In [1]:

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

dataset = pd.read_csv('crime_data_Proj1.csv')
columns = ['Unnamed: 0']
dataset.drop(columns, inplace = True, axis=1)
```

In [2]:

```
dataset.head(8)
```

Out[2]:

	ID	Case Number	Date	Block	IUCR	Primary Type	Description	Location Description	Arrest	Domestic	...	Ward	Cc
0	6407111	HP485721	07/26/2008 02:30:00 PM	085XX S MUSKEGON AVE	1320	CRIMINAL DAMAGE	TO VEHICLE	STREET	False	False	...	10.0	
1	11398199	JB372830	07/31/2018 10:57:00 AM	092XX S ELLIS AVE	143C	WEAPONS VIOLATION	UNLAWFUL POSS AMMUNITION	POOL ROOM	True	False	...	8.0	
2	5488785	HN308568	04/27/2007 10:30:00 AM	062XX N TRIPP AVE	0610	BURGLARY	FORCIBLE ENTRY	RESIDENCE	True	False	...	39.0	
3	11389116	JB361368	07/23/2018 08:55:00 AM	0000X N KEELER AVE	0560	ASSAULT	SIMPLE	NURSING HOME/RETIREMENT HOME	False	False	...	28.0	
4	12420431	JE297624	07/11/2021 06:40:00 AM	016XX W HARRISON ST	051A	ASSAULT	AGGRAVATED - HANDGUN	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	False	...	27.0	
5	1699235	G498287	08/21/2001 12:00:00 AM	003XX W 28 PL	0810	THEFT	OVER $500	STREET	False	False	...	NaN	
6	5061155	HM660983	10/14/2006 10:00:00 PM	006XX S CENTRAL AVE	0320	ROBBERY	STRONGARM - NO WEAPON	CTA PLATFORM	False	False	...	29.0	
7	9876456	HX527438	12/02/2014 11:48:00 AM	043XX W POTOMAC AVE	1811	NARCOTICS	POSS: CANNABIS 30GMS OR LESS	ALLEY	True	False	...	37.0	

8 rows × 22 columns

In [3]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2278726 entries, 0 to 2278725
Data columns (total 22 columns):
 #   Column                Dtype  
---  ------                -----  
 0   ID                    int64  
 1   Case Number           object 
 2   Date                  object 
 3   Block                 object 
 4   IUCR                  object 
 5   Primary Type          object 
 6   Description           object 
 7   Location Description  object 
 8   Arrest                bool   
 9   Domestic              bool   
 10  Beat                  int64  
 11  District              float64
 12  Ward                  float64
 13  Community Area        float64
 14  FBI Code              object 
 15  X Coordinate          float64
 16  Y Coordinate          float64
 17  Year                  int64  
 18  Updated On            object 
 19  Latitude              float64
 20  Longitude             float64
 21  Location              object 
dtypes: bool(2), float64(7), int64(3), object(10)
memory usage: 352.1+ MB
```

In [4]:

```
# idntifiying missing values

missing_val = dataset.isna().sum()
missing_val
```

Out[4]:

```
ID                           0
Case Number                  1
Date                         0
Block                        0
IUCR                         0
Primary Type                 0
Description                  0
Location Description      2877
Arrest                       0
Domestic                     0
Beat                         0
District                    12
Ward                    184695
Community Area          184267
FBI Code                     0
X Coordinate             23985
Y Coordinate             23985
Year                         0
Updated On                   0
Latitude                 23985
Longitude                23985
Location                 23985
dtype: int64
```

In [5]:

```python
# check for percentage of the missing values if percentage less than < 5% then i will drop it.
from pandas import read_csv

data = read_csv('crime_data_Proj1.csv', header= None, na_values= False )

for i in range(data.shape[1]):
    missing = data[[i]].isnull().sum()
    per = missing / data.shape[0] * 100
    print('> %d,  Missing: %d  (%.1f%%)' % (i, missing, per))
```

```
C:\Users\Shehu\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3444: DtypeWarning: Columns (1,11,
12,13,14,16,17,18,20,21) have mixed types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)

> 0,  Missing: 2  (0.0%)
> 1,  Missing: 0  (0.0%)
> 2,  Missing: 1  (0.0%)
> 3,  Missing: 0  (0.0%)
> 4,  Missing: 0  (0.0%)
> 5,  Missing: 0  (0.0%)
> 6,  Missing: 0  (0.0%)
> 7,  Missing: 0  (0.0%)
> 8,  Missing: 2877  (0.1%)
> 9,  Missing: 1675252  (73.5%)
> 10,  Missing: 1965311  (86.2%)
> 11,  Missing: 0  (0.0%)
> 12,  Missing: 12  (0.0%)
> 13,  Missing: 184695  (8.1%)
> 14,  Missing: 184282  (8.1%)
> 15,  Missing: 0  (0.0%)
> 16,  Missing: 24013  (1.1%)
> 17,  Missing: 24013  (1.1%)
> 18,  Missing: 0  (0.0%)
> 19,  Missing: 0  (0.0%)
> 20,  Missing: 23985  (1.1%)
> 21,  Missing: 23985  (1.1%)
> 22,  Missing: 23985  (1.1%)
```

In [2]:

```python
# treating ward and community Area features by replacing nan with 0

from numpy import nan

dataset['Ward'] = dataset['Ward'].replace(nan, 0)
dataset['Community Area'] = dataset['Community Area'].replace(nan, 0)
```

In [3]:

```python
# treating the missing value by droping nan

dataset.dropna(inplace = True)
```

In [5]:

```python
dataset.isna().sum()
```

Out[5]:

```
ID                      0
Case Number             0
Date                    0
Block                   0
IUCR                    0
Primary Type            0
Description             0
Location Description    0
Arrest                  0
Domestic                0
Beat                    0
District                0
Ward                    0
Community Area          0
FBI Code                0
X Coordinate            0
Y Coordinate            0
Year                    0
Updated On              0
Latitude                0
Longitude               0
Location                0
dtype: int64
```

In [9]:

```python
print(round(2252860 / 2278726 * 100,2), "percentage of the data has been retained.")
```

```
98.86 percentage of the data has been retained.
```

In [10]:

```python
dataset.describe(exclude='number').T.sort_values(by='unique')
```

Out[10]:

	count	unique	top	freq
Arrest	2252860	2	False	1654990
Domestic	2252860	2	False	1941700
FBI Code	2252860	26	06	473222
Primary Type	2252860	35	THEFT	473222
Location Description	2252860	198	STREET	586596
IUCR	2252860	391	0820	183064
Description	2252860	513	SIMPLE	267014
Updated On	2252860	3620	02/10/2018 03:50:01 PM	835879
Block	2252860	53378	100XX W OHARE ST	4785
Location	2252860	545165	(41.976290414, -87.905227221)	4227
Date	2252860	1334502	01/01/2007 12:01:00 AM	53
Case Number	2252860	2252813	HK172551	3

In [11]:

```python
# checking for duplicate in the dataset

# duplicate = dataset.duplicated()
# duplicate.any()

# dataset = dataset.drop_duplicates()

duplicate = dataset.duplicated(keep = False).sum()
duplicate
```

Out[11]:

```
0
```

In [6]:

```python
dataset.shape
```

Out[6]:

```
(2252860, 22)
```

In [15]:

```python
# Number of distinct crimes in the city

crimes = dataset['Primary Type'].unique()
print("The Number of distinct crimes are:", len(crimes))
print()
print("The Distinct Crimes are : \n", crimes)
```

```
The Number of distinct crimes are: 35

The Distinct Crimes are :
 ['CRIMINAL DAMAGE' 'WEAPONS VIOLATION' 'BURGLARY' 'ASSAULT' 'THEFT'
 'ROBBERY' 'NARCOTICS' 'MOTOR VEHICLE THEFT' 'BATTERY' 'OTHER OFFENSE'
 'PROSTITUTION' 'DECEPTIVE PRACTICE' 'INTIMIDATION'
 'INTERFERENCE WITH PUBLIC OFFICER' 'CRIMINAL TRESPASS' 'STALKING'
 'OFFENSE INVOLVING CHILDREN' 'PUBLIC PEACE VIOLATION' 'SEX OFFENSE'
 'CRIM SEXUAL ASSAULT' 'HOMICIDE' 'LIQUOR LAW VIOLATION'
 'CRIMINAL SEXUAL ASSAULT' 'KIDNAPPING' 'ARSON' 'GAMBLING'
 'CONCEALED CARRY LICENSE VIOLATION' 'PUBLIC INDECENCY' 'RITUALISM'
 'OBSCENITY' 'NON - CRIMINAL' 'OTHER NARCOTIC VIOLATION'
 'HUMAN TRAFFICKING' 'NON-CRIMINAL' 'NON-CRIMINAL (SUBJECT SPECIFIED)']
```

In [21]:

```python
district_crimes = dataset['District'].unique()
print(district_crimes)

# Filter out the Top 5 criminal districts
top_5_district = dataset['District'].value_counts().sort_values(ascending=False).head()
top_5_district
```

```
[ 4. 17. 11. 12.  2. 15. 25. 14.  8.  6. 19.  1.  7. 20. 18.  3.  9. 22.
  5. 16. 10. 24. 31.]
```

Out[21]:

```
8.0     152306
11.0    144802
6.0     131664
7.0     131602
4.0     128782
Name: District, dtype: int64
```

In [20]:

```
top_5_crimes = dataset['Primary Type'].value_counts().sort_values(ascending=False).head()
top_5_crimes
```

Out[20]:

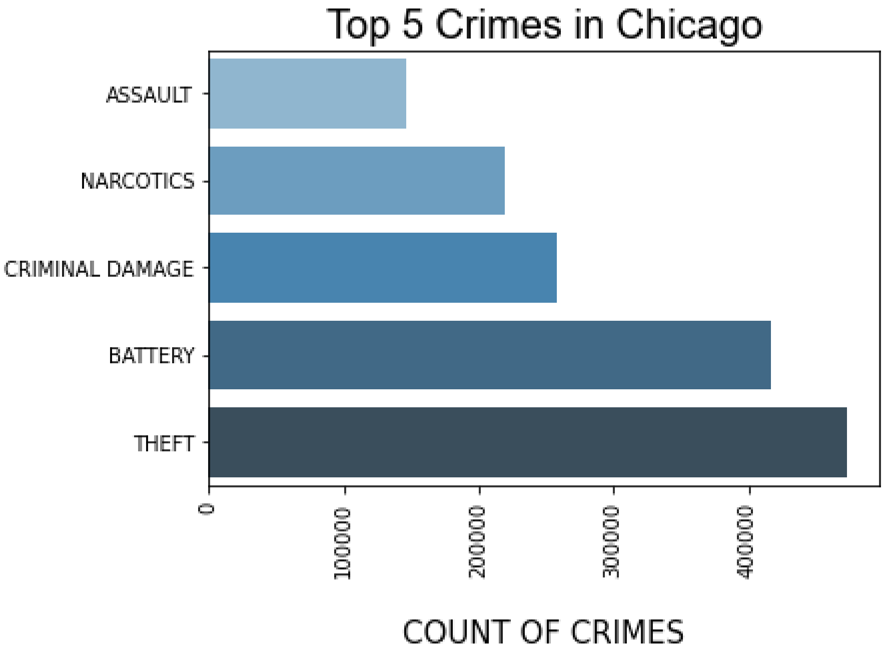
```
THEFT              473222
BATTERY            416908
CRIMINAL DAMAGE    258649
NARCOTICS          219441
ASSAULT            147046
Name: Primary Type, dtype: int64
```

In [18]:

```python
import seaborn as sns
import matplotlib.pyplot as plt

crime = dataset.groupby('Primary Type', as_index = False).agg({"ID": "count"})
crime_type = crime.sort_values(by = ['ID'], ascending = False).head()
crime_type = crime_type.sort_values(by='ID', ascending = True)
sns.barplot(x = 'ID', y = 'Primary Type', data = crime_type, palette = "Blues_d")

plt.title("Top 5 Crimes in Chicago", fontdict = {'fontsize': 20, 'fontname':'Arial', 'color': '#000000'})
plt.xlabel("\nCOUNT OF CRIMES", fontdict = {'fontsize': 15})
plt.ylabel("")
plt.xticks(rotation=90)
plt.show()
```



In [ ]:

```python

```

In [ ]:

```python
# feature engineering extracting the features month, period and day from the feature Date
```

In [4]:

```python
from dateutil.parser import parse
from datetime import datetime
tCol = dataset.Date

List = [(datetime.ctime(parse(x[0:-3])), x[-2:]) for x in tCol]
dayList = []
monthList = []
periodList = []

for row in List:
    day = row[0][0:4]
    month = row[0][4:7]
    if row[1] == 'AM':
        period = 'Morning'
    elif row[1] == 'PM' and int(row[0][11:13]) < 4:
        period = 'Afternoon'
    elif row[1] == 'PM' and int(row[0][11:13]) < 5:
        period = 'Evening'
    elif row[1] == 'PM' and int(row[0][11:13]) > 5:

        period = 'Night'
    else:
        period = 'Unknown'

    dayList.append(day)
    monthList.append(month)
    periodList.append(period)

print(len(dayList), len(monthList), len(periodList))
```

```
2252860 2252860 2252860
```

In [5]:

```python
# create new features month, period and day into the dataset (extracting from feature Date)

dataset['month'] = monthList
dataset['period'] = periodList
dataset['day'] = dayList
```

In [6]:

```python
# dropping the following features to avoid duplicate

del dataset['Date']
del dataset['Updated On']
```

In [7]:

```python
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2252860 entries, 0 to 2278725
Data columns (total 23 columns):
 #   Column                Dtype  
---  ------                -----  
 0   ID                    int64  
 1   Case Number           object 
 2   Block                 object 
 3   IUCR                  object 
 4   Primary Type          object 
 5   Description           object 
 6   Location Description  object 
 7   Arrest                bool   
 8   Domestic              bool   
 9   Beat                  int64  
 10  District              float64
 11  Ward                  float64
 12  Community Area        float64
 13  FBI Code              object 
 14  X Coordinate          float64
 15  Y Coordinate          float64
 16  Year                  int64  
 17  Latitude              float64
 18  Longitude             float64
 19  Location              object 
 20  month                 object 
 21  period                object 
 22  day                   object 
dtypes: bool(2), float64(7), int64(3), object(11)
memory usage: 382.4+ MB
```

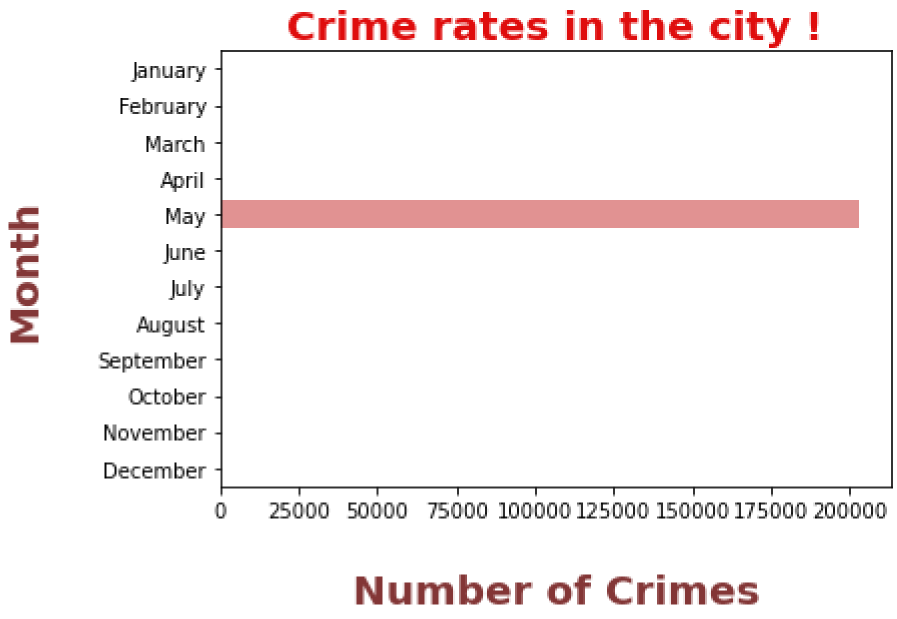
In [24]:

```
sns.countplot(y = 'month', data = dataset, palette = ["#DF0D0D"], order = ['January', 'February', 'March',

plt.title("Crime rates in the city !", fontdict={'fontsize': 20, 'color': '#DF0D0D', 'fontname':'Agency FB'
plt.ylabel("Month\n", fontdict = {'fontsize': 20}, weight = "bold", color = "#833636")
plt.xlabel("\nNumber of Crimes", fontdict =  {'fontsize': 20}, weight = "bold", color = "#833636")

plt.xticks(fontsize = 10, color = 'black')
plt.yticks(fontsize = 10, color = 'black')
plt.show()
```

```
findfont: Font family ['Agency FB'] not found. Falling back to DejaVu Sans.
```



In [ ]:

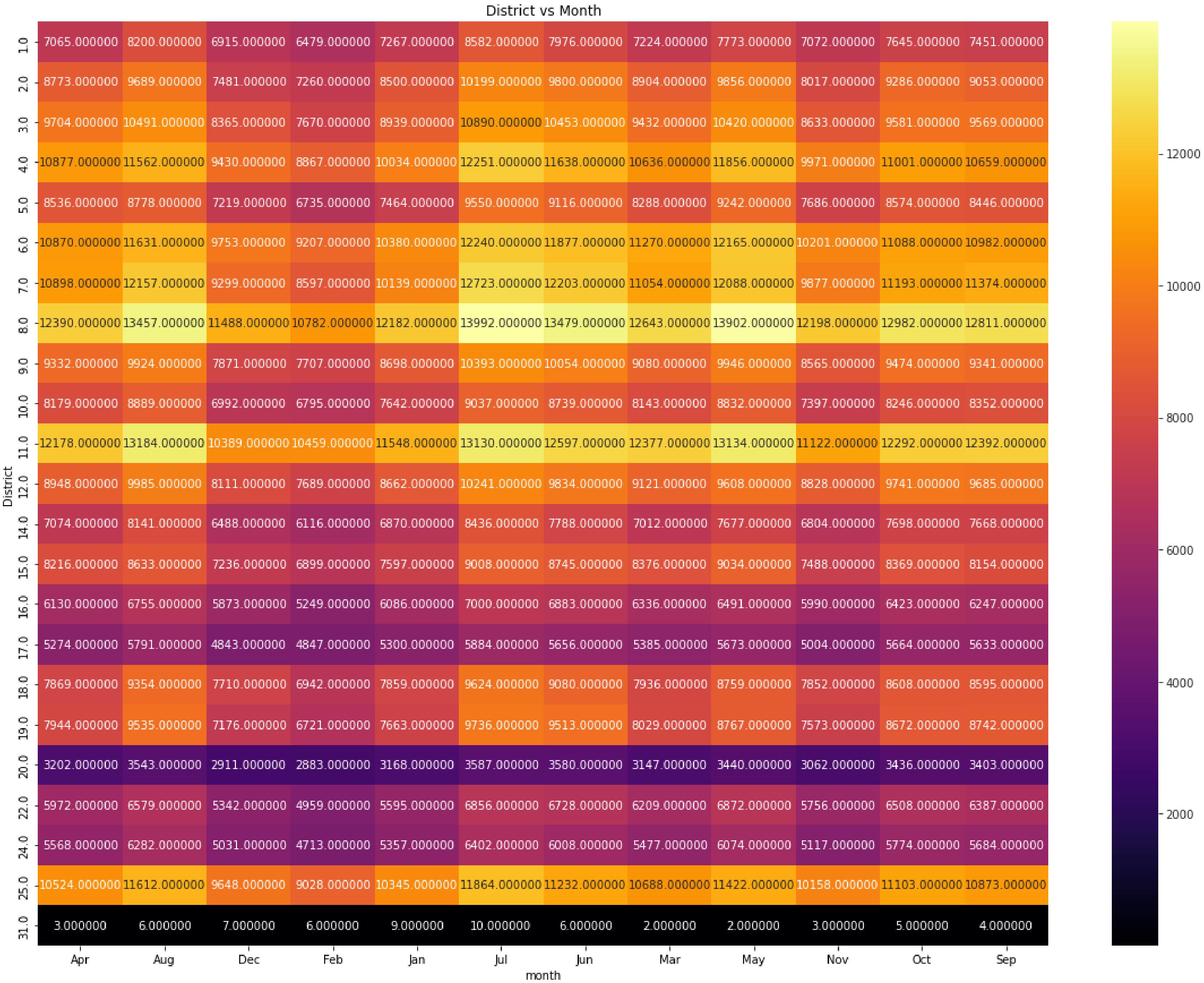
```
# The month of May have seen the most high crime rates in the city.
```

In [26]:

```
district_crime = dataset.groupby(['District','month'], as_index =  False).agg({'Primary Type':"count"})
district_crime.columns

district_crime = district_crime.pivot("District", "month", "Primary Type")


plt.figure(figsize = (20,15))
plt.title("District vs Month")
with sns.axes_style("white"):
    sns.heatmap(district_crime, mask = district_crime.isnull(), cmap = "inferno", annot = True, fmt = "f")
```



In [ ]:

```
# district 8 and 11 are having the high crime rate.
```

In [285]:

```python
# Location attributes = ['Location Description','Beat','District', 'Ward', 'Community Area','X Coordinate',
%matplotlib inline
top_crime = dataset.groupby(['District', 'Primary Type']).size().reset_index(name='counts').groupby('Distri
#print(topk)

# factor plot to make multiple plots
graph = sns.catplot("Primary Type", y = 'counts', col = "District", col_wrap = 3,
                   data = top_crime, kind = 'bar')
for ax in graph.axes:
    plt.setp(ax.get_xticklabels(), visible = True, rotation = 30, ha='right')

plt.subplots_adjust(hspace = 0.4)
```

```
C:\Users\Shehu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following vari
able as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing
other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

In [28]:

```
crime_period = dataset[['Primary Type','period']]
crime_period = crime_period.groupby(['Primary Type'])
crime_period.head()
```

Out[28]:

	Primary Type	period
0	CRIMINAL DAMAGE	Afternoon
1	WEAPONS VIOLATION	Morning
2	BURGLARY	Morning
3	ASSAULT	Morning
4	ASSAULT	Morning
...	...	...
668399	RITUALISM	Evening
867154	NON - CRIMINAL	Morning
880836	RITUALISM	Night
1031143	NON-CRIMINAL (SUBJECT SPECIFIED)	Night
1587959	NON-CRIMINAL (SUBJECT SPECIFIED)	Unknown

172 rows × 2 columns

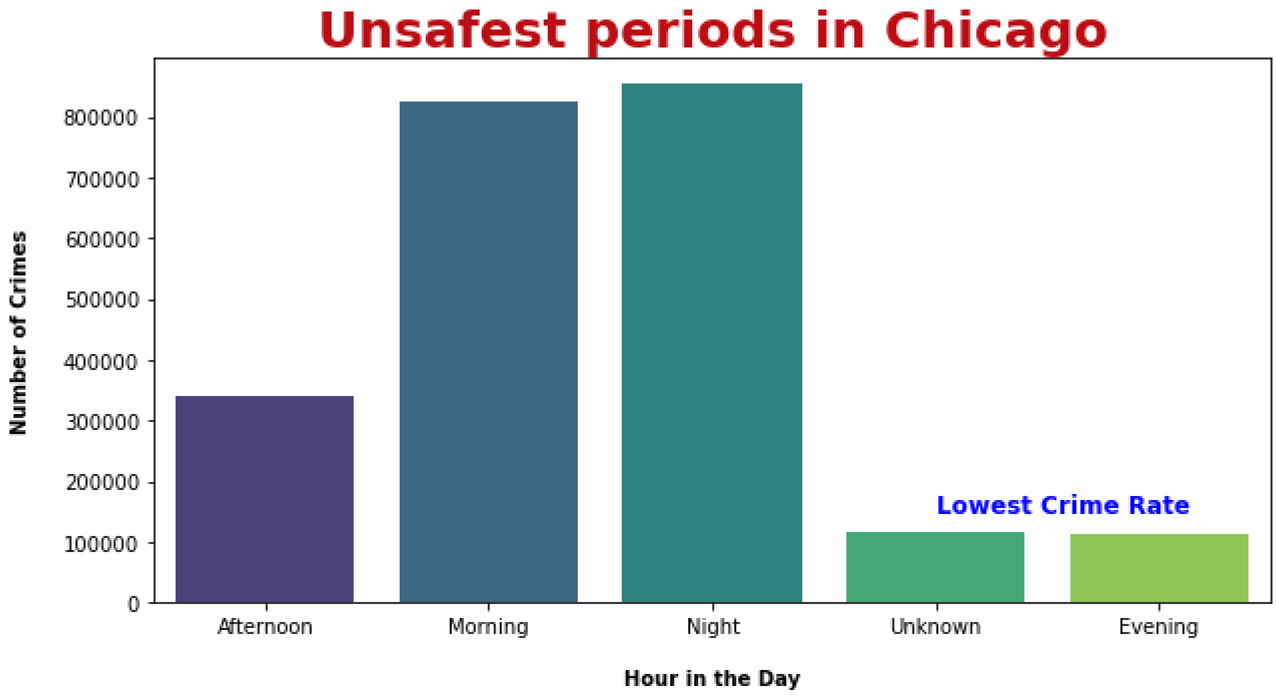
In [29]:

```python
fig, ax = plt.subplots(figsize=(10, 5))
sns.countplot(x = 'period', data = dataset, palette = "viridis")

# Aesthetic appeal
plt.title("Unsafest periods in Chicago", fontdict={'fontsize': 25, 'color': '#bb0e14','fontname':'Agency FB
plt.xlabel("\nHour in the Day", fontdict={'fontsize': 10}, weight='bold')
plt.ylabel("Number of Crimes\n", fontdict={'fontsize': 10}, weight="bold")

plt.text(3, 150000, 'Lowest Crime Rate', fontdict= {'fontsize': 12, 'color':"blue" }, weight='bold')
plt.show()
```

```
findfont: Font family ['Agency FB'] not found. Falling back to DejaVu Sans.
```



In [30]:

```
day_crime = dataset.groupby(['Primary Type', 'day']).agg({'day':"count"})
day_crime.columns = ["Count"]
day_crime
```

Out[30]:

```
                                Count
        Primary Type    day
------------------------------------
              ARSON     Fri       500
                        Mon       566
                        Sat       598
                        Sun       652
                        Thu       490
                ...     ...       ...
  WEAPONS VIOLATION     Sat      4558
                        Sun      4225
                        Thu      4239
                        Tue      4284
                        Wed      4109
```

235 rows × 1 columns

In [ ]:

```

```

In [ ]:

```
# Building Models
```

In [8]:

```python
# treating the boolean features (Arresrt, Domestic) by replaced True and False by 1 and 0

# dataset = dataset.replace({False: 0, True: 1})
Arrest = dataset['Arrest'].replace((True, False), (1, 0))
Domestic = dataset['Domestic'].replace((True, False), (1, 0))

dataset['Arrest'] = Arrest
dataset['Domestic'] = Domestic
```

In [9]:

```python
import numpy as np

for categorical_column in ['Community Area', 'Ward', 'District']:
    dataset[categorical_column] = dataset[categorical_column].astype(np.int64)

for categorical_column in ['Primary Type', 'Description', 'Location Description', 'month']:
    dataset[categorical_column] = dataset[categorical_column].astype('category')
```

In [10]:

```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

dataset['Primary Type'] = le.fit_transform(dataset['Primary Type']).astype('int64')
dataset['Location Description'] = le.fit_transform(dataset['Location Description']).astype('int64')
dataset['Description'] = le.fit_transform(dataset['Description']).astype('int64')
dataset['month'] = le.fit_transform(dataset['month']).astype('int64')
dataset['District'] = le.fit_transform(dataset['District']).astype('int64')
dataset['Location'] = le.fit_transform(dataset['Location']).astype('int64')
dataset['period'] = le.fit_transform(dataset['period']).astype('int64')
dataset['day'] = le.fit_transform(dataset['day']).astype('int64')
```

In [11]:

```python
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2252860 entries, 0 to 2278725
Data columns (total 23 columns):
 #   Column                Dtype  
---  ------                -----  
 0   ID                    int64  
 1   Case Number           object 
 2   Block                 object 
 3   IUCR                  object 
 4   Primary Type          int64  
 5   Description           int64  
 6   Location Description  int64  
 7   Arrest                int64  
 8   Domestic              int64  
 9   Beat                  int64  
 10  District              int64  
 11  Ward                  int64  
 12  Community Area        int64  
 13  FBI Code              object 
 14  X Coordinate          float64
 15  Y Coordinate          float64
 16  Year                  int64  
 17  Latitude              float64
 18  Longitude             float64
 19  Location              int64  
 20  month                 int64  
 21  period                int64  
 22  day                   int64  
dtypes: float64(4), int64(15), object(4)
memory usage: 412.5+ MB
```

In [ ]:

```python

```

In [133]:

```python
columns = ['Primary Type', 'Description', 'Location Description','Arrest','Domestic','Beat','District','War
           'Community Area','X Coordinate','Y Coordinate','Year','Latitude','Longitude','Location','month',

data = dataset[columns]
```

In [134]:

```
data.head()
```

Out[134]:

	Primary Type	Description	Location Description	Arrest	Domestic	Beat	District	Ward	Community Area	X Coordinate	Y Coordinate	Year	Latitude	Long
0	6	462	171	0	0	423	3	10	46	1196638.0	1848800.0	2008	41.739980	-87.55
1	34	468	139	1	0	413	3	8	47	1184499.0	1843935.0	2018	41.726922	-87.59
2	3	216	145	1	0	1711	15	39	12	1146911.0	1941022.0	2007	41.994138	-87.73
3	1	421	125	0	0	1115	10	28	26	1148388.0	1899882.0	2018	41.881217	-87.73
4	1	25	134	0	0	1231	11	27	28	1165430.0	1897441.0	2021	41.874174	-87.66

In [135]:

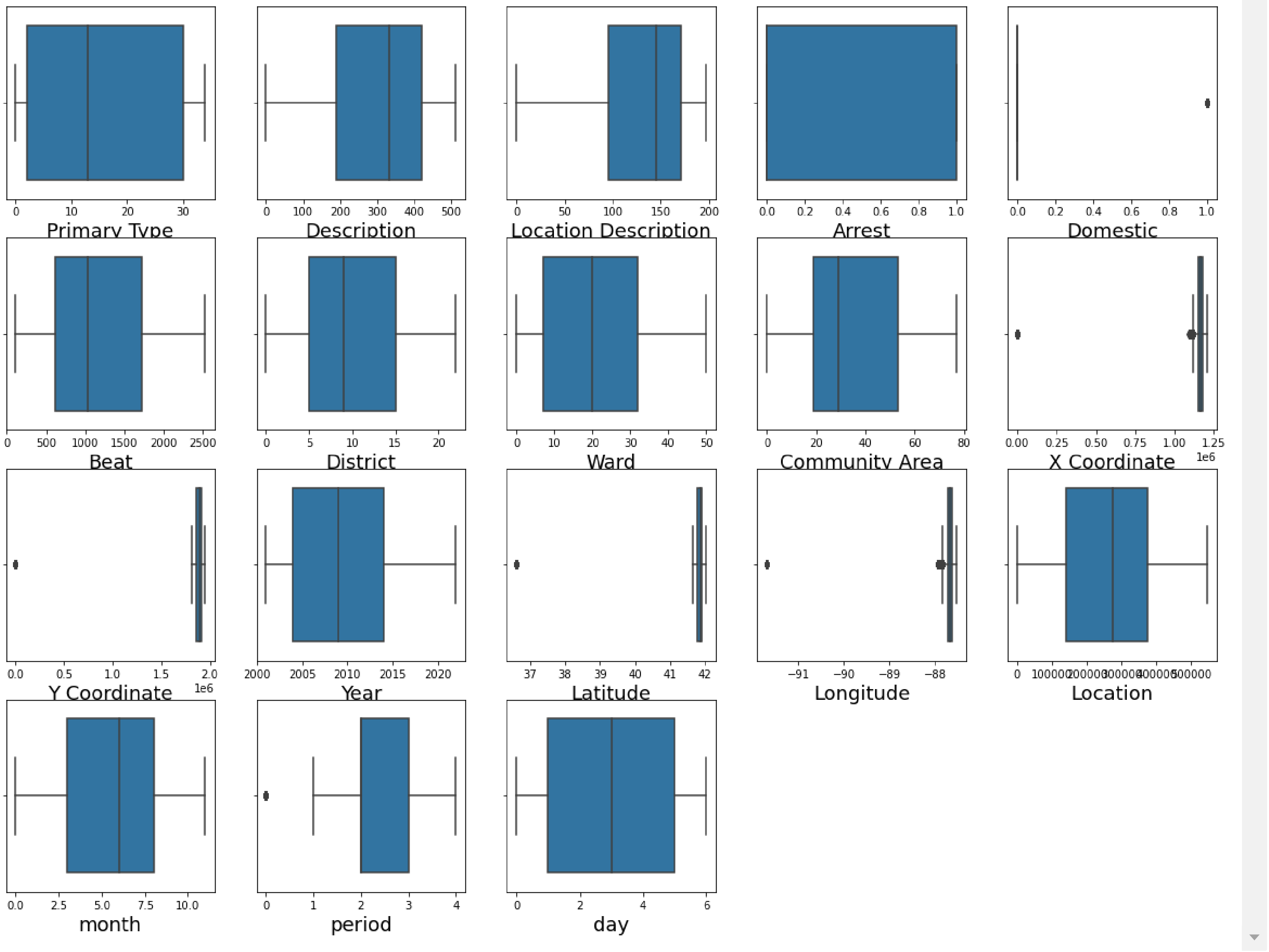
```
data.shape
```

Out[135]:

```
(2252860, 18)
```

In [138]:

```
# Checking for outliers
plt.figure(figsize = (20, 15))
for i in range (len(data.columns)):
    plt.subplot(4, 5, i+1)
    sns.boxplot(x = data.iloc[:, i])
    plt.xlabel(data.columns[i], size = 18)
```


Primary Type
0 10 20 30
Description
0 100 200 300 400 500
Location Description
0 50 100 150 200
Arrest
0.0 0.2 0.4 0.6 0.8 1.0
Domestic
0.0 0.2 0.4 0.6 0.8 1.0
Beat
0 500 1000 1500 2000 2500
District
0 5 10 15 20
Ward
0 10 20 30 40 50
Community Area
0 20 40 60 80
X Coordinate
0.00 0.25 0.50 0.75 1.00 1.25
1e6
Y Coordinate
0.0 0.5 1.0 1.5 2.0
1e6
Year
2000 2005 2010 2015 2020
Latitude
37 38 39 40 41 42
Longitude
−91 −90 −89 −88
Location
0 100000 200000 300000 400000 500000
month
0.0 2.5 5.0 7.5 10.0
period
0 1 2 3 4
day
0 2 4 6

In [139]:

```python
# Removing outliers
def Outliers(data, feature):
    IQ1 = data[feature].quantile(0.25)
    IQ3 = data[feature].quantile(0.75)
    IQR = IQ3 - IQ1

    lower_bound = IQ1 - 1.5 * IQR
    upper_bound = IQ3 + 1.5 * IQR

    index = data.index[ (data[feature] < lower_bound) | (data[feature] > upper_bound) ]
    return index
```

In [140]:

```python
Outliers(data, 'period')
```

Out[140]:

```
Int64Index([      0,      18,      24,      26,      28,      32,      44,
                 51,      78,      79,
            ...
            2278665, 2278667, 2278669, 2278687, 2278688, 2278690, 2278694,
            2278699, 2278700, 2278721],
           dtype='int64', length=341608)
```

In [144]:

```python
# Getting index of all the outliers

index = []

for i in data.columns:
    index.extend(Outliers(data, i))
index = set(index)
print("Total number of outliers are {}".format(len(index)))

# Dropping all the outliers
data.drop(index, inplace = True, axis = 0)
```

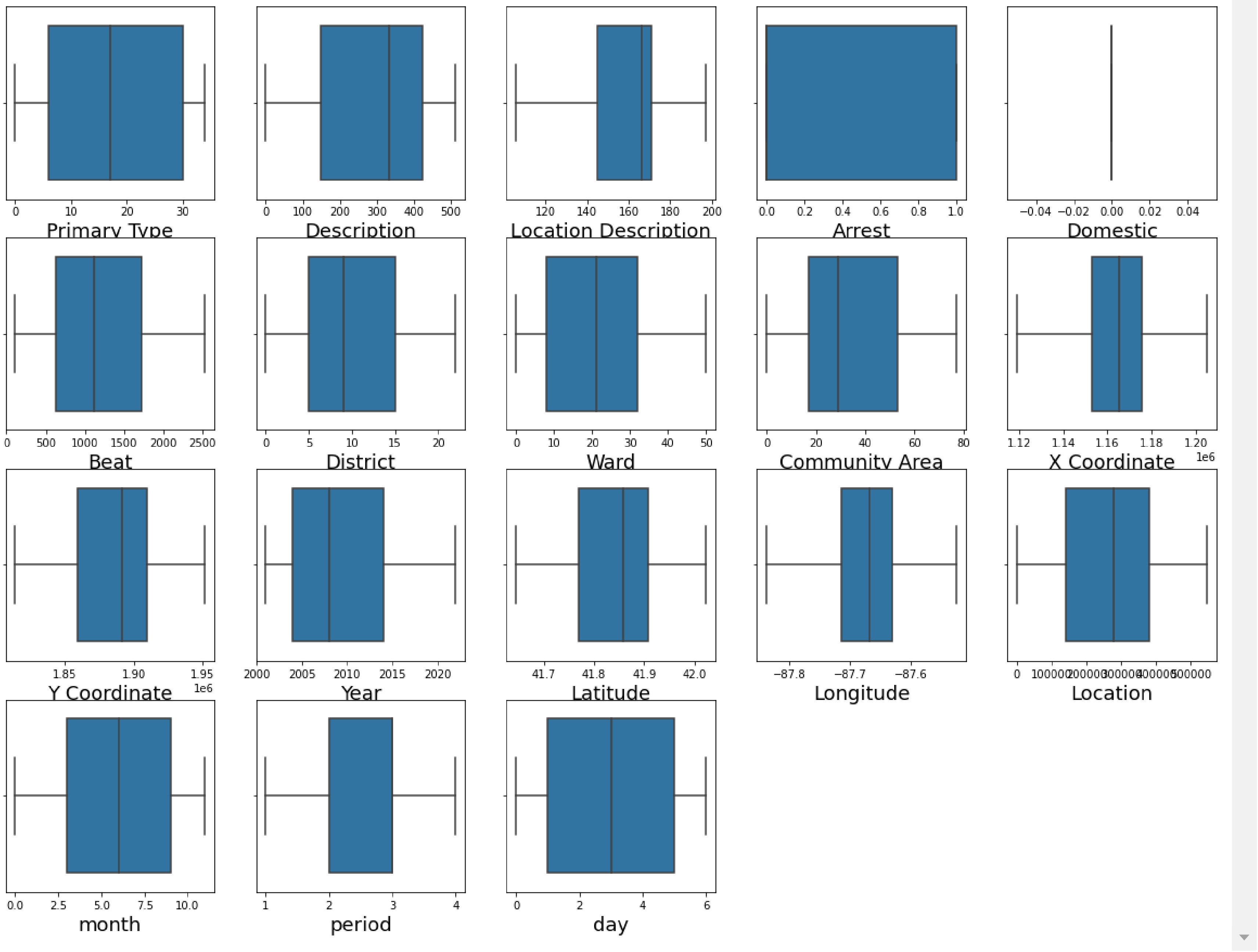
```
Total number of outliers are 0
```

In [145]:

```python
# checking for outliers in the dataseet

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize = (20, 15))
for i in range(len(data.columns)):
    plt.subplot(4, 5, i + 1)
    sns.boxplot(x = data.iloc[:, i])
    plt.xlabel(data.columns[i], size = 18)
```

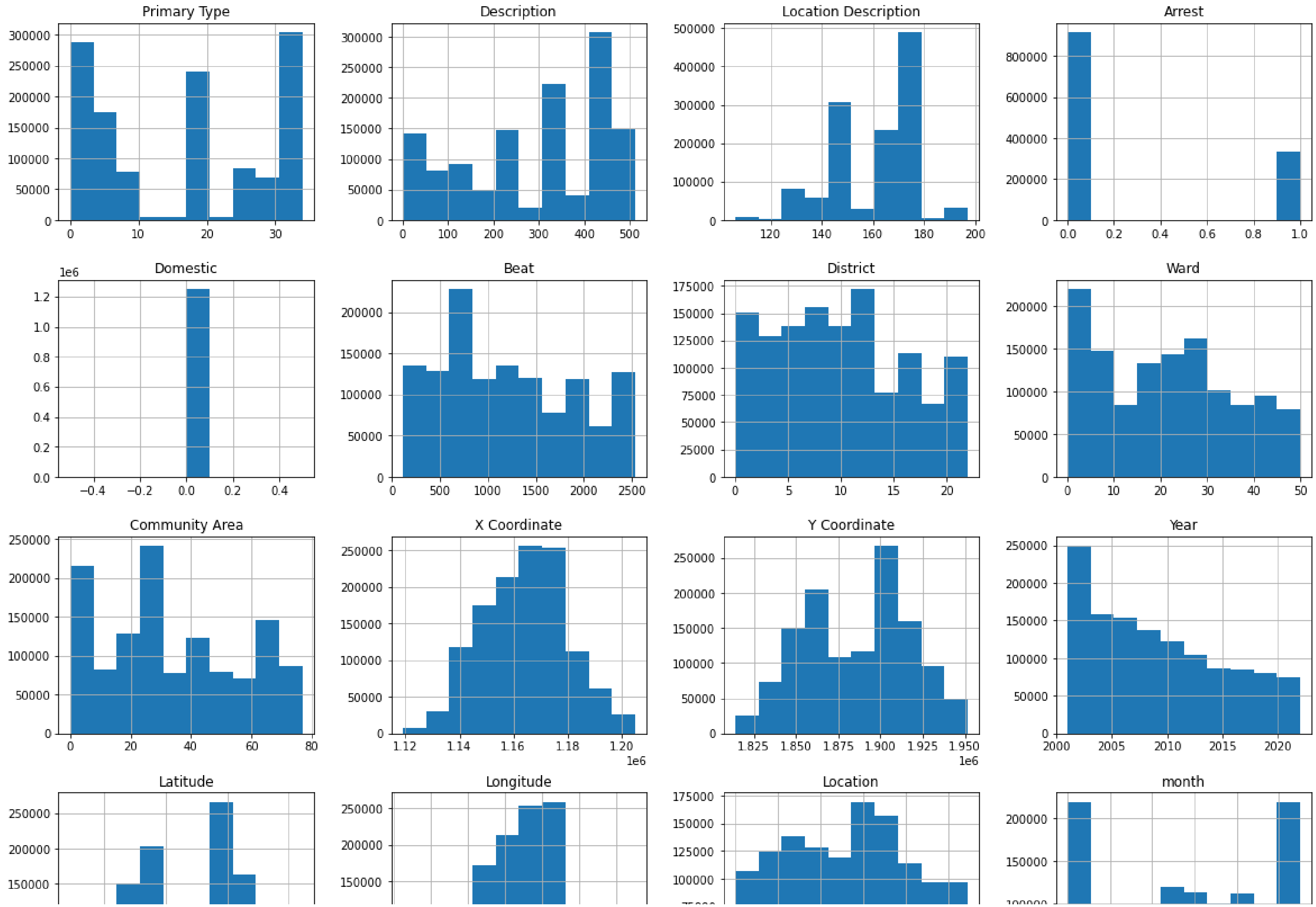

0 10 20 30
Primary Type
0 100 200 300 400 500
Description
120 140 160 180 200
Location Description
0.0 0.2 0.4 0.6 0.8 1.0
Arrest
−0.04 −0.02 0.00 0.02 0.04
Domestic
0 500 1000 1500 2000 2500
Beat
0 5 10 15 20
District
0 10 20 30 40 50
Ward
0 20 40 60 80
Community Area
1.12 1.14 1.16 1.18 1.20
1e6
X Coordinate
1.85 1.90 1.95
1e6
Y Coordinate
2000 2005 2010 2015 2020
Year
41.7 41.8 41.9 42.0
Latitude
−87.8 −87.7 −87.6
Longitude
0 100000 200000 300000 400000 500000
Location
0.0 2.5 5.0 7.5 10.0
month
1 2 3 4
period
0 2 4 6
day

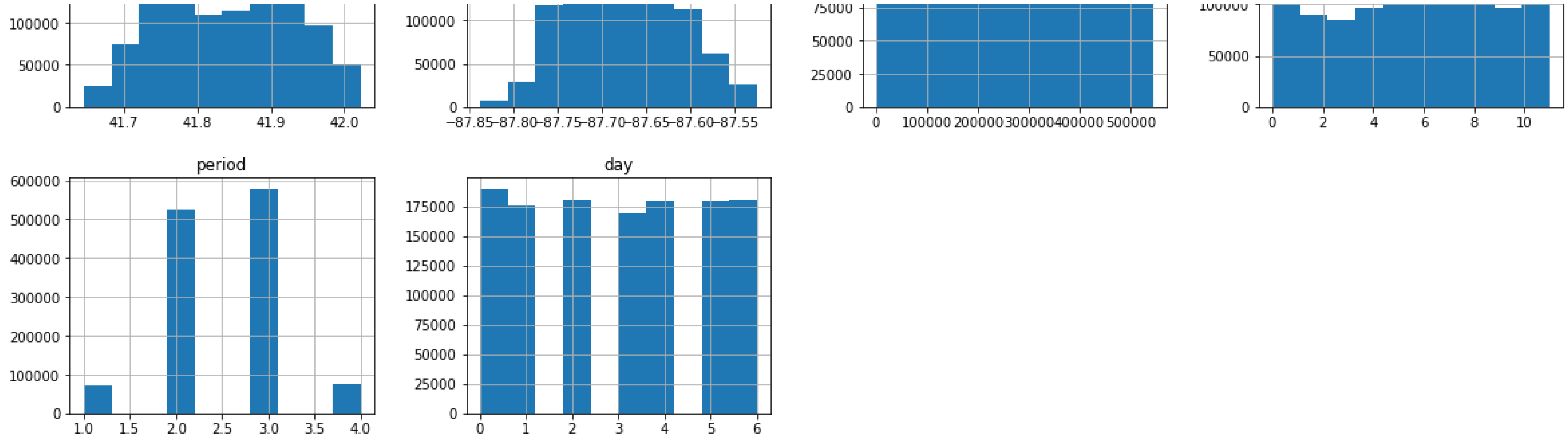
In [146]:

```
# univariate

import matplotlib.pyplot as plt

data.hist()
plt.gcf().set_size_inches(20,20)
plt.show()
```




100000
50000
0
41.7
41.8
41.9
42.0
100000
50000
0
−87.85
−87.80
−87.75
−87.70
−87.65
−87.60
−87.55
75000
50000
25000
0
0
100000
200000
300000
400000
500000
100000
50000
0
0
2
4
6
8
10
period
600000
500000
400000
300000
200000
100000
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
day
175000
150000
125000
100000
75000
50000
25000
0
0
1
2
3
4
5
6

```
In [147]: data.describe().T
```

```
Out[147]:
                          count           mean            std            min            25%            50%            75%            max
        Primary Type  1252041.0   1.675805e+01      12.223026   0.000000e+00   6.000000e+00   1.700000e+01   3.000000e+01   3.400000e+01
         Description  1252041.0   2.888157e+02     157.418869   0.000000e+00   1.480000e+02   3.340000e+02   4.220000e+02   5.120000e+02
            Location
         Description  1252041.0   1.589880e+02      15.787014   1.060000e+02   1.450000e+02   1.660000e+02   1.710000e+02   1.970000e+02
              Arrest  1252041.0   2.684744e-01       0.443166   0.000000e+00   0.000000e+00   0.000000e+00   1.000000e+00   1.000000e+00
            Domestic  1252041.0   0.000000e+00       0.000000   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00
                Beat  1252041.0   1.208138e+03     696.372231   1.110000e+02   6.310000e+02   1.113000e+03   1.732000e+03   2.535000e+03
            District  1252041.0   9.933527e+00       6.090848   0.000000e+00   5.000000e+00   9.000000e+00   1.500000e+01   2.200000e+01
                Ward  1252041.0   2.100356e+01      14.575182   0.000000e+00   8.000000e+00   2.100000e+01   3.200000e+01   5.000000e+01
      Community Area  1252041.0   3.395031e+01      23.044400   0.000000e+00   1.700000e+01   2.900000e+01   5.300000e+01   7.700000e+01
        X Coordinate  1252041.0   1.164286e+06   15664.644659   1.118919e+06   1.152698e+06   1.165365e+06   1.175652e+06   1.205119e+06
        Y Coordinate  1252041.0   1.886036e+06   31498.042822   1.813894e+06   1.859274e+06   1.891464e+06   1.909992e+06   1.951622e+06
                Year  1252041.0   2.009299e+03       5.884196   2.001000e+03   2.004000e+03   2.008000e+03   2.014000e+03   2.022000e+03
            Latitude  1252041.0   4.184289e+01       0.086617   4.164459e+01   4.176921e+01   4.185780e+01   4.190884e+01   4.202291e+01
           Longitude  1252041.0  -8.767265e+01       0.056993  -8.783807e+01  -8.771471e+01  -8.766852e+01  -8.763123e+01  -8.752453e+01
            Location  1252041.0   2.667257e+05  147704.129911   1.000000e+00   1.405400e+05   2.764410e+05   3.797280e+05   5.451640e+05
               month  1252041.0   5.586907e+00       3.457462   0.000000e+00   3.000000e+00   6.000000e+00   9.000000e+00   1.100000e+01
              period  1252041.0   2.523199e+00       0.696131   1.000000e+00   2.000000e+00   3.000000e+00   3.000000e+00   4.000000e+00
                 day  1252041.0   2.981380e+00       2.018929   0.000000e+00   1.000000e+00   3.000000e+00   5.000000e+00   6.000000e+00
```

```
In [148]: data.shape
```

```
Out[148]: (1252041, 18)
```

In [201]:

```
# day_type as a target variable feature
columns = ['Primary Type', 'Description', 'Location Description','Arrest','Beat','Ward',
           'Community Area','X Coordinate','Y Coordinate','Year','Location','month','day']
day_data = data[columns]

# period as a target variable feature
columns =  ['Primary Type', 'Description', 'Location Description','Arrest','Beat','Ward',
           'Community Area','X Coordinate','Y Coordinate','Year','Location','month','period']
period_data = data[columns]

# District as a target variable feature
columns = ['Primary Type', 'Description', 'Location Description','Arrest','Beat','Ward',
           'Community Area','X Coordinate','Y Coordinate','Year','Location','month','District']
district_data = data[columns]
```

In [151]:

```
day_data.columns
```

Out[151]:

```
Index(['Primary Type', 'Description', 'Location Description', 'Arrest', 'Beat',
       'Ward', 'Community Area', 'X Coordinate', 'Y Coordinate', 'Year',
       'Location', 'month', 'day'],
      dtype='object')
```

In [152]:

```
day_data.head()
```

Out[152]:

	Primary Type	Description	Location Description	Arrest	Beat	Ward	Community Area	X Coordinate	Y Coordinate	Year	Location	month	day
1	34	468	139	1	413	8	47	1184499.0	1843935.0	2018	59718	5	5
2	3	216	145	1	1711	39	12	1146911.0	1941022.0	2007	525966	0	0
3	1	421	125	0	1115	28	26	1148388.0	1899882.0	2018	316579	5	1
4	1	25	134	0	1231	27	28	1165430.0	1897441.0	2021	299524	5	3
5	33	334	171	0	2113	0	0	1174343.0	1885951.0	2001	256235	1	5

In [283]:

```
# data transformation using scikit learn MinMaxScaler to Rescale

from sklearn.preprocessing import MinMaxScaler

array = day_data.values
x = array[:,0:12]
y = array[:,12]

model = MinMaxScaler(feature_range= (0, 1))
rescaledx = model.fit_transform(x)

rescaledx_data = pd.DataFrame(rescaledx)
rescaledx_data.columns = ['Primary Type', 'Description', 'Location Description', 'Arrest', 'Beat',
                          'Ward', 'Community Area', 'X Coordinate', 'Y Coordinate', 'Year', 'Location', 'mo
```

In [284]:

```
rescaledx_data['day'] = day_data['day']
rescaledx_data['day'] = rescaledx_data['day'].fillna(0)
```

In [263]:

```
rescaledx_data.isna().sum()
```

Out[263]:

```
Primary Type            0
Description             0
Location Description    0
Arrest                  0
Beat                    0
Ward                    0
Community Area          0
X Coordinate            0
Y Coordinate            0
Year                    0
Location                0
month                   0
day                     0
dtype: int64
```

In [ ]:

```

```

```
In [264]: rescaledx_data.shape
```

```
Out[264]: (1252041, 13)
```

```
In [63]: ndf = rescaledx_data.iloc[0:80000, 0:13]
```

```
In [357]: ndf.shape
```

```
Out[357]: (80000, 13)
```

```
In [157]: rescaledx_data.describe().T
```

Out[157]:

	count	mean	std	min	25%	50%	75%	max
Primary Type	1252041.0	0.492884	0.359501	0.0	0.176471	0.500000	0.882353	1.0
Description	1252041.0	0.564093	0.307459	0.0	0.289062	0.652344	0.824219	1.0
Location Description	1252041.0	0.582286	0.173484	0.0	0.428571	0.659341	0.714286	1.0
Arrest	1252041.0	0.268474	0.443166	0.0	0.000000	0.000000	1.000000	1.0
Beat	1252041.0	0.452615	0.287282	0.0	0.214521	0.413366	0.668729	1.0
Ward	1252041.0	0.420071	0.291504	0.0	0.160000	0.420000	0.640000	1.0
Community Area	1252041.0	0.440913	0.299278	0.0	0.220779	0.376623	0.688312	1.0
X Coordinate	1252041.0	0.526297	0.181724	0.0	0.391868	0.538817	0.658155	1.0
Y Coordinate	1252041.0	0.523799	0.228697	0.0	0.329490	0.563212	0.697738	1.0
Year	1252041.0	0.395188	0.280200	0.0	0.142857	0.333333	0.619048	1.0
Location	1252041.0	0.489257	0.270936	0.0	0.257793	0.507078	0.696538	1.0
month	1252041.0	0.507901	0.314315	0.0	0.272727	0.545455	0.818182	1.0
day	1252041.0	1.637712	2.107257	0.0	0.000000	0.000000	3.000000	6.0

In [358]:

```python
# from pandas import read_csv
# from sklearn.feature_selection import RFE
# from sklearn.linear_model import LogisticRegression

# array = ndf.values
# X = array[:, 0:12]
# Y = array[:, 12]

# # feature extraction
# model = LogisticRegression(solver= 'lbfgs', max_iter= 50000)
# rfe = RFE(model, 8)
# fit = rfe.fit(X, Y)

# # print("Num Features: {}".format(fit.n_features_))
# # print("Selected Features: {}".format(fit.support_))
# print("Feature Ranking: {}".format(fit.ranking_))
```

```
C:\Users\Shehu\anaconda3\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass n_features_to_s
elect=8 as keyword args. From version 1.0 (renaming of 0.25) passing these as positional arguments will result
in an error
  warnings.warn(f"Pass {args_msg} as keyword args. From version "

Feature Ranking: [1 1 1 1 4 2 5 1 1 1 1 3]
```

In [274]:

```
# Univeriate selection

from numpy import set_printoptions
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

array = rescaledx_data.values
x = array[:, 0:12]
y = array[:, 12]

test = SelectKBest(score_func=chi2, k = 8)
fit = test.fit(x, y)

set_printoptions(precision = 3)
print(fit.scores_)
features = fit.transform(x)
print(features[0:2, :])
```

```
[2.038 1.567 0.622 5.64  0.238 1.084 0.676 0.029 0.267 1.4   0.436 2.22 ]
[[1.    0.914 0.363 1.    0.16  0.61  0.81  0.455]
 [0.088 0.422 0.429 1.    0.78  0.156 0.286 0.   ]]
```

In [275]:

```python
dfeature = pd.DataFrame(features, columns=['a','b','c','d','e','f','g','h'])
col_dict = dict()
cols = list(rescaledx_data.columns)[0:-1]

for col in cols:
    dcol = rescaledx_data[col].values.tolist()
    col_dict[col] = dcol

count = 0
col_len = len(dfeature.columns.tolist())

while count < col_len:
    col_label = ''
    if count == 0:
        col_label = 'a'
    elif count == 1:
        col_label = 'b'
    elif count == 2:
        col_label = 'c'
    elif count == 3:
        col_label = 'd'
    elif count == 4:
        col_label = 'e'
    elif count == 5:
        col_label = 'f'
    elif count == 6:
        col_label = 'g'
    elif count == 7:
        col_label = 'h'
    else:
        pass

    coldata = dfeature.loc[:, col_label]
    coldatav = list(coldata.values)

    for item in col_dict.items():
        if coldatav == item[1]:
            print(item[0])

    count += 1
```

```
Primary Type
Description
```

```
Location Description
Arrest
Ward
Community Area
Year
month
```

In [279]:

```python
dataframe = rescaledx_data.loc[:, ['Primary Type', 'Description','Arrest',
                                   'Community Area','Year', 'month', 'day']]
```

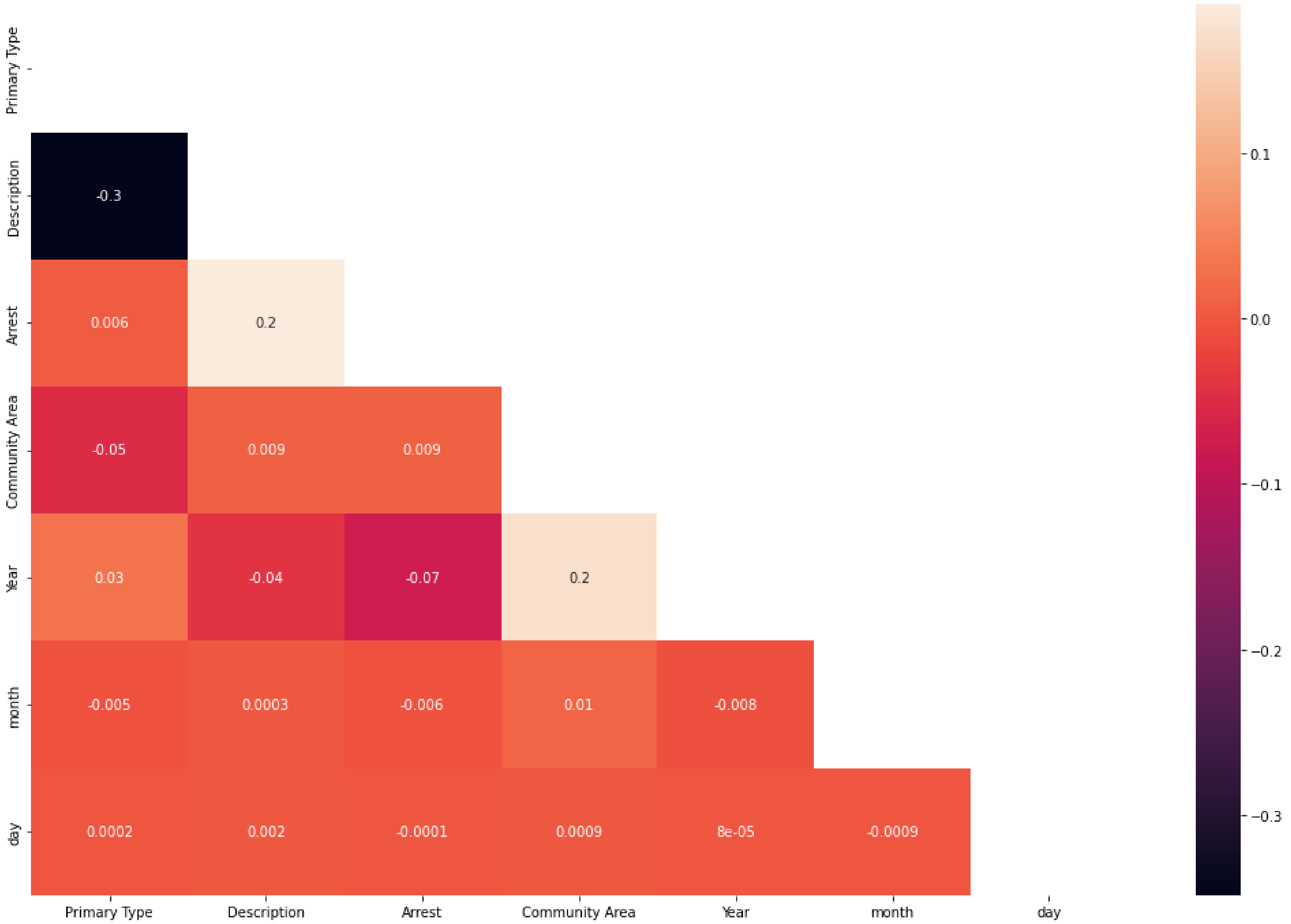
In [280]:

```python
import numpy as np

matrix = np.triu(dataframe.corr())
plt.figure(figsize = (18, 12))
sns.heatmap(dataframe.corr(), annot = True, mask = matrix, fmt='.1g')
```

Out[280]:

```
<AxesSubplot:>
```



In [ ]:

```
import seaborn as sns
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize= (12,7))
sns.countplot(dataframe['day'])
```

In [ ]:

```

```

In [271]:

```python
# Evaluate using Cross Validation
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

array = dataframe.values
x = array[:, 0:6]
y = array[:, 6]

num_folds = 10
seed = 7
kfold = KFold(n_splits=num_folds)
model = LogisticRegression()
results = cross_val_score(model, X, Y, cv=kfold)

print("Accuracy: {}.3f%%, {}.3f%%".format(results.mean()*100.0, results.std()*100.0))
```

```
Accuracy: 45.103954295633095.3f%%, 0.08728157932558331.3f%%
```

In [282]:

```python
# Evaluate using a train and a test set
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

array = dataframe.values
x = array[:, 0:6]
y = array[:, 6]

test_size = 0.33
seed = 10
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=seed)

model = LogisticRegression()
model.fit(x_train, y_train)
result = model.score(x_test, y_test)

print("Accuracy: {}".format(result * 100.0))
```

```
Accuracy: 52.46796749069399
```

In [ ]:

```

```

In [ ]:

```python
# models
```

In [164]:

```python
# CART Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier

array = dataframe.values
x = array[:, 0:8]
y = array[:, 8]

kfold = KFold(n_splits=10)
model = DecisionTreeClassifier()
results = cross_val_score(model, x, y, cv=kfold)

print(results.mean())
```

```
0.30925345087007694
```

In [198]:

```python
# Linear Discriminant Aanaalysis

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

array = dataframe.values
x = array[:, 0:8]
y = array[:, 8]

num_folds = 10
kfold = KFold(n_splits= num_folds, shuffle= True, random_state= 7)
model = LinearDiscriminantAnalysis()
results = cross_val_score(model, x, y, cv= kfold)

print(results.mean())
```

```
0.5339010489874335
```

In [73]:

```python
# Gaussian Naive Bayes Classification
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB

array = dataframe.values
X = array[:, 0:7]
Y = array[:, 7]

kfold = KFold(n_splits=10)
model = GaussianNB()
results = cross_val_score(model, x, y, cv = kfold)

print(results.mean())
```

```
0.5338858519667504
```

In [74]:

```python
# LogissticRegression Classification

import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

array = dataframe.values
x = array[:, 0:8]
y = array[:, 8]

num_folds = 10
test_size = 0.33
seed = 7

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= test_size, random_state= seed)
kfold = KFold(n_splits= num_folds)
model = LogisticRegression(max_iter= 500)
results = cross_val_score(model, x, y, cv= kfold)
score = np.mean(results)
y_pred = cross_val_predict(model, x, y, cv= 3)
con_m = confusion_matrix(y, y_pred)
my_model = model.fit(x_train, y_train)
predicted = my_model.predict(x_test)
report = classification_report(y_test, predicted)

print(results)
print()
print('The mean score is', score * 100)
print()
print(con_m)
print()
print(report)
```

```
C:\Users\Shehu\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
C:\Users\Shehu\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
[0.534 0.535 0.533 0.533 0.533 0.534 0.535 0.534 0.532 0.535]

The mean score is 53.388585196675045

[[668471      0      0      0      0      0      0]
 [ 96561      0      0      0      0      0      0]
 [ 98846      0      0      0      0      0      0]
 [ 92614      0      0      0      0      0      0]
 [ 98470      0      0      0      0      0      0]
 [ 98095      0      0      0      0      0      0]
 [ 99029      0      0      0      0      0      0]]

              precision    recall  f1-score   support

         0.0       0.53      1.00      0.70    220414
         1.0       0.00      0.00      0.00     32067
         2.0       0.00      0.00      0.00     32450
         3.0       0.00      0.00      0.00     30577
         4.0       0.00      0.00      0.00     32648
         5.0       0.00      0.00      0.00     32215
         6.0       0.00      0.00      0.00     32818

    accuracy                           0.53    413189
   macro avg       0.08      0.14      0.10    413189
weighted avg       0.28      0.53      0.37    413189
```

```
C:\Users\Shehu\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Pr
ecision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divisi
on` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [78]:

```
# cm = con_m
# labls = ['N', 'Y']
# sns.set(font_scale= 2)
# cmn = cm.astype('float') / cm.sum(axis= 1)[:, np.newaxis]
# fig, ax = plt.subplots(figsize= (20, 12))
# sns.heatmap(cmn, annot= True, fmt='.2f', xticklabels= labls, yticklabels=labls)
# plt.ylabel('Actual COF ', size= 20)
# plt.xlabel('Predicted COF', size= 20)
# plt.title('Confusion Matrix of Prediction', size= 24)

# # plt.figure(figsize= (16, 8))
# plt.show(block=False)
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
# period as a target variable
```

In [165]:

```
period_data.head()
```

Out[165]:

	Primary Type	Description	Location Description	Arrest	Beat	Ward	Community Area	X Coordinate	Y Coordinate	Year	Location	month	period
1	34	468	139	1	413	8	47	1184499.0	1843935.0	2018	59718	5	2
2	3	216	145	1	1711	39	12	1146911.0	1941022.0	2007	525966	0	2
3	1	421	125	0	1115	28	26	1148388.0	1899882.0	2018	316579	5	2
4	1	25	134	0	1231	27	28	1165430.0	1897441.0	2021	299524	5	2
5	33	334	171	0	2113	0	0	1174343.0	1885951.0	2001	256235	1	2

In [166]:

```
period_data.columns
```

Out[166]:

```
Index(['Primary Type', 'Description', 'Location Description', 'Arrest', 'Beat',
       'Ward', 'Community Area', 'X Coordinate', 'Y Coordinate', 'Year',
       'Location', 'month', 'period'],
      dtype='object')
```

In [224]:

```
# data transformation using scikit learn MinMaxScaler to Rescale

from sklearn.preprocessing import MinMaxScaler

array = period_data.values
x = array[:,0:12]
y = array[:,12]

model = MinMaxScaler(feature_range= (0, 1))
rescaledx = model.fit_transform(x)

rescaledx_data = pd.DataFrame(rescaledx)
rescaledx_data.columns = ['Primary Type', 'Description', 'Location Description', 'Arrest', 'Beat',
                          'Ward', 'Community Area', 'X Coordinate', 'Y Coordinate', 'Year','Location', 'mon
```

In [225]:

```
rescaledx_data['period'] = period_data['period']
rescaledx_data['period'] = rescaledx_data['period'].fillna(rescaledx_data['period'].median()).astype('int64
```

In [226]:

```
rescaledx_data.isna().sum()
```

Out[226]:

```
Primary Type            0
Description             0
Location Description    0
Arrest                  0
Beat                    0
Ward                    0
Community Area          0
X Coordinate            0
Y Coordinate            0
Year                    0
Location                0
month                   0
period                  0
dtype: int64
```

In [227]:

```
# Univeriate selection

from numpy import set_printoptions
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

array = rescaledx_data.values
x = array[:, 0:12]
y = array[:, 12]

test = SelectKBest(score_func=chi2, k = 9)
fit = test.fit(x, y)

set_printoptions(precision = 3)
print(fit.scores_)
features = fit.transform(x)
print(features[0:2, :])
```

```
[0.786 0.458 0.057 1.238 0.442 1.123 0.137 0.2   0.245 1.022 0.383 0.219]
[[1.    0.914 1.    0.125 0.16  0.218 0.81  0.11  0.455]
 [0.088 0.422 1.    0.66  0.78  0.923 0.286 0.965 0.   ]]
```

In [228]:

```
dfeature = pd.DataFrame(features, columns=['a','b','c','d','e','f','g','h','i'])
col_dict = dict()
cols = list(rescaledx_data.columns)[0:-1]

for col in cols:
    dcol = rescaledx_data[col].values.tolist()
    col_dict[col] = dcol

count = 0
col_len = len(dfeature.columns.tolist())

while count < col_len:
    col_label = ''
    if count == 0:
        col_label = 'a'
    elif count == 1:
        col_label = 'b'
    elif count == 2:
        col_label = 'c'
    elif count == 3:
        col_label = 'd'
    elif count == 4:
        col_label = 'e'
    elif count == 5:
        col_label = 'f'
    elif count == 6:
        col_label = 'g'
    elif count == 7:
        col_label = 'h'
    elif count == 8:
        col_label = 'i'
    else:
        pass

    coldata = dfeature.loc[:, col_label]
    coldatav = list(coldata.values)

    for item in col_dict.items():
        if coldatav == item[1]:
            print(item[0])

    count += 1
```

```
Primary Type
Description
Arrest
Beat
Ward
Y Coordinate
Year
Location
month
```

In [257]:

```
dataframe1 = rescaledx_data.loc[:, ['Primary Type','Description','Arrest','Beat',
                                    'Year', 'month', 'period']]
```

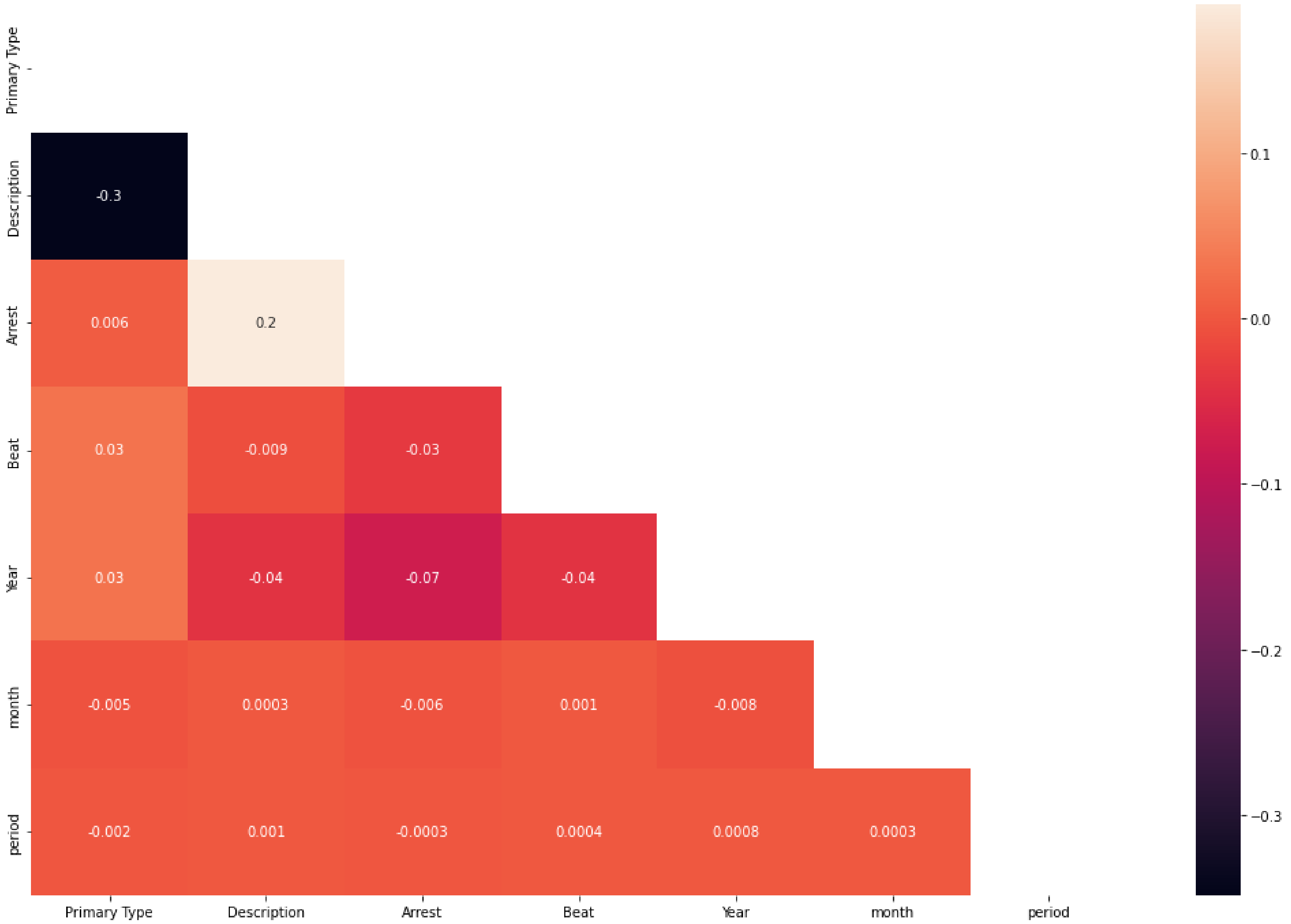
In [258]:

```python
import numpy as np

matrix = np.triu(dataframe1.corr())
plt.figure(figsize = (18, 12))
sns.heatmap(dataframe1.corr(), annot = True, mask = matrix, fmt='.1g')
```

Out[258]:

```
<AxesSubplot:>
```


Primary Type
Description
Arrest
Beat
Year
month
period
-0.3
0.006
0.2
0.03
-0.009
-0.03
0.03
-0.04
-0.07
-0.04
-0.005
0.0003
-0.006
0.001
-0.008
-0.002
0.001
-0.0003
0.0004
0.0008
0.0003
Primary Type
Description
Arrest
Beat
Year
month
period
0.1
0.0
−0.1
−0.2
−0.3

```
In [ ]:
```

In [259]:

```
# Evaluate using a train and a test set
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

array = dataframe1.values
x = array[:, 0:6]
y = array[:, 6]

test_size = 0.33
seed = 10
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=seed)

model = LogisticRegression()
model.fit(x_train, y_train)
result = model.score(x_test, y_test)

print("Accuracy: {}".format(result * 100.0))
```

```
Accuracy: 70.49257697725415
```

In [ ]:

```

```

In [241]:

```python
# Using Random Forest for classification
import sklearn.metrics as metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# from sklearn.externals import joblib

array = dataframe1.values
x = array[:, 0:7]
y = array[:, 7]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 10 )

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 10)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

print("Accuracy:",(metrics.accuracy_score(y_test, y_pred)*100),"\n")

cm = pd.crosstab(y_test, y_pred, rownames=['Actual Alarm'], colnames=['Predicted Alarm'])
print("\n----------Confusion Matrix------------------------------------")
print(cm)

print("\n----------Classification Report------------------------------------")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 58.47398334243844


----------Confusion Matrix------------------------------------
Predicted Alarm   1.0    2.0     3.0   4.0
Actual Alarm
1.0               210   2013    7721   176
2.0              1494  14065   55403  1275
3.0              4606  43560  168568  3775
```

```
4.0               215   1942    7801   187

----------Classification Report-----------------------------------
              precision    recall  f1-score   support

         1.0       0.03      0.02      0.03     10120
         2.0       0.23      0.19      0.21     72237
         3.0       0.70      0.76      0.73    220509
         4.0       0.03      0.02      0.02     10145

    accuracy                           0.58    313011
   macro avg       0.25      0.25      0.25    313011
weighted avg       0.55      0.58      0.57    313011
```

In [245]:

```python
# Gaussian Naive Bayes Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB

array = dataframe1.values
x = array[:, 0:6]
y = array[:, 6]

kfold = KFold(n_splits=10)
model = GaussianNB()
results = cross_val_score(model, x, y, cv = kfold)

print(results.mean())
```

```
0.7047333113287193
```

In [189]:

```python
# CART Classification

from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier

array = dataframe1.values
X = array[:, 0:6]
Y = array[:, 6]

kfold = KFold(n_splits=10)
model = DecisionTreeClassifier()
results = cross_val_score(model, x, y, cv=kfold)

print(results.mean())
```

```
0.3196293095967449
```

In [190]:

```python
from sklearn.tree import DecisionTreeClassifier

array = dataframe1.values
X = array[:, 0:6]
Y = array[:, 6]

test_size = 0.33
seed = 10
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=seed)

model = DecisionTreeClassifier()
model.fit(x,y)

score = model.score(x_test, y_test)
pred = model.predict(x_test)

print(pred)
```

In [233]:

```python
# Linear Discriminant Aanaalysis

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

array = dataframe1.values
x = array[:, 0:9]
y = array[:, 9]

num_folds = 10
kfold = KFold(n_splits= num_folds, shuffle= True, random_state= 7)
model = LinearDiscriminantAnalysis()
results = cross_val_score(model, x, y, cv= kfold)

print(results.mean())
```

```
0.7047333125407503
```

In [ ]:

```python

```

In [ ]:

```python
# district as a target variable
```

In [203]:

```python
district_data.head()
```

Out[203]:

	Primary Type	Description	Location Description	Arrest	Beat	Ward	Community Area	X Coordinate	Y Coordinate	Year	Location	month	District
1	34	468	139	1	413	8	47	1184499.0	1843935.0	2018	59718	5	3
2	3	216	145	1	1711	39	12	1146911.0	1941022.0	2007	525966	0	15
3	1	421	125	0	1115	28	26	1148388.0	1899882.0	2018	316579	5	10
4	1	25	134	0	1231	27	28	1165430.0	1897441.0	2021	299524	5	11
5	33	334	171	0	2113	0	0	1174343.0	1885951.0	2001	256235	1	1

In [202]:

```python
district_data.columns
```

Out[202]:

```
Index(['Primary Type', 'Description', 'Location Description', 'Arrest', 'Beat',
       'Ward', 'Community Area', 'X Coordinate', 'Y Coordinate', 'Year',
       'Location', 'month', 'District'],
      dtype='object')
```

In [204]:

```
# data transformation using scikit learn MinMaxScaler to Rescale

from sklearn.preprocessing import MinMaxScaler

array = district_data.values
x = array[:,0:12]
y = array[:,12]

model = MinMaxScaler(feature_range= (0, 1))
rescaledx = model.fit_transform(x)

rescaledx_data = pd.DataFrame(rescaledx)
rescaledx_data.columns = ['Primary Type', 'Description', 'Location Description', 'Arrest', 'Beat',
                          'Ward', 'Community Area', 'X Coordinate', 'Y Coordinate', 'Year',
                          'Location', 'month']
```

In [ ]:

```

```

In [207]:

```
rescaledx_data['District'] = district_data['District']
rescaledx_data['District'] = rescaledx_data['District'].fillna(0)
```

In [ ]:

```
# Univeriate selection

from numpy import set_printoptions
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

array = rescaledx_data.values
x = array[:, 0:12]
y = array[:, 12]

test = SelectKBest(score_func=chi2, k = 9)
fit = test.fit(x, y)

set_printoptions(precision = 3)
print(fit.scores_)
features = fit.transform(x)
print(features[0:2, :])
```

In [ ]:

```
dfeature = pd.DataFrame(features, columns=['a','b','c','d','e','f','g','h','i'])
col_dict = dict()
cols = list(rescaledx_data.columns)[0:-1]

for col in cols:
    dcol = rescaledx_data[col].values.tolist()
    col_dict[col] = dcol

count = 0
col_len = len(dfeature.columns.tolist())

while count < col_len:
    col_label = ''
    if count == 0:
        col_label = 'a'
    elif count == 1:
        col_label = 'b'
    elif count == 2:
        col_label = 'c'
    elif count == 3:
        col_label = 'd'
    elif count == 4:
        col_label = 'e'
    elif count == 5:
        col_label = 'f'
    elif count == 6:
        col_label = 'g'
    elif count == 7:
        col_label = 'h'
    elif count == 8:
        col_label = 'i'
    else:
        pass

    coldata = dfeature.loc[:, col_label]
    coldatav = list(coldata.values)

    for item in col_dict.items():
        if coldatav == item[1]:
            print(item[0])

    count += 1
```

In [ ]:

```
dataframe2 = rescaledx_data.loc[:, ['Primary Type','Description','Arrest',
                                    'Year','Location', 'month', 'period']]
```

In [ ]:

```
import numpy as np

matrix = np.triu(dataframe2.corr())
plt.figure(figsize = (18, 12))
sns.heatmap(dataframe1.corr(), annot = True, mask = matrix, fmt='.1g')
```

In [ ]:

```
# Evaluate using a train and a test set
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

array = dataframe2.values
x = array[:, 0:6]
y = array[:, 6]

test_size = 0.33
seed = 10
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=seed)

model = LogisticRegression()
model.fit(x_train, y_train)
result = model.score(x_test, y_test)

print("Accuracy: {}".format(result * 100.0))
```

In [ ]:

```

```

In [ ]:

```python
# CART Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier

array = dataframe1.values
X = array[:, 0:6]
Y = array[:, 6]

kfold = KFold(n_splits=10)
model = DecisionTreeClassifier()
results = cross_val_score(model, x, y, cv=kfold)

print(results.mean())
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [249]:

In [ ]:

In [ ]:

In [ ]:

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [17]:
```

```
In [ ]:
```