IDS Project:

Dataset chosen: US Accidents in four years

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing
from pandas import Series,DataFrame
from scipy import stats
from scipy.stats import chi2_contingency
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
df=pd.read_csv("US_Accidents_May19.csv")
df.head()
```

Out[2]:

	ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Li
0	A-1	MapQuest	201.0	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN	Na
1	A-2	MapQuest	201.0	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN	Na
2	A-3	MapQuest	201.0	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN	Na
3	A-4	MapQuest	201.0	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN	Na
4	A-5	MapQuest	201.0	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN	Na

5 rows × 49 columns

Cleaning Dataset

Cleaning numerical data

In [3]:
```
m = df['Severity'].mean()
df['Severity'].fillna(m, inplace=True)
m = df['Start_Lat'].mean()
df['Start_Lat'].fillna(m, inplace=True)
m = df['Start_Lng'].mean()
df['Start_Lng'].fillna(m, inplace=True)
m = df['Distance(mi)'].mean()
df['Distance(mi)'].fillna(m, inplace=True)
m = df['Temperature(F)'].mean()
df['Temperature(F)'].fillna(m, inplace=True)
m = df['Wind_Chill(F)'].mean()
df['Wind_Chill(F)'].fillna(m, inplace=True)
m = df['Humidity(%)'].mean()
df['Humidity(%)'].fillna(m, inplace=True)
m = df['Pressure(in)'].mean()
df['Pressure(in)'].fillna(m, inplace=True)
m = df['Visibility(mi)'].mean()
df['Visibility(mi)'].fillna(m, inplace=True)
m = df['Wind_Speed(mph)'].mean()
df['Wind_Speed(mph)'].fillna(m, inplace=True)
m = df['Precipitation(in)'].mean()
df['Precipitation(in)'].fillna(m, inplace=True)
```

Cleaning categorical data

In [4]:
```
d = pd.Series(df['Start_Time'])
d.fillna(method='ffill')
d = pd.Series(df['End_Time'])
d.fillna(method='ffill')
d = pd.Series(df['Description'])
d.fillna(method='ffill')
try:
    d = pd.Series(df['Number'])
    d.fillna(method='ffill')
except:
    d = pd.Series(df['Number'])
    d.fillna(method='bfill')
d = pd.Series(df['Street'])
d.fillna(method='ffill')
d = pd.Series(df['Side'])
d.fillna(method='ffill')
d = pd.Series(df['City'])
d.fillna(method='ffill')
d = pd.Series(df['County'])
d.fillna(method='ffill')
d = pd.Series(df['State'])
d.fillna(method='ffill')
d = pd.Series(df['Zipcode'])
d.fillna(method='ffill')
d = pd.Series(df['Country'])
d.fillna(method='ffill')
d = pd.Series(df['Timezone'])
d.fillna(method='ffill')
d = pd.Series(df['Airport_Code'])
d.fillna(method='ffill')
d = pd.Series(df['Weather_Timestamp'])
d.fillna(method='ffill')
d = pd.Series(df['Wind_Direction'])
d.fillna(method='ffill')
d = pd.Series(df['Weather_Condition'])
d.fillna(method='ffill')
d = pd.Series(df['Amenity'])
d.fillna(method='ffill')
d = pd.Series(df['Bump'])
d.fillna(method='ffill')
d = pd.Series(df['Crossing'])
d.fillna(method='ffill')
d = pd.Series(df['Give_Way'])
d.fillna(method='ffill')
d = pd.Series(df['Junction'])
d.fillna(method='ffill')
d = pd.Series(df['No_Exit'])
d.fillna(method='ffill')
d = pd.Series(df['Railway'])
d.fillna(method='ffill')
d = pd.Series(df['Roundabout'])
d.fillna(method='ffill')
d = pd.Series(df['Station'])
d.fillna(method='ffill')
d = pd.Series(df['Stop'])
d.fillna(method='ffill')
```

```
d = pd.Series(df['Traffic_Calming'])
d.fillna(method='ffill')
d = pd.Series(df['Traffic_Signal'])
d.fillna(method='ffill')
d = pd.Series(df['Turning_Loop'])
d.fillna(method='ffill')
d = pd.Series(df['Sunrise_Sunset'])
d.fillna(method='ffill')
d = pd.Series(df['Civil_Twilight'])
d.fillna(method='ffill')
d = pd.Series(df['Nautical_Twilight'])
d.fillna(method='ffill')
d = pd.Series(df['Astronomical_Twilight'])
a=d.fillna(method='ffill')
```

Normalizing and Standardizing data

Normalizing

In [5]:
```
normalized_X = preprocessing.normalize(df[['Severity','Start_Lat','Start_Lng',
'Distance(mi)','Temperature(F)','Wind_Chill(F)','Humidity(%)','Pressure(in)',
'Visibility(mi)','Wind_Speed(mph)','Precipitation(in)']])
min_max_scaler = preprocessing.MinMaxScaler()
w=df[['Severity','Start_Lat','Start_Lng','Distance(mi)','Temperature(F)','Wind
_Chill(F)','Humidity(%)','Pressure(in)','Visibility(mi)','Wind_Speed(mph)','Pr
ecipitation(in)']]
np_scaled = min_max_scaler.fit_transform(w)
q=['Severity','Start_Lat','Start_Lng','Distance(mi)','Temperature(F)','Wind_Ch
ill(F)','Humidity(%)','Pressure(in)','Visibility(mi)','Wind_Speed(mph)','Preci
pitation(in)']
df_normalized = pd.DataFrame(np_scaled, columns = q)
df_normalized
```

```
C:\Users\THUSHAR KARANTH\Anaconda3\lib\site-packages\sklearn\preprocessing\da
ta.py:323: DataConversionWarning: Data with input dtype int64, float64 were a
ll converted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
```

Out[5]:

	Severity	Start_Lat	Start_Lng	Distance(mi)	Temperature(F)	Wind_Chill(F)	Humidity(%)
0	0.75	0.626058	0.705349	0.000030	0.461755	0.827561	0.906250
1	0.50	0.628633	0.726694	0.000030	0.465781	0.827561	1.000000
2	0.50	0.593230	0.705803	0.000030	0.458132	0.892889	1.000000
3	0.75	0.621252	0.702796	0.000030	0.454509	0.872187	0.958333
4	0.50	0.616342	0.703095	0.000030	0.458132	0.892889	0.885417
5	0.75	0.635695	0.725059	0.000030	0.465781	0.912691	0.968750
6	0.50	0.621683	0.702362	0.000000	0.450081	0.872187	1.000000
7	0.75	0.622179	0.702981	0.000030	0.450081	0.872187	1.000000
8	0.50	0.622493	0.703380	0.000000	0.447262	0.827561	0.989583
9	0.75	0.635695	0.725059	0.000030	0.463768	0.897390	1.000000
10	0.75	0.629646	0.721684	0.000030	0.456522	0.869487	0.927083
11	0.75	0.628823	0.726699	0.000030	0.463768	0.897390	1.000000
12	0.50	0.620838	0.703763	0.000000	0.449275	0.827561	1.000000
13	0.50	0.623013	0.702170	0.000030	0.458132	0.873087	0.885417
14	0.50	0.630433	0.725262	0.000030	0.463768	0.897390	1.000000
15	0.50	0.621176	0.703407	0.000030	0.449275	0.827561	1.000000
16	0.50	0.621276	0.702475	0.000030	0.456522	0.827561	0.989583
17	0.50	0.621433	0.702198	0.000000	0.458132	0.873087	0.885417
18	0.50	0.620962	0.703169	0.000030	0.463768	0.882088	0.927083
19	0.50	0.623010	0.702120	0.000030	0.458132	0.865887	0.885417
20	0.50	0.633727	0.725805	0.000000	0.449275	0.859586	1.000000
21	0.50	0.622300	0.702464	0.000000	0.458132	0.865887	0.885417
22	0.50	0.616362	0.702438	0.000030	0.454509	0.850585	0.885417
23	0.75	0.632539	0.723847	0.000030	0.462158	0.884788	0.958333
24	0.50	0.621810	0.701863	0.000030	0.465781	0.871287	0.750000
25	0.50	0.638046	0.729988	0.003956	0.465781	0.902790	1.000000
26	0.50	0.620658	0.703594	0.000000	0.458132	0.854185	0.854167
27	0.50	0.622380	0.702884	0.000000	0.473833	0.889289	0.687500
28	0.50	0.622954	0.702213	0.000030	0.473833	0.903690	0.635417
29	0.50	0.621701	0.703175	0.000000	0.471014	0.916292	0.739583
...	...	...	...	...	...	...	...
2243909	0.50	0.376925	0.117000	0.002413	0.671900	0.827561	0.395833
2243910	0.50	0.391557	0.124875	0.001067	0.727456	0.827561	0.166667
2243911	0.50	0.387492	0.110183	0.000833	0.671900	0.827561	0.354167
2243912	0.50	0.488952	0.055378	0.000968	0.591385	0.827561	0.614583

	Severity	Start_Lat	Start_Lng	Distance(mi)	Temperature(F)	Wind_Chill(F)	Humidity(%)
2243913	0.50	0.380244	0.114548	0.002422	0.664654	0.827561	0.395833
2243914	0.50	0.377951	0.117261	0.002338	0.695652	0.827561	0.291667
2243915	0.50	0.390230	0.115462	0.003375	0.559712	0.827561	0.645079
2243916	0.75	0.380471	0.114328	0.001918	0.695652	0.827561	0.291667
2243917	0.75	0.338304	0.128497	0.000381	0.683172	0.827561	0.354167
2243918	0.75	0.338485	0.128443	0.000665	0.683172	0.827561	0.354167
2243919	0.50	0.379493	0.110200	0.000593	0.623188	0.827561	0.552083
2243920	0.50	0.330703	0.131454	0.004319	0.643317	0.827561	0.541667
2243921	0.50	0.331703	0.131118	0.004616	0.610709	0.827561	0.781250
2243922	0.50	0.498473	0.084293	0.002410	0.715781	0.827561	0.145833
2243923	0.50	0.438680	0.068880	0.001097	0.599436	0.827561	0.572917
2243924	0.50	0.431020	0.078358	0.005713	0.666667	0.827561	0.302083
2243925	0.50	0.390864	0.112562	0.000444	0.667472	0.827561	0.375000
2243926	0.75	0.392405	0.108581	0.001421	0.708132	0.827561	0.197917
2243927	0.50	0.391532	0.115393	0.000558	0.688406	0.827561	0.291667
2243928	0.50	0.380092	0.114746	0.002110	0.679549	0.827561	0.322917
2243929	0.50	0.372232	0.118640	0.001679	0.635668	0.827561	0.541667
2243930	0.75	0.370095	0.120322	0.004589	0.635668	0.827561	0.541667
2243931	0.50	0.375164	0.118318	0.001598	0.635668	0.827561	0.541667
2243932	0.50	0.380623	0.119465	0.002254	0.667472	0.827561	0.364583
2243933	0.75	0.391033	0.109233	0.001592	0.683172	0.827561	0.302083
2243934	0.50	0.388241	0.111453	0.000635	0.643317	0.827561	0.468750
2243935	0.50	0.387896	0.111441	0.001747	0.643317	0.827561	0.468750
2243936	0.75	0.387287	0.130201	0.003645	0.710145	0.827561	0.156250
2243937	1.00	0.397357	0.104599	0.005968	0.695652	0.827561	0.187500
2243938	1.00	0.399254	0.106445	0.003834	0.695652	0.827561	0.250000

2243939 rows × 11 columns

Standardizing

In [6]:
```
standardized_X = preprocessing.scale(w)
da=pd.DataFrame(standardized_X)
da.columns=q
da
```

```
C:\Users\THUSHAR KARANTH\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
DataConversionWarning: Data with input dtype int64, float64 were all converte
d to float64 by the scale function.
  """Entry point for launching an IPython kernel.
```

Out[6]:

	Severity	Start_Lat	Start_Lng	Distance(mi)	Temperature(F)	Wind_Chill(F)	Humidity
0	1.124826	0.685992	0.631602	-0.181363	-1.288887e+00	-2.038114e-13	1.134212e
1	-0.697322	0.698679	0.703411	-0.181363	-1.235917e+00	-2.038114e-13	1.541349e
2	-0.697322	0.524258	0.633130	-0.181363	-1.336560e+00	1.289077e+00	1.541349e
3	1.124826	0.662318	0.623011	-0.181363	-1.384232e+00	8.805751e-01	1.360400e
4	-0.697322	0.638124	0.624019	-0.181363	-1.336560e+00	1.289077e+00	1.043737e
5	1.124826	0.733472	0.697912	-0.181363	-1.235917e+00	1.679817e+00	1.405637e
6	-0.697322	0.664439	0.621553	-0.187889	-1.442499e+00	8.805751e-01	1.541349e
7	1.124826	0.666881	0.623636	-0.181363	-1.442499e+00	8.805751e-01	1.541349e
8	-0.697322	0.668430	0.624976	-0.187889	-1.479578e+00	-2.038114e-13	1.496112e
9	1.124826	0.733472	0.697912	-0.181363	-1.262402e+00	1.377881e+00	1.541349e
10	1.124826	0.703671	0.686557	-0.181363	-1.357748e+00	8.272923e-01	1.224687e
11	1.124826	0.699617	0.703427	-0.181363	-1.262402e+00	1.377881e+00	1.541349e
12	-0.697322	0.660277	0.626267	-0.187889	-1.453093e+00	-2.038114e-13	1.541349e
13	-0.697322	0.670991	0.620908	-0.181363	-1.336560e+00	8.983361e-01	1.043737e
14	-0.697322	0.707548	0.698595	-0.181363	-1.262402e+00	1.377881e+00	1.541349e
15	-0.697322	0.661942	0.625069	-0.181363	-1.453093e+00	-2.038114e-13	1.541349e
16	-0.697322	0.662434	0.621934	-0.181363	-1.357748e+00	-2.038114e-13	1.496112e
17	-0.697322	0.663209	0.621001	-0.187889	-1.336560e+00	8.983361e-01	1.043737e
18	-0.697322	0.660889	0.624266	-0.181363	-1.262402e+00	1.075945e+00	1.224687e
19	-0.697322	0.670979	0.620737	-0.181363	-1.336560e+00	7.562486e-01	1.043737e
20	-0.697322	0.723776	0.700419	-0.187889	-1.453093e+00	6.319221e-01	1.541349e
21	-0.697322	0.667479	0.621894	-0.187889	-1.336560e+00	7.562486e-01	1.043737e
22	-0.697322	0.638226	0.621808	-0.181363	-1.384232e+00	4.543128e-01	1.043737e
23	1.124826	0.717923	0.693835	-0.181363	-1.283590e+00	1.129228e+00	1.360400e
24	-0.697322	0.665066	0.619874	-0.181363	-1.235917e+00	8.628142e-01	4.556505e
25	-0.697322	0.745054	0.714492	0.673539	-1.235917e+00	1.484447e+00	1.541349e
26	-0.697322	0.659387	0.625698	-0.187889	-1.336560e+00	5.253565e-01	9.080250e
27	-0.697322	0.667874	0.623307	-0.187889	-1.129977e+00	1.218033e+00	1.842258e
28	-0.697322	0.670701	0.621050	-0.181363	-1.129977e+00	1.502208e+00	-4.196145e
29	-0.697322	0.664529	0.624288	-0.187889	-1.167056e+00	1.750861e+00	4.104131e
...	...	...	...	...	...	...	
2243909	-0.697322	-0.541420	-1.347765	0.337451	1.476142e+00	-2.038114e-13	-1.082423e
2243910	-0.697322	-0.469331	-1.321272	0.044436	2.207126e+00	-2.038114e-13	-2.077647e
2243911	-0.697322	-0.489362	-1.370699	-0.006467	1.476142e+00	-2.038114e-13	-1.263373e
2243912	-0.697322	0.010507	-1.555081	0.022900	4.167438e-01	-2.038114e-13	-1.324363e

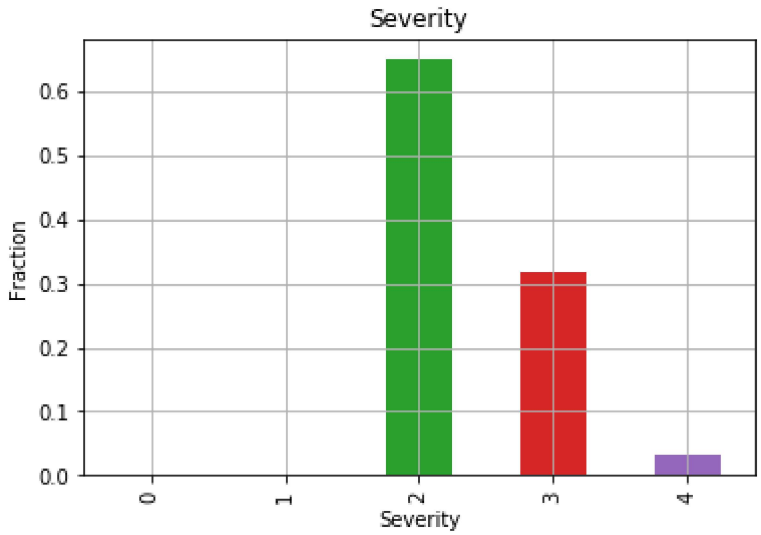
	Severity	Start_Lat	Start_Lng	Distance(mi)	Temperature(F)	Wind_Chill(F)	Humidity
2243913	-0.697322	-0.525069	-1.356016	0.339409	1.380796e+00	-2.038114e-13	-1.082423e
2243914	-0.697322	-0.536366	-1.346887	0.321137	1.788664e+00	-2.038114e-13	-1.534797e
2243915	-0.697322	-0.475871	-1.352939	0.546935	-9.371706e-14	-2.038114e-13	-1.285726e
2243916	1.124826	-0.523952	-1.356755	0.229773	1.788664e+00	-2.038114e-13	-1.534797e
2243917	1.124826	-0.731697	-1.309088	-0.105009	1.624457e+00	-2.038114e-13	-1.263373e
2243918	1.124826	-0.730805	-1.309267	-0.043012	1.624457e+00	-2.038114e-13	-1.263373e
2243919	-0.697322	-0.528767	-1.370641	-0.058675	8.352060e-01	-2.038114e-13	-4.038611e
2243920	-0.697322	-0.769146	-1.299138	0.752503	1.100055e+00	-2.038114e-13	-4.490985e
2243921	-0.697322	-0.764217	-1.300269	0.817110	6.709993e-01	-2.038114e-13	5.913629e
2243922	-0.697322	0.057416	-1.457802	0.336799	2.053514e+00	-2.038114e-13	-2.168122e
2243923	-0.697322	-0.237173	-1.509655	0.050962	5.226836e-01	-2.038114e-13	-3.133862e
2243924	-0.697322	-0.274908	-1.477767	1.055960	1.407281e+00	-2.038114e-13	-1.489560e
2243925	-0.697322	-0.472747	-1.362696	-0.091304	1.417875e+00	-2.038114e-13	-1.172898e
2243926	1.124826	-0.465155	-1.376089	0.121442	1.952871e+00	-2.038114e-13	-1.941934e
2243927	-0.697322	-0.469456	-1.353173	-0.066506	1.693318e+00	-2.038114e-13	-1.534797e
2243928	-0.697322	-0.525817	-1.355348	0.271539	1.576784e+00	-2.038114e-13	-1.399085e
2243929	-0.697322	-0.564540	-1.342249	0.177565	9.994126e-01	-2.038114e-13	-4.490985e
2243930	1.124826	-0.575073	-1.336589	0.811236	9.994126e-01	-2.038114e-13	-4.490985e
2243931	-0.697322	-0.550097	-1.343333	0.159945	9.994126e-01	-2.038114e-13	-4.490985e
2243932	-0.697322	-0.523202	-1.339474	0.302864	1.417875e+00	-2.038114e-13	-1.218135e
2243933	1.124826	-0.471914	-1.373897	0.158640	1.624457e+00	-2.038114e-13	-1.489560e
2243934	-0.697322	-0.485670	-1.366426	-0.049538	1.100055e+00	-2.038114e-13	-7.657607e
2243935	-0.697322	-0.487368	-1.366469	0.192575	1.100055e+00	-2.038114e-13	-7.657607e
2243936	1.124826	-0.490371	-1.303356	0.605669	1.979356e+00	-2.038114e-13	-2.122884e
2243937	2.946974	-0.440759	-1.389487	1.111431	1.788664e+00	-2.038114e-13	-1.987172e
2243938	2.946974	-0.431413	-1.383277	0.646782	1.788664e+00	-2.038114e-13	-1.715747e

2243939 rows × 11 columns

Plotting graphs and making inferences

In [7]:
```
import datetime
print('There are {} accidents in the data'.format(len(df)))
df.Severity.value_counts(normalize=True).sort_index().plot.bar()
plt.grid()
plt.title('Severity')
plt.xlabel('Severity')
plt.ylabel('Fraction');
```

```
There are 2243939 accidents in the data
```



In [8]:
```
bool_cols = [col for col in df.columns if df[col].dtype ==np.dtype('bool')]
booldf = df[bool_cols]
not_one_hot = booldf[booldf.sum(axis=1) > 1]
print("There are {} non one hot metadata rows, which are {:.1f}% of the data".
format(len(not_one_hot),100*len(not_one_hot)/len(df)))
```

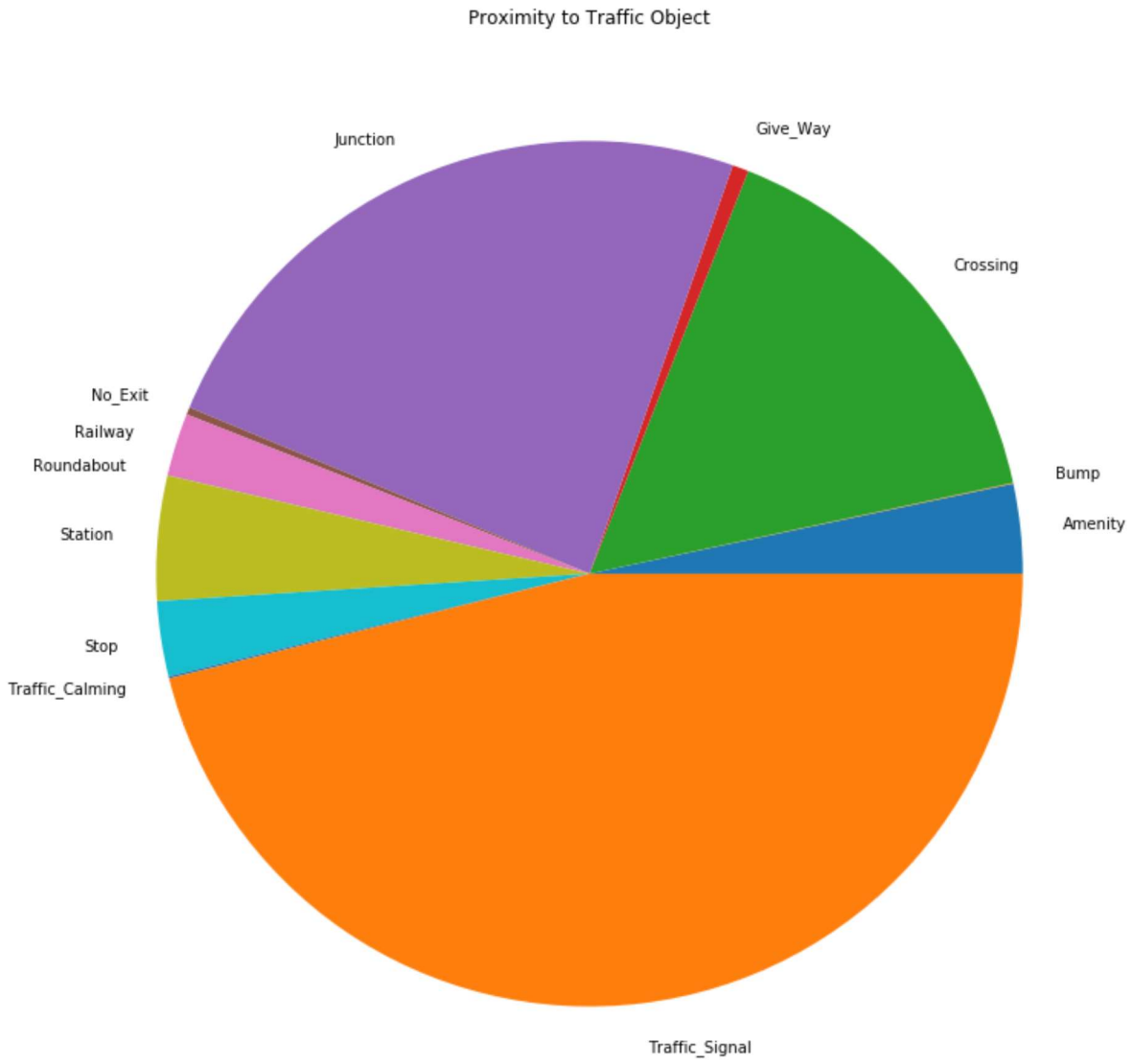
```
There are 134798 non one hot metadata rows, which are 6.0% of the data
```

In [9]:
```
bools = booldf.sum(axis=0)
```

In [10]:
```
bools
```

Out[10]:
```
Amenity             25977
Bump                  239
Crossing           121783
Give_Way             4724
Junction           187365
No_Exit              2166
Railway             18198
Roundabout            128
Station             36225
Stop                21771
Traffic_Calming       618
Traffic_Signal     358648
Turning_Loop            0
dtype: int64
```

In [11]:
```
bools.plot.pie(figsize=(13,13))
plt.ylabel('')
plt.title('Proximity to Traffic Object');
```

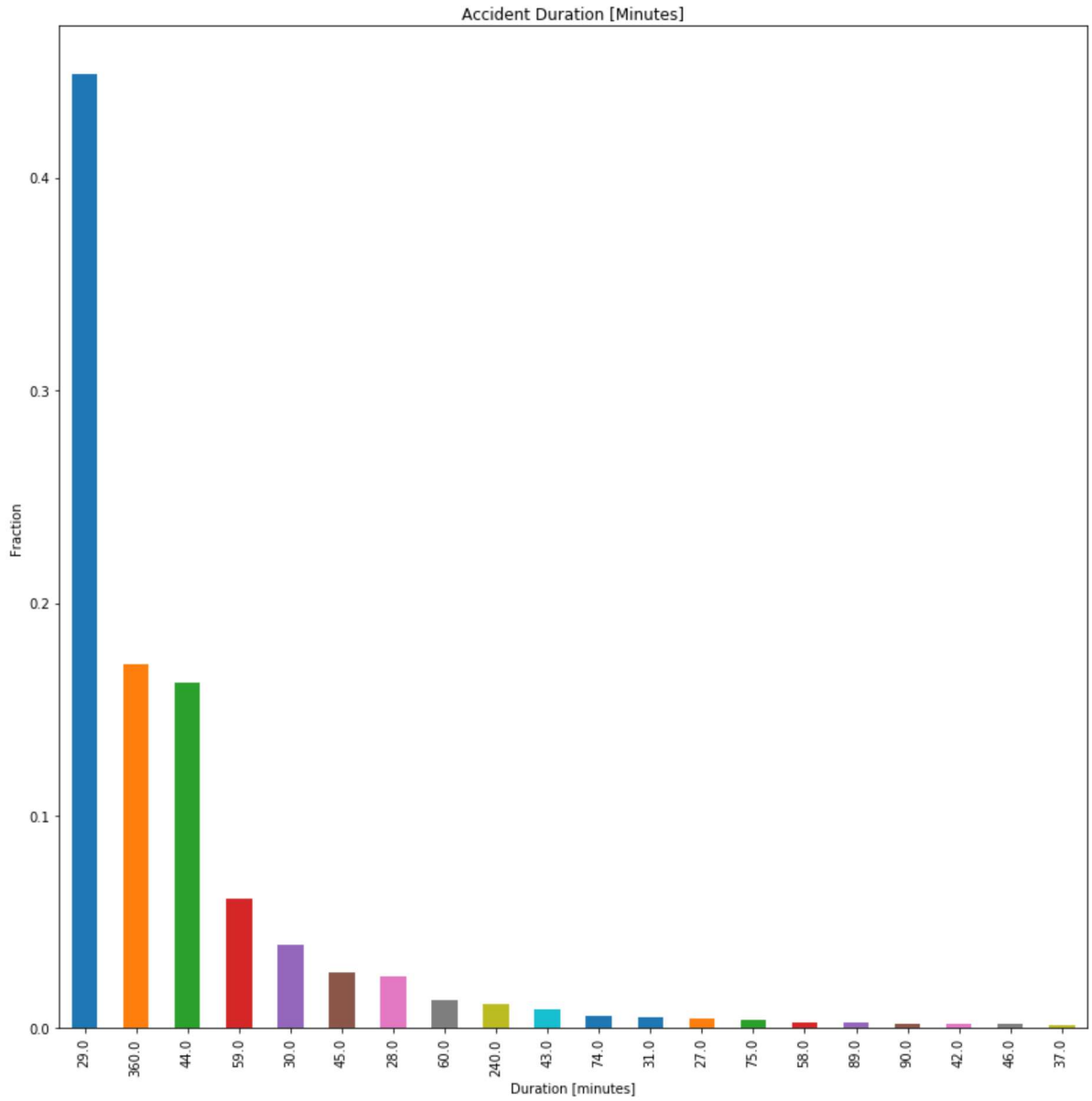


In [12]:
```
st = pd.to_datetime(df.Start_Time, format='%Y-%m-%d %H:%M:%S')
end = pd.to_datetime(df.End_Time, format='%Y-%m-%d %H:%M:%S')
```

In [13]:
```
diff = (end-st)
top20 = diff.astype('timedelta64[m]').value_counts().nlargest(20)
print('top 20 accident durations correspond to {:.1f}% of the data'.format(top
20.sum()*100/len(diff)))
(top20/top20.sum()).plot.bar(figsize=(14,14))
plt.title('Accident Duration [Minutes]')
plt.xlabel('Duration [minutes]')
plt.ylabel('Fraction')
```

```
top 20 accident durations correspond to 96.1% of the data
```

Out[13]:
```
Text(0, 0.5, 'Fraction')
```



Most common durations are 'Below half an hour', 'exactly 1h', 'Below 3/4 of an hour', 'Below 1h' respectively. Hence they are probably approximate, and probably correspond to the time it took to resolve the accident rather than the accident itself

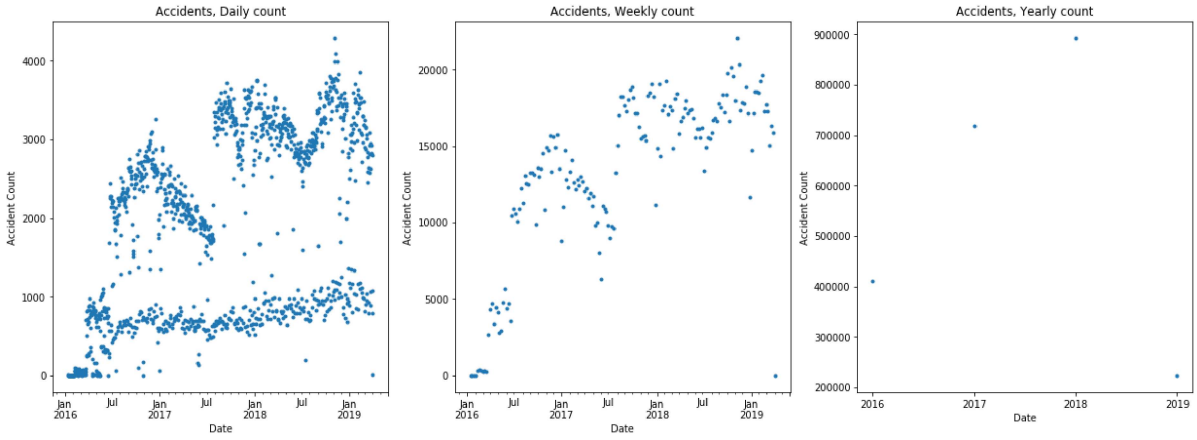
In [14]:
```
df['time'] = pd.to_datetime(df.Start_Time, format='%Y-%m-%d %H:%M:%S')
df = df.set_index('time')
df.head()
```

Out[14]:

time	ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat
2016-02-08 05:46:00	A-1	MapQuest	201.0	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN
2016-02-08 06:07:59	A-2	MapQuest	201.0	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN
2016-02-08 06:49:27	A-3	MapQuest	201.0	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN
2016-02-08 07:23:34	A-4	MapQuest	201.0	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN
2016-02-08 07:39:07	A-5	MapQuest	201.0	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN

5 rows × 49 columns

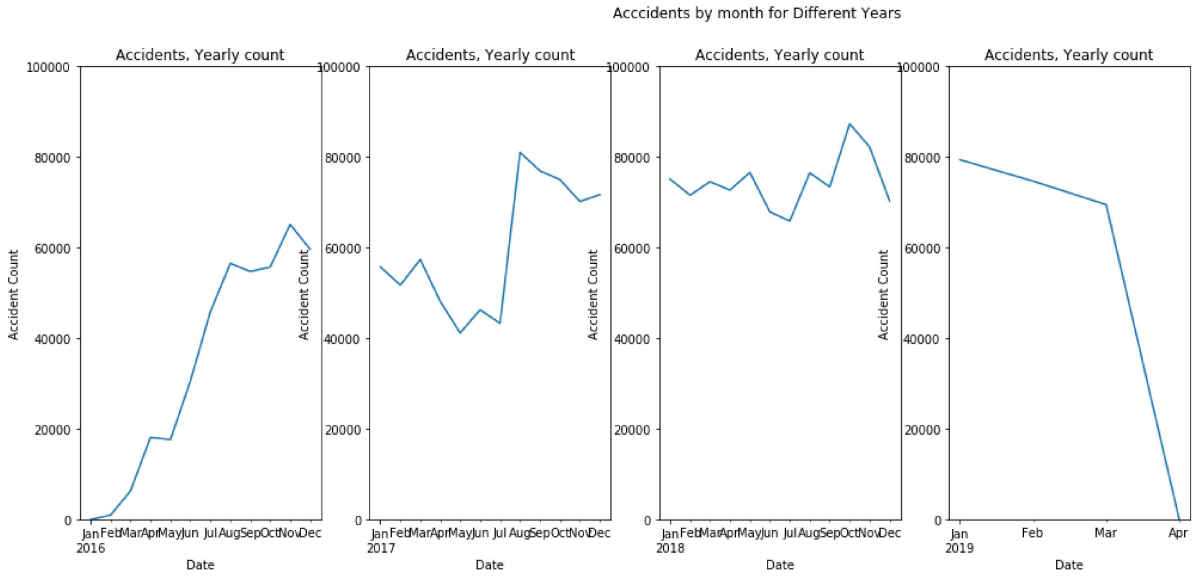
In [15]:
```
freq_text = {'D':'Daily','W':'Weekly','Y':'Yearly'}
plt.subplots(1,3,figsize=(21,7))
for i, (fr,text) in enumerate(freq_text.items(),1):
    plt.subplot(1,3,i)
    sample = df.ID['2016':].resample(fr).count()
    sample.plot(style='.')
    plt.title('Accidents, {} count'.format(text))
    plt.xlabel('Date')
    plt.ylabel('Accident Count');
```



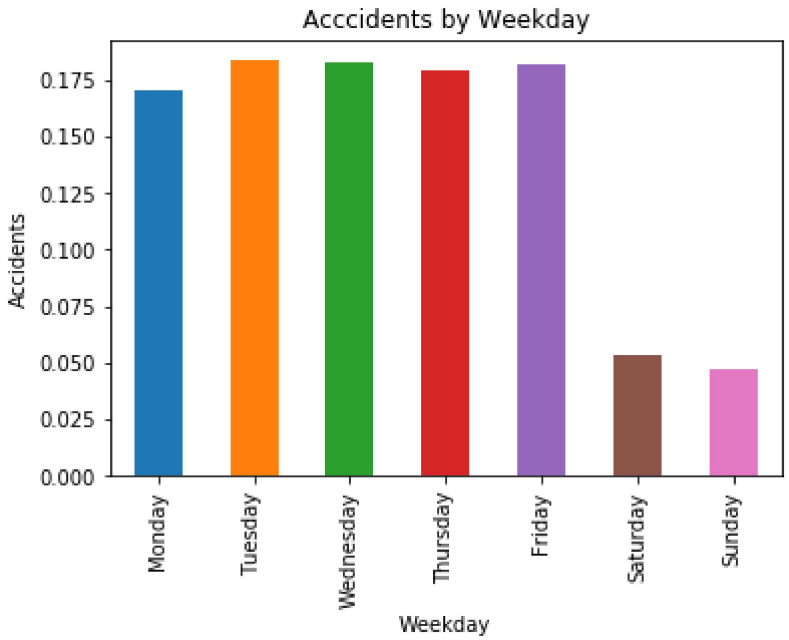
The are two populations for days (prob. weekday and weekend) as they merge when we look at weeks. There is some seasonal pattern, and the amount increases each year.

In [16]:
```
years = ['2016','2017','2018','2019']
fig, _ = plt.subplots(1,3,figsize=(21,7), sharex='all', sharey='all')

fig.suptitle('Acccidents by month for Different Years')
plt.xlabel('month')
plt.ylabel('Accidents')
for i, year in enumerate(years,1):
    plt.subplot(1,5,i)
    sample = df.loc[year].ID.resample('M').count()
    sample.plot()
    plt.ylim(0,100000)
    plt.title('Accidents, {} count'.format(text))
    plt.xlabel('Date')
    plt.ylabel('Accident Count');
```



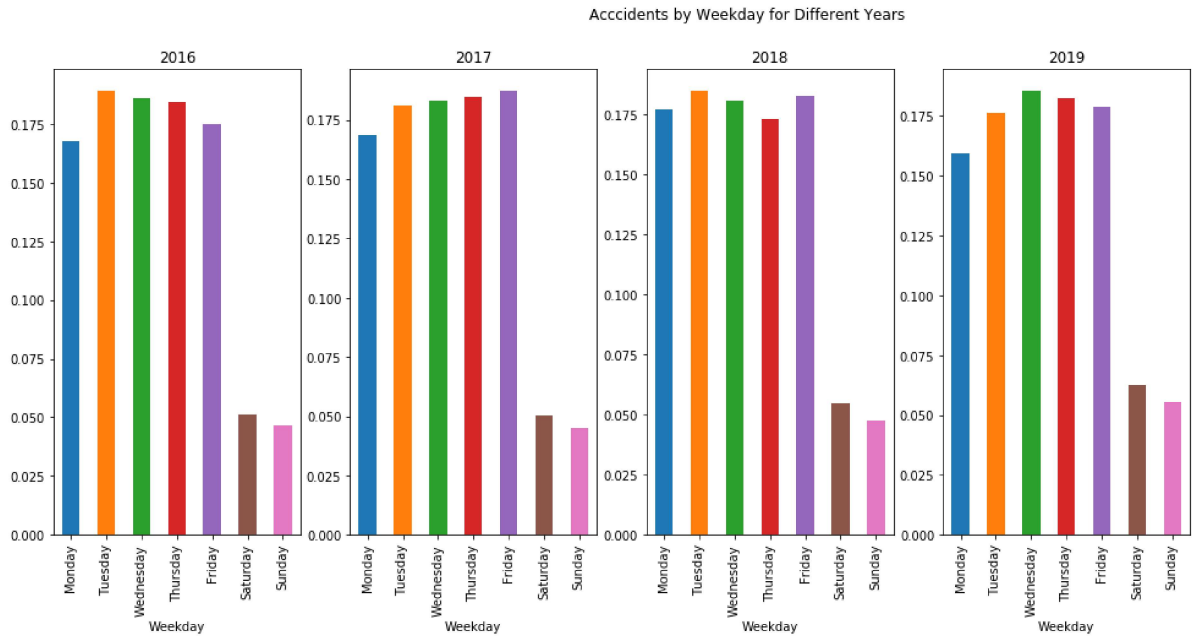
In [17]:
```
df['Weekday'] = df.index.weekday_name
weekday = df.groupby('Weekday').ID.count()
weekday = weekday/weekday.sum()
dayOfWeek=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sund
ay']
weekday[dayOfWeek].plot.bar()
plt.title('Acccidents by Weekday')
plt.xlabel('Weekday')
plt.ylabel('Accidents');
```



From the graph it is clear that the number of accidents occur in weekdays is more compared to weekends.

In [18]:
```
years = ['2016','2017','2018','2019']
fig, _ = plt.subplots(1,3,figsize=(21,7), sharex='all', sharey='all')

fig.suptitle('Acccidents by Weekday for Different Years')
plt.xlabel('Weekday')
plt.ylabel('Accidents')
for i, year in enumerate(years,1):
    weekday = df.loc[year].groupby('Weekday').ID.count()
    weekday = weekday/weekday.sum()
    dayOfWeek=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday',
'Sunday']
    plt.subplot(1,5,i)
    plt.title(year)
    weekday[dayOfWeek].plot.bar()
```



Here no phenomenon is observed.

Hypothesis testing

Using Chi squared test (taking random sample of 100 from the dataset)

In [19]:
```
q=df[['Severity','Start_Lat','Start_Lng','Distance(mi)','Temperature(F)','Wind
_Chill(F)','Humidity(%)','Pressure(in)','Visibility(mi)','Wind_Speed(mph)','Pr
ecipitation(in)']]
```

Assuming

H0:Temperature and Pressure are dependent.

H1:Temperature and Pressure are independent.

In [36]:
```
df1=df.sample(n=100)
tables=pd.crosstab(df1['Temperature(F)'],df1['Pressure(in)'])
chi2, p, dof, expected=chi2_contingency(tables.values)
print('Chi-square statistic %0.3f p_value %0.3f' %(chi2,p))
```

```
Chi-square statistic 3726.111 p_value 0.130
```

Here p>0.1 so we accept the H0 i.e., Pressure and Temperature are related.

Assuming

H0:Temperature and Humidity are dependent.

H1:Temperature and Humidity are independent.

In [24]:
```
df1=df.sample(n=100)
tables=pd.crosstab(df1['Temperature(F)'],df1['Humidity(%)'])
chi2, p, dof, expected=chi2_contingency(tables.values)
print('Chi-square statistic %0.3f p_value %0.3f' %(chi2,p))
```

```
Chi-square statistic 3674.167 p_value 0.034
```

Here p<0.1 so we reject the H0 i.e., Humidity and Temperature are not related.

Assuming

H0:Severity and Visibility are dependent.

H1:Severity and Visibility are independent.

In [29]:
```
df1=df.sample(n=100)
tables=pd.crosstab(df1['Severity'],df1['Visibility(mi)'])
chi2, p, dof, expected=chi2_contingency(tables.values)
print('Chi-square statistic %0.3f p_value %0.3f' %(chi2,p))
```

```
Chi-square statistic 17.913 p_value 0.329
```

Here p>0.1 so we accept the H0 i.e., Severity and Visibility are related.

Correlation

Only Humidity vs Temperature graph has positive correlation

In [23]:
```
from sklearn.linear_model import LinearRegression
X = da.iloc[:, 6].values.reshape(-1, 1)  # values converts it into a numpy arr
ay
Y = da.iloc[:, 4].values.reshape(-1, 1)  # -1 means that calculate the dimensi
on of rows, but have 1 column
linear_regressor = LinearRegression()  # create object for the class
linear_regressor.fit(X, Y)  # perform linear regression
Y_pred = linear_regressor.predict(X)  # make predictions
plt.scatter(X, Y,)
plt.plot(X, Y_pred, color='red')
plt.show()
```