Load Data In pandas

In [3]:
```
import pandas as pd
data=pd.read_csv('Data_set.csv')
data
```

Out[3]:

	SeqID	Date Of Stop	Time Of Stop	Agency	SubAgency	Description	L
0	ad0f8188-5f69-42bc-837b-d0057c08a75a	10/20/2019	12:02:00	MCP	2nd District, Bethesda	DRIVING VEHICLE IN EXCESS OF REASONABLE AND PR...	RD/WH
1	2ce06b1f-51ea-484b-82c9-f247c7a1938b	10/20/2019	11:41:00	MCP	2nd District, Bethesda	DRIVING VEHICLE ON HIGHWAY WITHOUT CURRENT REG...	GRAN AVE/KENSI
2	7e30365e-00d2-4cdb-8b41-84966d40f17f	10/20/2019	9:14:00	MCP	2nd District, Bethesda	FAILURE OF LICENSEE TO NOTIFY ADMINISTRATION O...	NEBE MARINE
3	7e30365e-00d2-4cdb-8b41-84966d40f17f	10/20/2019	9:14:00	MCP	2nd District, Bethesda	DRIVER FAILURE TO STOP AT STOP SIGN LINE	NEBE MARINE
4	f96a53ba-b1c8-4eba-bc92-4c496d831d62	10/20/2019	8:42:00	MCP	5th District, Germantown	FAILURE TO DISPLAY REGISTRATION CARD UPON DEMA...	GERMAI RO/ RICHTEF
...	...	...	...	...	...	...	
1048570	87eff5b5-2930-49da-b034-93b98ae30426	11/28/2014	11:08:00	MCP	3rd District, Silver Spring	DRIVING UNREGISTERED TRAILER ON HIGHWAY	COLUMBI/ F
1048571	9b7867ac-cfde-4c11-8198-b72afd125618	11/28/2014	10:59:00	MCP	3rd District, Silver Spring	DRIVER USING HANDS TO USE HANDHELD TELEPHONE W...	CAMERC GEORG
1048572	59b79f1d-aac5-423c-9bff-c3e5e96a321c	11/28/2014	10:58:00	MCP	4th District, Wheaton	DISPLAYING EXPIRED REGISTRATION PLATE ISSUED B...	GEOI UNIVERSIT
1048573	59b79f1d-aac5-423c-9bff-c3e5e96a321c	11/28/2014	10:58:00	MCP	4th District, Wheaton	DRIVING VEHICLE ON HIGHWAY WITHOUT CURRENT REG...	GEOI UNIVERSIT
1048574	59b79f1d-aac5-423c-9bff-c3e5e96a321c	11/28/2014	10:58:00	MCP	4th District, Wheaton	OPERATING UNREGISTERED MOTOR VEHICLE ON HIGHWAY	GEOI UNIVERSIT

1048575 rows × 43 columns

In [47]:
```
import matplotlib.pyplot as plt
```

For All Record

In [3]:
```
data.shape
```

Out[3]: (1048575, 43)

Column Names

In [4]:
```
data.head(5)
```

Out[4]:

	SeqID	Date Of Stop	Time Of Stop	Agency	SubAgency	Description	Location
0	ad0f8188-5f69-42bc-837b-d0057c08a75a	10/20/2019	12:02:00	MCP	2nd District, Bethesda	DRIVING VEHICLE IN EXCESS OF REASONABLE AND PR...	RIVER RD/WHITTIER BLVD
1	2ce06b1f-51ea-484b-82c9-f247c7a1938b	10/20/2019	11:41:00	MCP	2nd District, Bethesda	DRIVING VEHICLE ON HIGHWAY WITHOUT CURRENT REG...	GRANDVIEW AVE/KENSINGTON BLVD
2	7e30365e-00d2-4cdb-8b41-84966d40f17f	10/20/2019	9:14:00	MCP	2nd District, Bethesda	FAILURE OF LICENSEE TO NOTIFY ADMINISTRATION O...	NEBEL ST @ MARINELLI DR
3	7e30365e-00d2-4cdb-8b41-84966d40f17f	10/20/2019	9:14:00	MCP	2nd District, Bethesda	DRIVER FAILURE TO STOP AT STOP SIGN LINE	NEBEL ST @ MARINELLI DR
4	f96a53ba-b1c8-4eba-bc92-4c496d831d62	10/20/2019	8:42:00	MCP	5th District, Germantown	FAILURE TO DISPLAY REGISTRATION CARD UPON DEMA...	GERMANTOWN ROAD AND RICHTER FARM

5 rows × 43 columns

Number of violation In 2018

In [5]:
```
# 1
data.columns
```

Out[5]:
```
Index(['SeqID', 'Date Of Stop', 'Time Of Stop', 'Agency', 'SubAgency',
       'Description', 'Location', 'Latitude', 'Longitude', 'Accident', 'Belt
s',
       'Personal Injury', 'Property Damage', 'Fatal', 'Commercial License',
       'HAZMAT', 'Commercial Vehicle', 'Alcohol', 'Work Zone',
       'Search Conducted', 'Search Disposition', 'Search Outcome',
       'Search Reason', 'Search Reason For Stop', 'Search Type',
       'Search Arrest Reason', 'State', 'VehicleType', 'Year', 'Make', 'Mode
l',
       'Color', 'Violation Type', 'Charge', 'Article',
       'Contributed To Accident', 'Race', 'Gender', 'Driver City',
       'Driver State', 'DL State', 'Arrest Type', 'Geolocation'],
      dtype='object')
```

In [6]:
```
# 2
data.shape[0]
```

Out[6]:
```
1048575
```

In [12]:
```
# 3

sum(data["Year"] == 2018)
```

Out[12]:
```
17197
```

In [62]:
```
# 4

sum((data["Year"] >= 2010) & (data.Year <= 2015))
#[True, False, True] and [True, True, False]
```

Out[62]:
```
356496
```

In [70]:
```
# 5
sum(data["Year"] == 2013)
```

Out[70]:
```
65319
```

In [71]:
```
#6
sum(data["Year"] == 2015)
```

Out[71]:
```
65094
```

In [99]:
```
# 7
sum((data["Year"] == 2015) & (data["Fatal"] == "Yes"))
```

Out[99]:
```
10
```

In [95]:
```
#data.groupby("").str.split("/", expand = True)
j=0
v=0
d = data["Date Of Stop"].str.split("/", expand = True)
d.columns= ["Month", "Day", "Date-Year"]
d["Date-Year"].unique()
dYear = d.groupby("Date-Year")["Month"].count()
```

In [96]:
```
# Year with maximum violations
dYear.idxmax()
```

Out[96]: '2015'

In [100]:
```
#Month with max violations
dMonth = d.groupby("Month")["Date-Year"].count()
dMonth.idxmax()
```

Out[100]: '3'

In [101]:
```
#Day with max violations
dDay = d.groupby("Day")["Month"].count()
dDay.idxmax()
```

Out[101]: '18'

In [107]:
```
dTime = data["Time Of Stop"].str.split(":", expand = True)
#dTime.groupby(0, axis = 1).count()
```

Out[107]:

	0	1	2
0	12	02	00
1	11	41	00
2	9	14	00
3	9	14	00
4	8	42	00
...	...	...	...
1048570	11	08	00
1048571	10	59	00
1048572	10	58	00
1048573	10	58	00
1048574	10	58	00

1048575 rows × 3 columns

In [110]:
```
# Time in hour with highest violations
dTime.groupby(0)[1].count().idxmax()
```

Out[110]: '22'

In [124]:
```
dAlcohol = data[data["Alcohol"] == "Yes"]
d = dAlcohol["Date Of Stop"].str.split("/", expand = True)
t = dAlcohol["Time Of Stop"].str.split(":", expand = True)
d.columns = ["Month", "Day", "Date-Year"]
t.columns = ["Hour", "Minute", "Second"]
dAlcohol = dAlcohol.join(t)
dAlcohol = dAlcohol.join(d)
```

In [128]:
```
#Year with highest ALcohol Violations
dAlcohol.groupby("Date-Year").count()["SeqID"].idxmax()
```

Out[128]: '2016'

In [129]:
```
#Month with highest ALcohol Violations
dAlcohol.groupby("Month").count()["SeqID"].idxmax()
```

Out[129]: '8'

In [130]:
```
#Day with highest ALcohol Violations
dAlcohol.groupby("Day").count()["SeqID"].idxmax()
```

Out[130]: '5'

In [131]:
```
#Time with highest ALcohol Violations
dAlcohol.groupby("Hour").count()["SeqID"].idxmax()
```

Out[131]: '17'

In [136]:
```
#Brand with highest number of violations
data.groupby("Make")["SeqID"].count().idxmax()
```

Out[136]: 'TOYOTA'

In [4]:
```
date = data["Date Of Stop"].str.split("/", expand = True)
time = data["Time Of Stop"].str.split(":", expand = True)
time.columns = ["Hour", "Minute", "Second"]
date.columns = ["Day", "Month", "Date-Year"]
data = data.join(date)
data = data.join(time)
```

In [25]:
```
newYear = data[(data["Month"] == '1') & (data["Day"] == '1')]
```

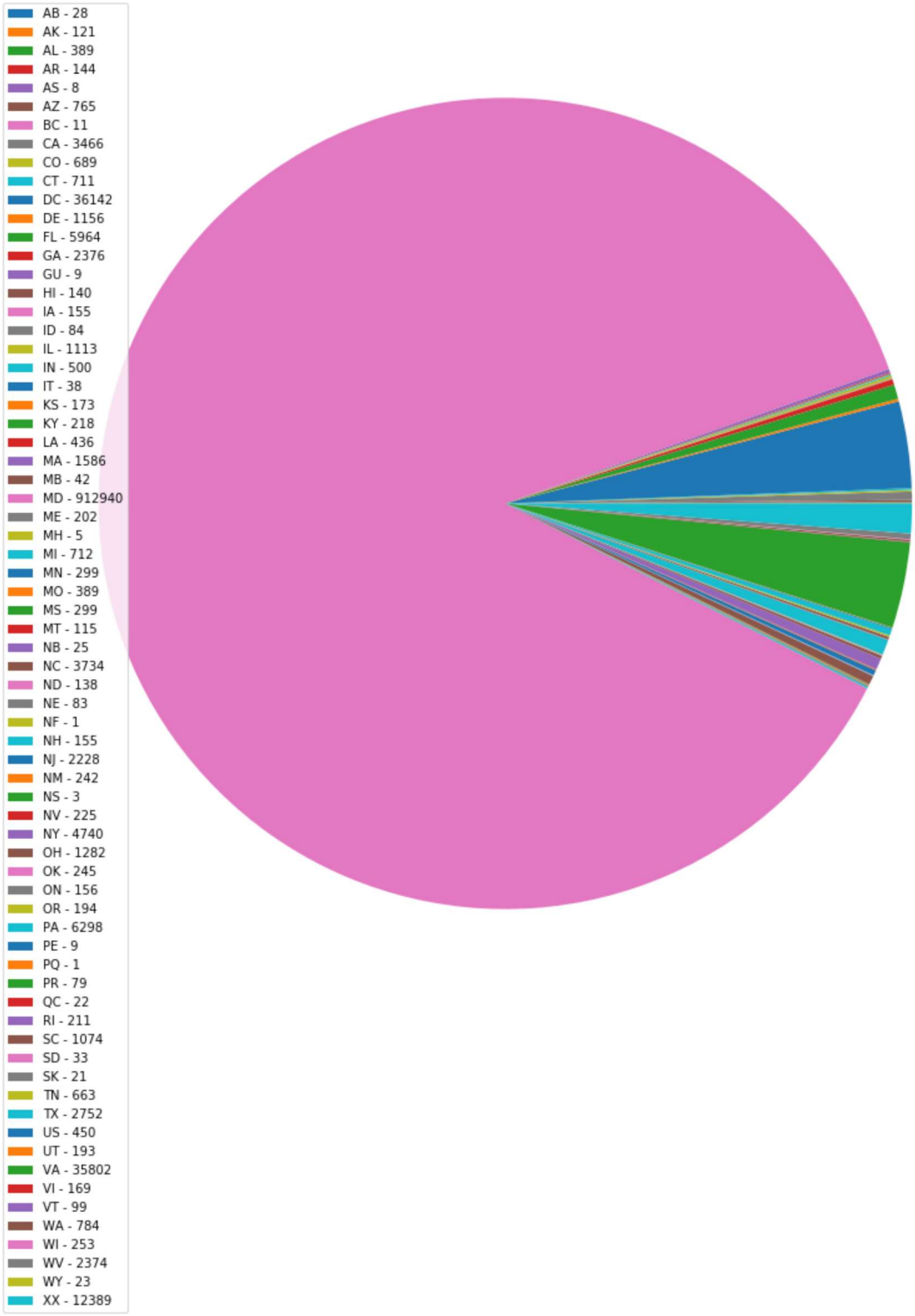
In [29]:
```
# Year with highest number of violations on New Year's Eve
newYear.groupby("Date-Year").count()["SeqID"].idxmax()
```

Out[29]: '2016'

In [48]:
```
stateData = data.groupby("DL State")["SeqID"].count()
```

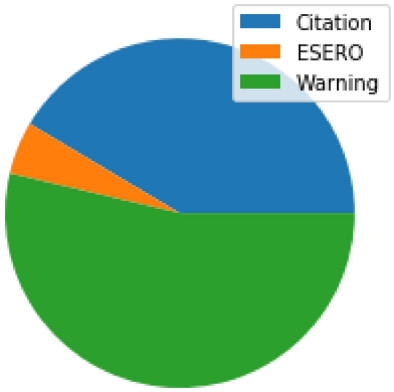
In [58]:
```
# Violations in different states

plt.figure(figsize = (15, 15))
plt.pie(stateData);
plt.legend([i[0] + " - " +str(i[1]) for i in stateData.iteritems()]);
```



In [64]:
```
trend = data.groupby("Violation Type")["SeqID"].count()
```

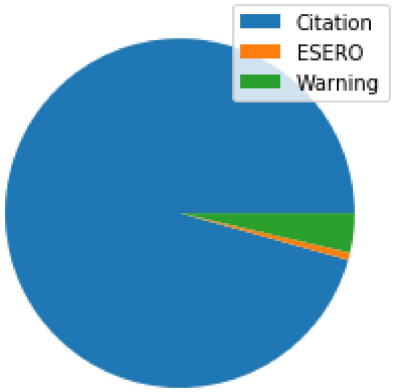
In [68]:
```
plt.pie(trend);
plt.legend(trend.index);
```



In [73]:
```
dAlco = data[data["Alcohol"] == "Yes"]
```

In [75]:
```
trendAlco = dAlco.groupby("Violation Type")["SeqID"].count()
```

In [80]:
```
plt.pie(trendAlco);
plt.legend(trendAlco.index);
```



In [ ]:

In [ ]:

In [ ]: