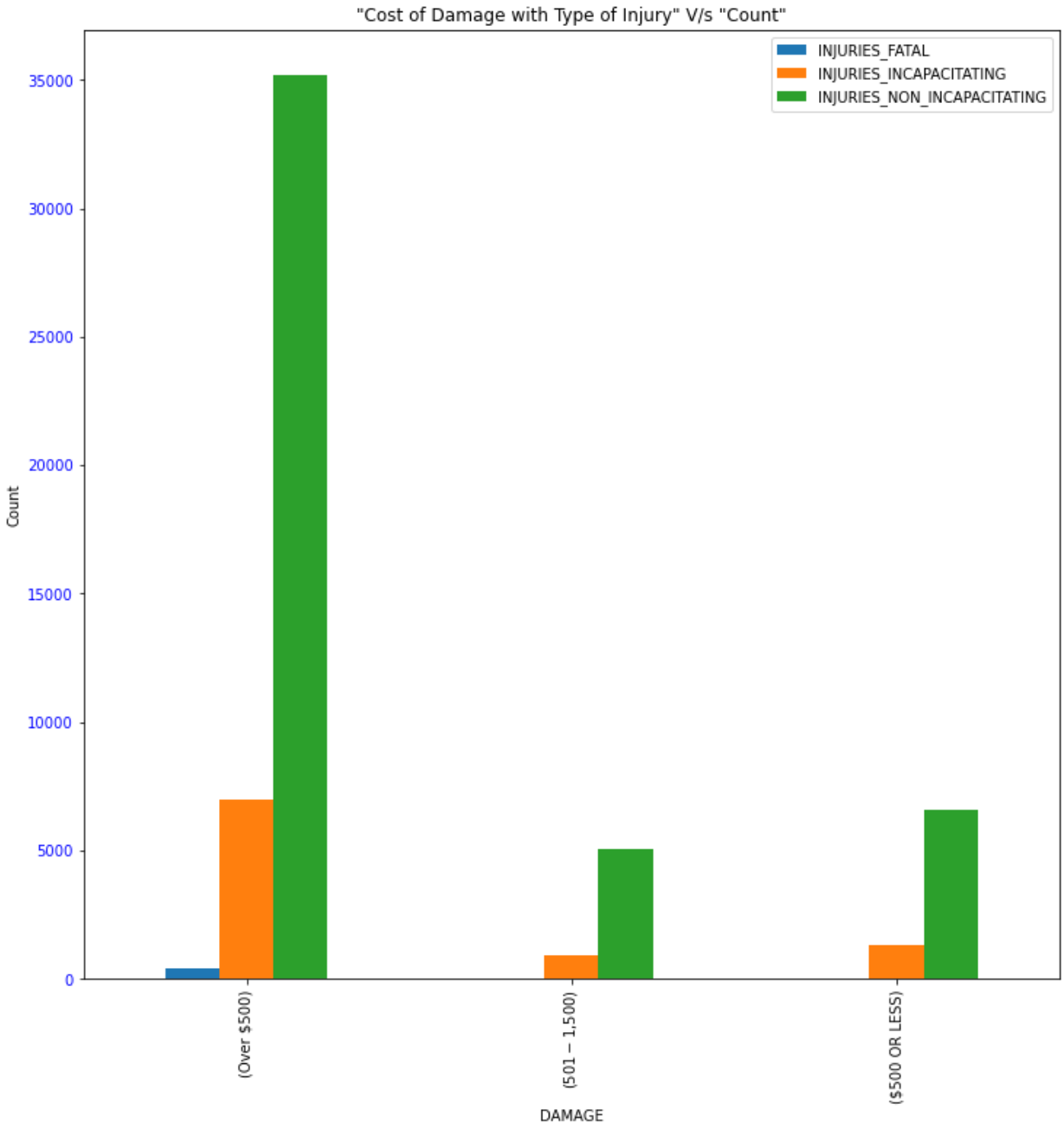
8. Come up with at least two more interesting insights and visualize them. (Suggestions: Season/weather/road condition and fatalities, or hit and run, having right of the way ... }) You must have at least one visualization for any questions/insight you are investigating.

In [136]:
```
# 8.1
# Cost of Damage with Type of Injury V/s Count
d = df.groupby('DAMAGE')[['INJURIES_FATAL', 'INJURIES_INCAPACITATING', 'INJ
plt.title('"Cost of Damage with Type of Injury" V/s "Count"')
rcParams['figure.figsize'] = 12, 12

bars = ('(Over $500)', '($501 - $1,500)', '($500 OR LESS)')
x_pos = np.arange(len(bars))
plt.xticks(x_pos, bars, color='Black')
plt.yticks(color='blue')

plt.ylabel('Count')
```

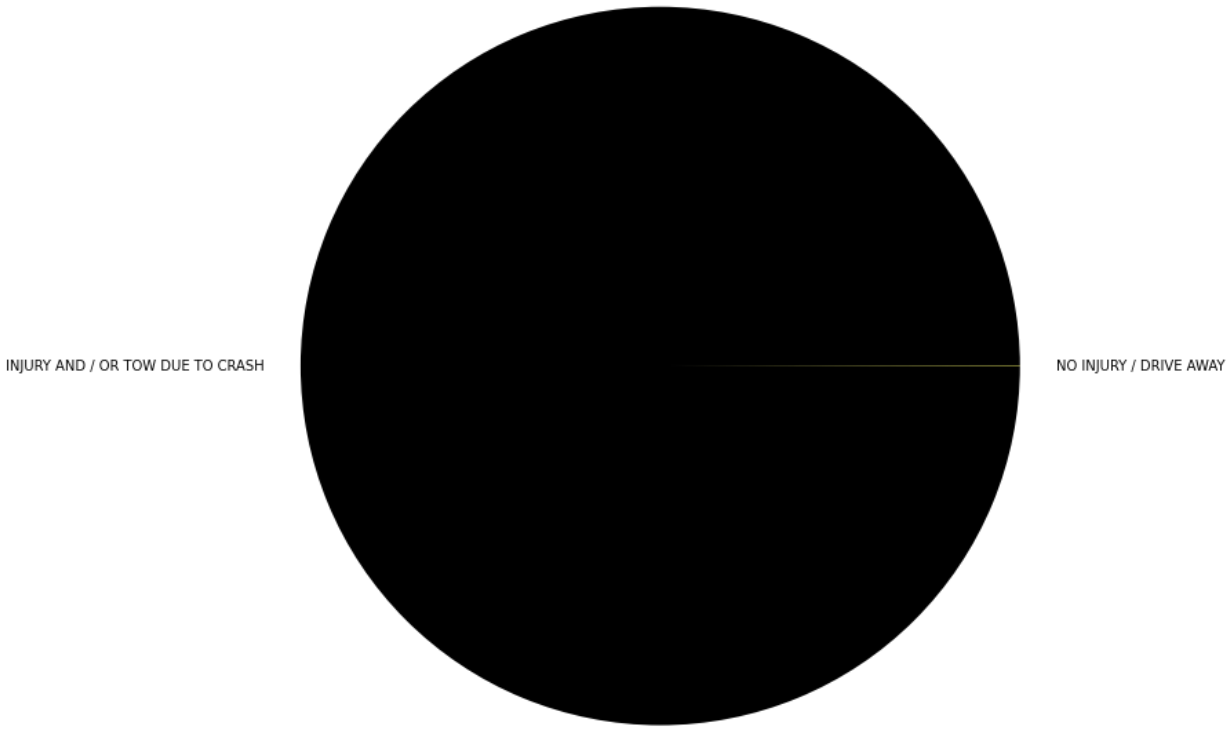
Out[136]: Text(0, 0.5, 'Count')



In [137]:
```
# 8.2
# Type of Crash with No. of Injury V/s Count

e = df.groupby('CRASH_TYPE')['INJURIES_TOTAL'].sum()
colors = ['#000000','yellow']
e.plot.pie(colors=colors)
plt.ylabel('')
e
```

Out[137]:
```
CRASH_TYPE
INJURY AND / OR TOW DUE TO CRASH    84603.0
NO INJURY / DRIVE AWAY                 13.0
Name: INJURIES_TOTAL, dtype: float64
```



The No Injury is really low value even less than 1% so we can see a small section in the pie chart