Import all the necessary libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from encodings.aliases import aliases
```

Read in the csv files and remove duplicates

In [2]:
```
#to find encoding that works

alias_values = set(aliases.values())

for encoding in set(aliases.values()):
    try:
        df = pd.read_csv("crime.csv", nrows=10, encoding=encoding)
        print('Successful : ', encoding)

    except:
        pass
```

```
Successful :  cp037
Successful :  iso8859_9
Successful :  cp869
Successful :  cp500
Successful :  iso8859_5
Successful :  cp273
Successful :  iso8859_16
Successful :  cp1125
Successful :  cp864
Successful :  cp857
Successful :  kz1048
Successful :  cp1253
Successful :  latin_1
Successful :  iso8859_10
Successful :  cp861
Successful :  big5hkscs
Successful :  cp1251
Successful :  cp1140
Successful :  cp852
Successful :  cp850
Successful :  cp1255
Successful :  cp860
Successful :  cp932
Successful :  cp855
Successful :  iso8859_2
Successful :  cp1258
Successful :  iso8859_8
Successful :  iso8859_14
Successful :  iso8859_3
Successful :  mac_iceland
Successful :  cp862
Successful :  ptcp154
Successful :  cp437
Successful :  mac_roman
Successful :  cp866
Successful :  cp1026
Successful :  mac_turkish
Successful :  iso8859_6
Successful :  iso8859_13
Successful :  utf_16_be
Successful :  cp775
Successful :  cp863
Successful :  cp1252
```

```
Successful :  utf_16_le
Successful :  cp1257
Successful :  mac_cyrillic
Successful :  mac_latin2
Successful :  cp1256
Successful :  hp_roman8
Successful :  cp1250
Successful :  cp858
Successful :  cp865
Successful :  gbk
Successful :  iso8859_4
Successful :  mac_greek
Successful :  gb18030
Successful :  iso8859_11
Successful :  iso8859_15
Successful :  koi8_r
Successful :  cp1254
Successful :  iso8859_7
Successful :  mbcs
Successful :  cp949
```

In [3]:
```
#Reading the CSV files

df = pd.read_csv('crime.csv', encoding='ISO-8859-11')

df.head() #checking how dataset is looking
```

Out[3]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCURRED
0	I182070945	619	Larceny	LARCENY ALL OTHERS	D14	808	NaN	2018-09-(
1	I182070943	1402	Vandalism	VANDALISM	C11	347	NaN	2018-08-2
2	I182070941	3410	Towed	TOWED MOTOR VEHICLE	D4	151	NaN	2018-09-(
3	I182070940	3114	Investigate Property	INVESTIGATE PROPERTY	D4	272	NaN	2018-09-(
4	I182070938	3114	Investigate Property	INVESTIGATE PROPERTY	B3	421	NaN	2018-09-(

In [4]:
```
df.shape #checking the shape of data
```

Out[4]: (319073, 17)

In [5]:
```
df.duplicated().sum() #checking how many duplicates rows are there.
```

Out[5]: 23

In [6]:
```
df.drop_duplicates(inplace=True) #Dropping all the duplicate rows
```

In [7]:
```
df.shape #checking the shape again to see if the duplicate rows worked
```

Out[7]: (319050, 17)

Explore the data set

In [8]:
```
df.head() #checking to see how data is looking in the beginning
```

Out[8]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCURRED
0	I182070945	619	Larceny	LARCENY ALL OTHERS	D14	808	NaN	2018-09-(
1	I182070943	1402	Vandalism	VANDALISM	C11	347	NaN	2018-08-2
2	I182070941	3410	Towed	TOWED MOTOR VEHICLE	D4	151	NaN	2018-09-(
3	I182070940	3114	Investigate Property	INVESTIGATE PROPERTY	D4	272	NaN	2018-09-(
4	I182070938	3114	Investigate Property	INVESTIGATE PROPERTY	B3	421	NaN	2018-09-(

In [9]:
```
df.tail() #checking to see how data is looking in the end.
```

Out[9]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCU
319068	I050310906-00	3125	Warrant Arrests	WARRANT ARREST	D4	285	NaN	201
319069	I030217815-08	111	Homicide	MURDER, NON-NEGLIGIENT MANSLAUGHTER	E18	520	NaN	201
319070	I030217815-08	3125	Warrant Arrests	WARRANT ARREST	E18	520	NaN	201
319071	I010370257-00	3125	Warrant Arrests	WARRANT ARREST	E13	569	NaN	201
319072	142052550	3125	Warrant Arrests	WARRANT ARREST	D4	903	NaN	201

In [10]:
```
df.info() #Summary info about the dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 319050 entries, 0 to 319072
Data columns (total 17 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   INCIDENT_NUMBER      319050 non-null  object 
 1   OFFENSE_CODE         319050 non-null  int64  
 2   OFFENSE_CODE_GROUP   319050 non-null  object 
 3   OFFENSE_DESCRIPTION  319050 non-null  object 
 4   DISTRICT             317285 non-null  object 
 5   REPORTING_AREA       319050 non-null  object 
 6   SHOOTING             1019 non-null    object 
 7   OCCURRED_ON_DATE     319050 non-null  object 
 8   YEAR                 319050 non-null  int64  
 9   MONTH                319050 non-null  int64  
 10  DAY_OF_WEEK          319050 non-null  object 
 11  HOUR                 319050 non-null  int64  
 12  UCR_PART             318960 non-null  object 
 13  STREET               308179 non-null  object 
 14  Lat                  299052 non-null  float64
 15  Long                 299052 non-null  float64
 16  Location             319050 non-null  object 
dtypes: float64(2), int64(4), object(11)
memory usage: 43.8+ MB
```

In [11]:
```
df.OCCURRED_ON_DATE = pd.to_datetime(df.OCCURRED_ON_DATE)
#changing the data type of OCCURRED_ON_DATE column from object to date time.
```

In [12]:
```
df.info() #checking data types again to see if it is working
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 319050 entries, 0 to 319072
Data columns (total 17 columns):
 #   Column               Non-Null Count   Dtype         
---  ------               --------------   -----         
 0   INCIDENT_NUMBER      319050 non-null  object        
 1   OFFENSE_CODE         319050 non-null  int64         
 2   OFFENSE_CODE_GROUP   319050 non-null  object        
 3   OFFENSE_DESCRIPTION  319050 non-null  object        
 4   DISTRICT             317285 non-null  object        
 5   REPORTING_AREA       319050 non-null  object        
 6   SHOOTING             1019 non-null    object        
 7   OCCURRED_ON_DATE     319050 non-null  datetime64[ns]
 8   YEAR                 319050 non-null  int64         
 9   MONTH                319050 non-null  int64         
 10  DAY_OF_WEEK          319050 non-null  object        
 11  HOUR                 319050 non-null  int64         
 12  UCR_PART             318960 non-null  object        
 13  STREET               308179 non-null  object        
 14  Lat                  299052 non-null  float64       
 15  Long                 299052 non-null  float64       
 16  Location             319050 non-null  object        
dtypes: datetime64[ns](1), float64(2), int64(4), object(10)
memory usage: 43.8+ MB
```

In [13]:
```
#Extracting date time from the columns column

df.OCCURRED_ON_DATE.dt.year
```

Out[13]:
```
0         2018
1         2018
2         2018
3         2018
4         2018
          ...
319068    2016
319069    2015
319070    2015
319071    2016
319072    2015
Name: OCCURRED_ON_DATE, Length: 319050, dtype: int64
```

In [14]:
```
df.OCCURRED_ON_DATE.dt.month
```

Out[14]:
```
0         9
1         8
2         9
3         9
4         9
         ..
319068    6
319069    7
319070    7
319071    5
319072    6
Name: OCCURRED_ON_DATE, Length: 319050, dtype: int64
```

In [15]:
```
df.OCCURRED_ON_DATE.dt.week
```

```
C:\Users\arjun\AppData\Local\Temp\ipykernel_35460\1330590546.py:1: FutureWarning: Series.dt.weekofyear and Series.dt.week have been deprecated. Please use Series.dt.isocalendar().week instead.
  df.OCCURRED_ON_DATE.dt.week
```

Out[15]:
```
0         35
1         34
2         36
3         36
4         36
          ..
319068    22
319069    28
319070    28
319071    22
319072    26
Name: OCCURRED_ON_DATE, Length: 319050, dtype: int64
```

In [16]:
```
df.OCCURRED_ON_DATE.dt.day
```

Out[16]:
```
0          2
1         21
2          3
3          3
4          3
          ..
319068     5
319069     9
319070     9
319071    31
319072    22
Name: OCCURRED_ON_DATE, Length: 319050, dtype: int64
```

In [17]:
```
df.OCCURRED_ON_DATE.dt.hour
```

Out[17]:
```
0         13
1          0
2         19
3         21
4         21
          ..
319068    17
319069    13
319070    13
319071    19
319072     0
Name: OCCURRED_ON_DATE, Length: 319050, dtype: int64
```

In [18]:
```
df.OCCURRED_ON_DATE.dt.minute
```

Out[18]:
```
0          0
1          0
2         27
3         16
4          5
          ..
319068    25
319069    38
319070    38
319071    35
319072    12
Name: OCCURRED_ON_DATE, Length: 319050, dtype: int64
```

In [19]:
```
#summary information about numeric data

df.describe()
```

Out[19]:

	OFFENSE_CODE	YEAR	MONTH	HOUR	Lat	Long
count	319050.000000	319050.000000	319050.000000	319050.000000	299052.000000	299052.000000
mean	2317.516957	2016.560674	6.609622	13.118176	42.214373	-70.908260
std	1185.308921	0.996312	3.273677	6.294258	2.159845	3.493746
min	111.000000	2015.000000	1.000000	0.000000	-1.000000	-71.178674
25%	1001.000000	2016.000000	4.000000	9.000000	42.297438	-71.097135
50%	2907.000000	2017.000000	7.000000	14.000000	42.325538	-71.077524
75%	3201.000000	2017.000000	9.000000	18.000000	42.348624	-71.062467
max	3831.000000	2018.000000	12.000000	23.000000	42.395042	-1.000000

In [20]:
```
#summary about non-numeric data

df.describe(include = 'object')
```

Out[20]:

	INCIDENT_NUMBER	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	DAY_OF_WEEK	UCR_P
count	319050	319050	319050	317285	319050	1019	319050	318
unique	282517	67	244	12	879	1	7	
top	I162030584	Motor Vehicle Accident Response	SICK/INJURED/MEDICAL - PERSON	B2		Y	Friday	Part Tl
freq	13	37132	18783	49940	20250	1019	48489	158

In [21]:
```
#checking Nulls values in the dataframe

df.isnull().sum()
```

Out[21]:
```
INCIDENT_NUMBER             0
OFFENSE_CODE                0
OFFENSE_CODE_GROUP          0
OFFENSE_DESCRIPTION         0
DISTRICT                 1765
REPORTING_AREA              0
SHOOTING               318031
OCCURRED_ON_DATE            0
YEAR                        0
MONTH                       0
DAY_OF_WEEK                 0
HOUR                        0
UCR_PART                   90
STREET                  10871
Lat                     19998
Long                    19998
Location                    0
dtype: int64
```

In [22]:
```
#Checking null values in each column

df.nunique()
```

Out[22]:
```
INCIDENT_NUMBER        282517
OFFENSE_CODE              222
OFFENSE_CODE_GROUP         67
OFFENSE_DESCRIPTION       244
DISTRICT                   12
REPORTING_AREA            879
SHOOTING                    1
OCCURRED_ON_DATE       233229
YEAR                        4
MONTH                      12
DAY_OF_WEEK                 7
HOUR                       24
UCR_PART                    4
STREET                   4657
Lat                     18178
Long                    18178
Location                18194
dtype: int64
```

Lets answer some questions

In [23]:
```
# What are the 20 most common crimes in terms of offense group?

df.OFFENSE_CODE_GROUP.value_counts().sort_values(ascending=False).head(20)
```

Out[23]:
```
Motor Vehicle Accident Response    37132
Larceny                            25935
Medical Assistance                 23540
Investigate Person                 18749
Other                              18073
Drug Violation                     16545
Simple Assault                     15826
Vandalism                          15414
Verbal Disputes                    13099
Towed                              11287
Investigate Property               11124
Larceny From Motor Vehicle         10847
Property Lost                       9751
Warrant Arrests                     8392
Aggravated Assault                  7807
Violations                          6095
Fraud                               5829
Residential Burglary                5606
Missing Person Located              4958
Auto Theft                          4850
Name: OFFENSE_CODE_GROUP, dtype: int64
```

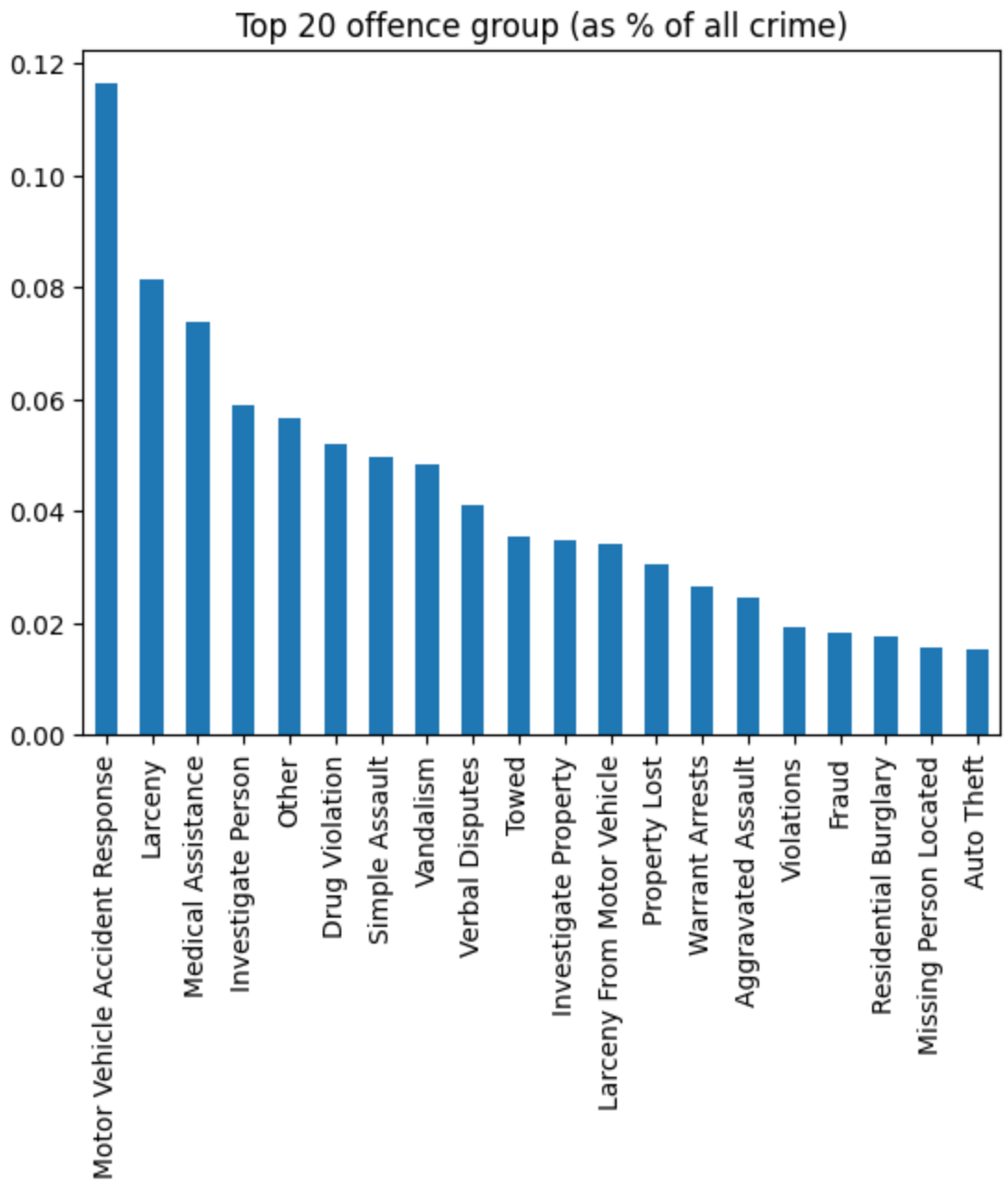
In [24]:
```
# What are the 20 least common crimes in terms of offense group?

df.OFFENSE_CODE_GROUP.value_counts().sort_values(ascending=True).head(20)
```

Out[24]:
```
Burglary - No Property Taken                    2
HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE       2
Biological Threat                               2
INVESTIGATE PERSON                              4
HUMAN TRAFFICKING                               7
Gambling                                        8
Manslaughter                                    8
Explosives                                     27
Phone Call Complaints                          31
Aircraft                                       36
Bomb Hoax                                      75
HOME INVASION                                  77
Arson                                          94
Criminal Harassment                           131
Homicide                                      161
Prostitution                                  207
Harbor Related Incidents                      212
Prisoner Related Incidents                    253
Service                                       285
Embezzlement                                  296
Name: OFFENSE_CODE_GROUP, dtype: int64
```

In [25]:
```
offence_group_vals = df.OFFENSE_CODE_GROUP.value_counts()[:20]

display(offence_group_vals / df.shape[0])

#Creating bar chart

(offence_group_vals / df.shape[0]).plot(kind = 'bar')
plt.title("Top 20 offence group (as % of all crime)")
```

```
Motor Vehicle Accident Response    0.116383
Larceny                            0.081288
Medical Assistance                 0.073782
Investigate Person                 0.058765
Other                              0.056646
Drug Violation                     0.051857
Simple Assault                     0.049604
Vandalism                          0.048312
Verbal Disputes                    0.041056
Towed                              0.035377
Investigate Property               0.034866
Larceny From Motor Vehicle         0.033998
Property Lost                      0.030563
Warrant Arrests                    0.026303
Aggravated Assault                 0.024470
Violations                         0.019104
Fraud                              0.018270
Residential Burglary               0.017571
Missing Person Located             0.015540
Auto Theft                         0.015201
Name: OFFENSE_CODE_GROUP, dtype: float64
```

Out[25]: Text(0.5, 1.0, 'Top 20 offence group (as % of all crime)')



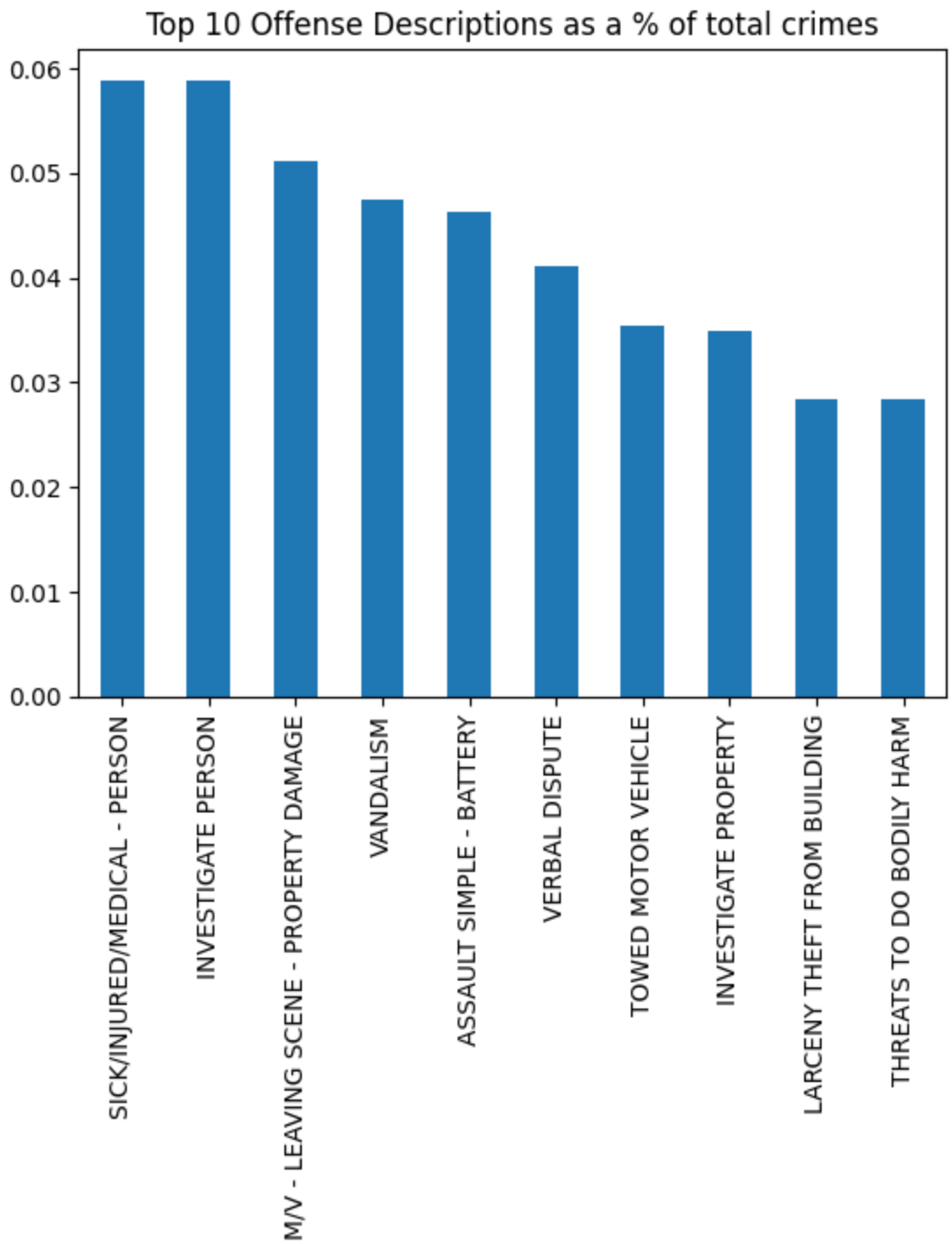
In [26]:
```
# Question 1 - What are the most common offense descriptions?

df.OFFENSE_DESCRIPTION.value_counts().head(10)
```

Out[26]:
```
SICK/INJURED/MEDICAL - PERSON            18783
INVESTIGATE PERSON                       18753
M/V - LEAVING SCENE - PROPERTY DAMAGE    16323
VANDALISM                                15153
ASSAULT SIMPLE - BATTERY                 14791
VERBAL DISPUTE                           13099
TOWED MOTOR VEHICLE                      11287
INVESTIGATE PROPERTY                     11124
LARCENY THEFT FROM BUILDING               9069
THREATS TO DO BODILY HARM                 9042
Name: OFFENSE_DESCRIPTION, dtype: int64
```

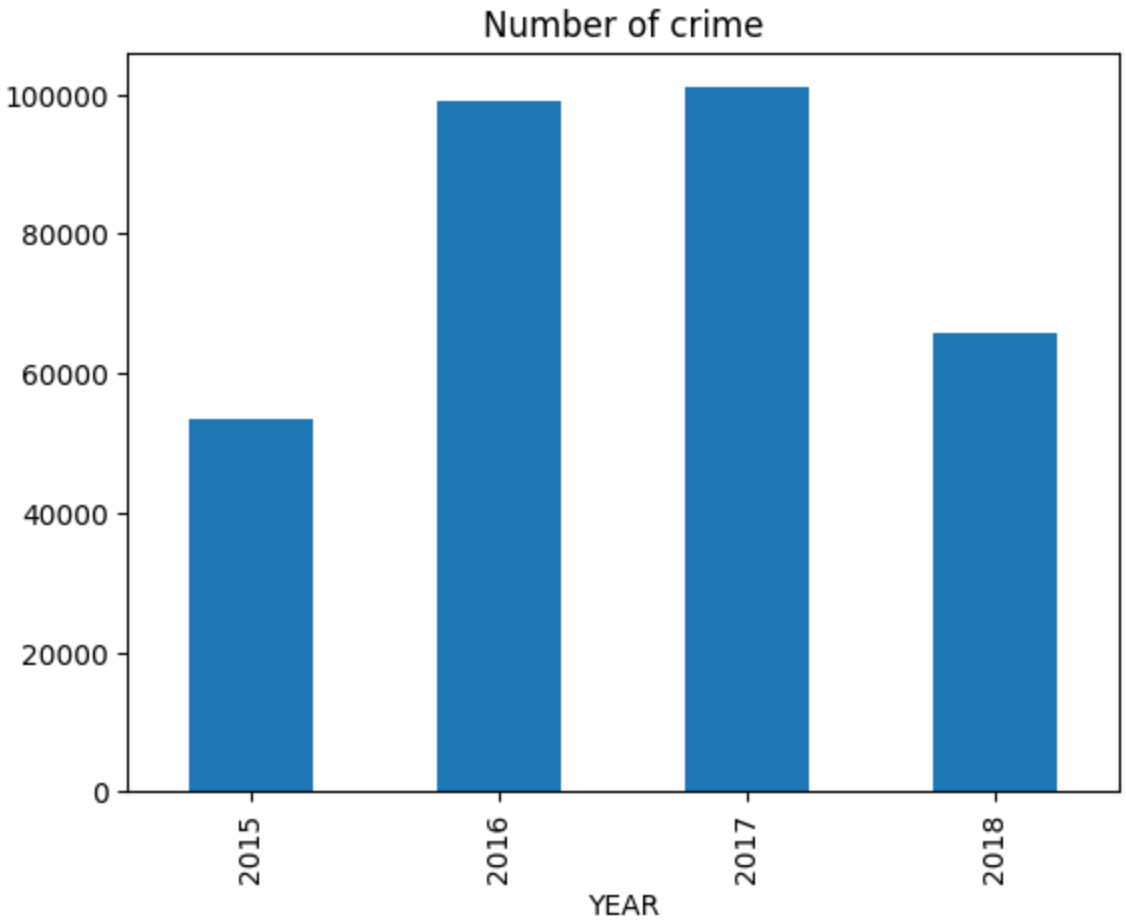
In [27]:
```
# Question 2 - Now try and create a bar chart of
#the Top 10 Offense Descriptions as a % of total crimes

offence_desc_val = df.OFFENSE_DESCRIPTION.value_counts()[:10]

display(offence_group_vals / df.shape[0])

#creating var chart

(offence_desc_val / df.shape[0]).plot(kind = 'bar')

plt.title('Top 10 Offense Descriptions as a % of total crimes')
```

```
Motor Vehicle Accident Response    0.116383
Larceny                            0.081288
Medical Assistance                 0.073782
Investigate Person                 0.058765
Other                              0.056646
Drug Violation                     0.051857
Simple Assault                     0.049604
Vandalism                          0.048312
Verbal Disputes                    0.041056
Towed                              0.035377
Investigate Property               0.034866
Larceny From Motor Vehicle         0.033998
Property Lost                      0.030563
Warrant Arrests                    0.026303
Aggravated Assault                 0.024470
Violations                         0.019104
Fraud                              0.018270
Residential Burglary               0.017571
Missing Person Located             0.015540
Auto Theft                         0.015201
Name: OFFENSE_CODE_GROUP, dtype: float64
```

Out[27]: Text(0.5, 1.0, 'Top 10 Offense Descriptions as a % of total crimes')


Top 10 Offense Descriptions as a % of total crimes
0.06
0.05
0.04
0.03
0.02
0.01
0.00
SICK/INJURED/MEDICAL - PERSON
INVESTIGATE PERSON
M/V - LEAVING SCENE - PROPERTY DAMAGE
VANDALISM
ASSAULT SIMPLE - BATTERY
VERBAL DISPUTE
TOWED MOTOR VEHICLE
INVESTIGATE PROPERTY
LARCENY THEFT FROM BUILDING
THREATS TO DO BODILY HARM

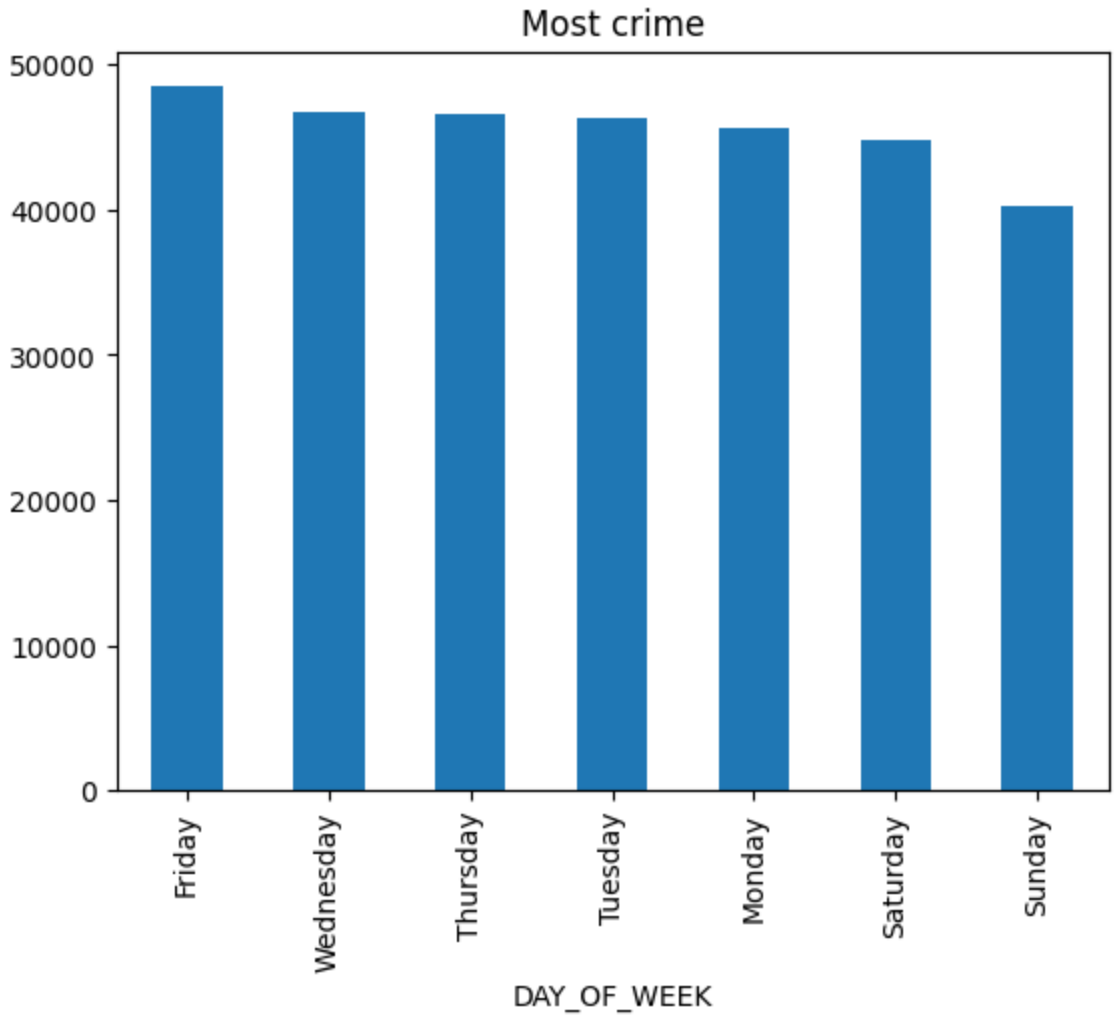
In [28]:
```
# In which year were the most crimes committed?
df.groupby('YEAR').count()['INCIDENT_NUMBER'].plot(kind = 'bar');

plt.title("Number of crime");
```



In [29]:
```
# Question 3 - Are there more crimes committed on specific days?

display(df.groupby('DAY_OF_WEEK').count()['INCIDENT_NUMBER'].sort_values(ascending=False))


df.groupby('DAY_OF_WEEK').count()['INCIDENT_NUMBER'].sort_values(ascending=False).plot(kind='bar');
plt.title('Most crime')
```

```
DAY_OF_WEEK
Friday       48489
Wednesday    46727
Thursday     46655
Tuesday      46376
Monday       45674
Saturday     44816
Sunday       40313
Name: INCIDENT_NUMBER, dtype: int64
```

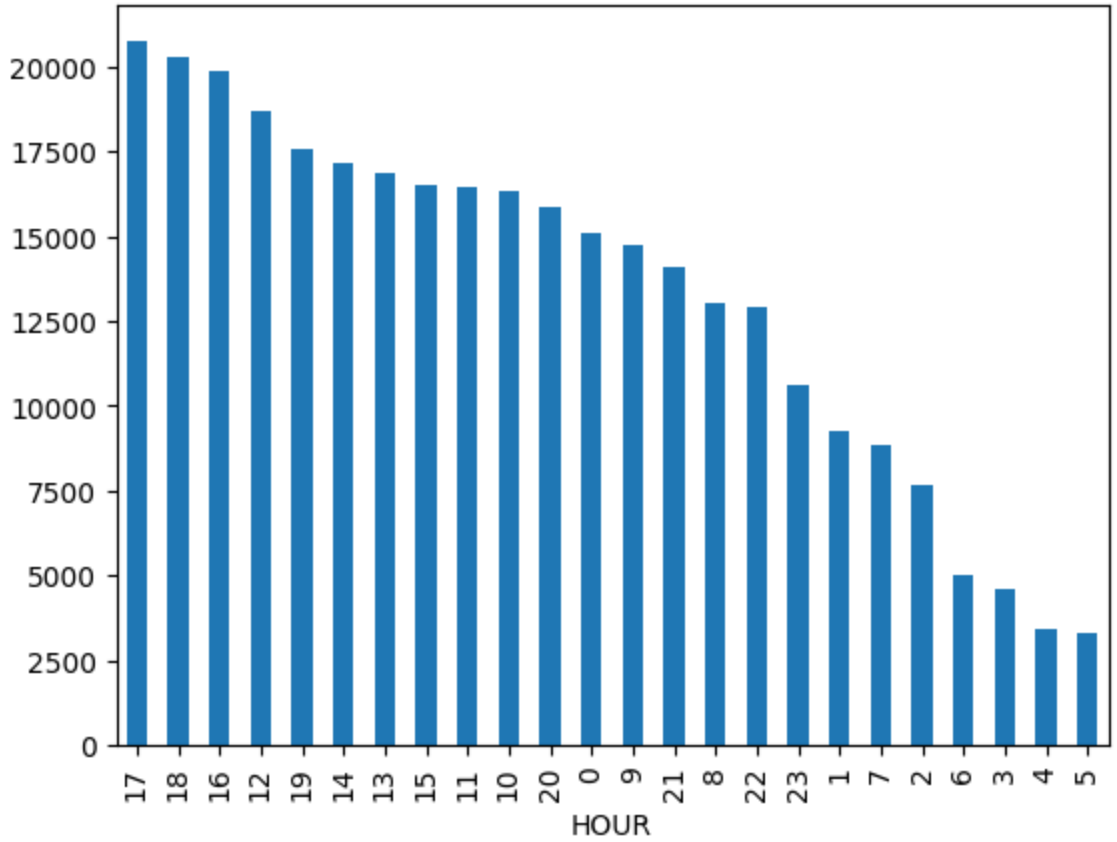
Out[29]: Text(0.5, 1.0, 'Most crime')



In [30]:
```
# Are there more crimes during specific hours?

display(df.groupby('HOUR').count()['INCIDENT_NUMBER'].sort_values(ascending=False).head(5)); #only top5


df.groupby('HOUR').count()['INCIDENT_NUMBER'].sort_values(ascending=False).plot(kind = 'bar')
```

```
HOUR
17    20762
18    20301
16    19870
12    18676
19    17587
Name: INCIDENT_NUMBER, dtype: int64
```

Out[30]: <Axes: xlabel='HOUR'>

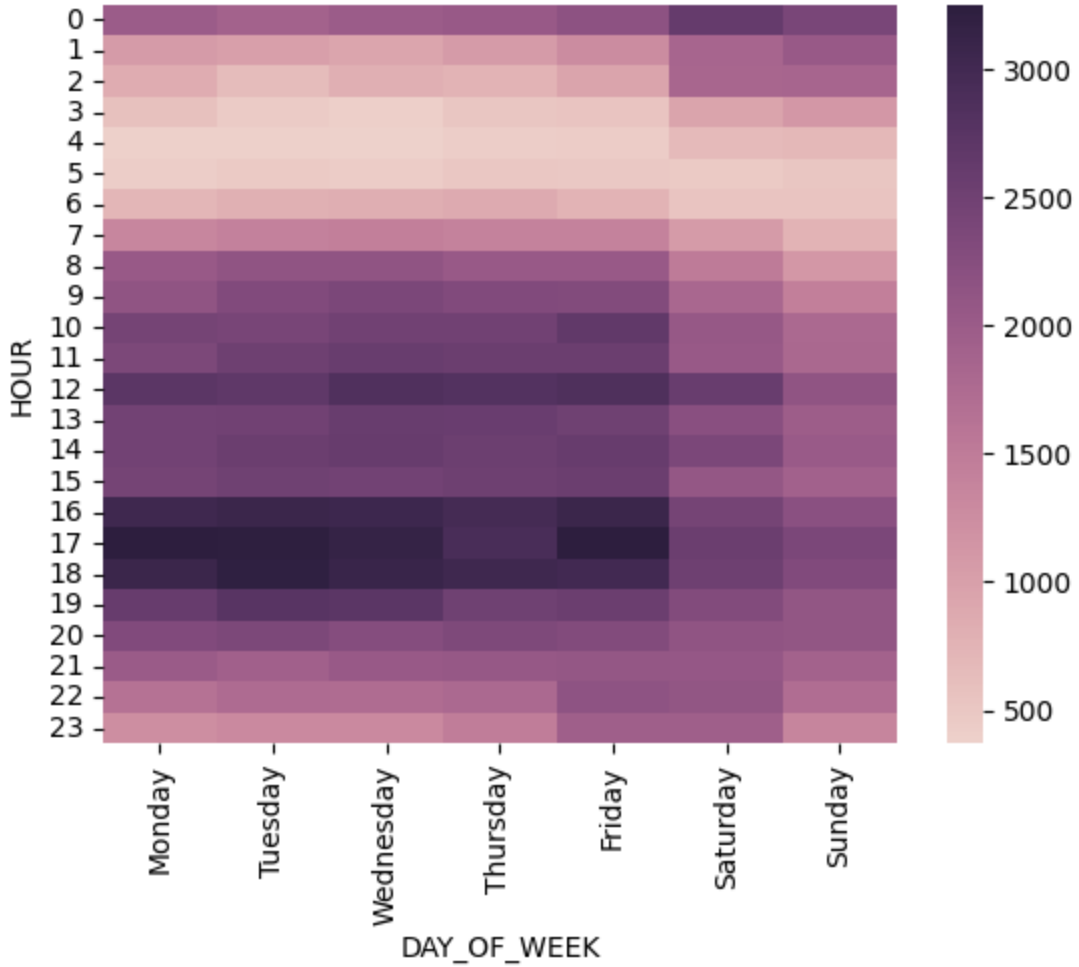


In [31]:
```
# On what days and during which hours are the most crimes committed?
```

In [32]:
```
week_and_hour = df.groupby(['HOUR', 'DAY_OF_WEEK']).count()['INCIDENT_NUMBER'].unstack()
```

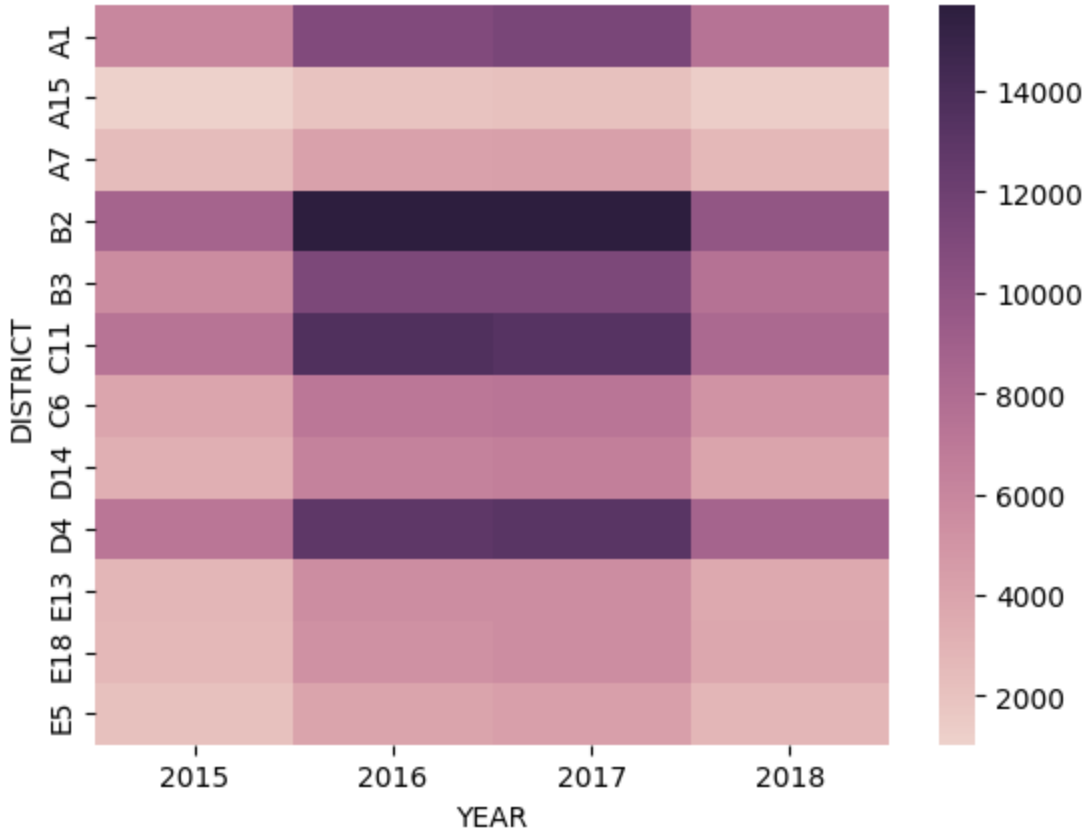
In [33]:
```
week_and_hour = week_and_hour[['Monday', 'Tuesday','Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']]
```

In [34]:
```
sns.heatmap(week_and_hour, cmap=sns.cubehelix_palette(as_cmap=True));
```



In [35]:
```
# Question 4 - In which districts were the most crimes commmitted on yearly basis?

district_and_year = df.groupby(['DISTRICT', 'YEAR']).count()['INCIDENT_NUMBER'].unstack()
```

```
sns.heatmap(district_and_year, cmap=sns.cubehelix_palette(as_cmap=True));
```



Conclusion:

In this data analysis project, we set out to explore a dataset on crime incidents. Here's what we did and what we found:

1. Getting Started: We began by preparing our tools and loading the data. We made sure the data was clean and ready for analysis.

2. Understanding Time: We looked at when the crimes occurred, breaking it down into years, months, weeks, days, hours, and minutes.

3. Summarizing the Data: We got an overall picture of the data, understanding what kinds of crimes were most common and some important statistics about them.

4. Answering Questions: We addressed key questions like the most common crimes, the busiest year for crime, and whether certain days or hours saw more criminal activity.

5. District Insights: We also explored how crime varied across different districts and years, helping identify patterns.

In a nutshell, our analysis provides valuable information about the dataset, which can be useful for law enforcement, policymakers, and researchers. To dive even deeper and predict future crime trends, more advanced analysis and machine learning techniques could be applied.