Import libraries and read dataset file

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from encodings.aliases import aliases # Python has a file containing a dictionary of encoding names and assc

# matplotlib plots will appear directly below the cell in which the plot function was called.
%matplotlib inline
```

In [2]:
```
# Find encodings that work as the default encoding doesn't work

# Below line creates a set of all available encodings
alias_values = set(aliases.values())

for encoding in set(aliases.values()):
    try:
        df = pd.read_csv("crime.csv", nrows = 10, encoding = encoding) # Only read 10 lines for faster read
        print('Successful', encoding)
    except:
        pass
```

```
Successful cp857
Successful kz1048
Successful cp1254
Successful cp273
Successful cp1255
Successful cp860
Successful utf_16_be
Successful cp437
Successful iso8859_4
Successful iso8859_13
Successful iso8859_14
Successful cp949
Successful mac_greek
Successful iso8859_9
Successful mac_roman
Successful koi8_r
Successful mac_latin2
Successful mac_turkish
Successful big5hkscs
Successful cp863
Successful cp775
Successful cp1125
Successful mac_iceland
Successful cp869
Successful iso8859_5
Successful hp_roman8
Successful cp1251
Successful cp1140
Successful mac_cyrillic
Successful cp862
Successful cp1252
Successful iso8859_6
Successful ptcp154
Successful cp855
Successful cp1250
Successful cp1253
Successful mbcs
Successful iso8859_11
Successful cp852
Successful cp1258
Successful cp932
Successful cp1026
Successful iso8859_8
Successful iso8859_10
Successful cp1257
Successful cp864
Successful iso8859_15
Successful latin_1
Successful iso8859_3
Successful cp858
Successful cp850
Successful utf_16_le
Successful iso8859_16
Successful cp865
Successful gb18030
Successful iso8859_2
Successful cp037
Successful cp1256
Successful cp500
Successful gbk
Successful cp861
Successful cp866
Successful iso8859_7
```

In [3]:
```
crime = pd.read_csv("crime.csv", encoding = "iso8859_11")
```

In [4]:
```
crime.head()
```

Out[4]:

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

In [5]:
```
crime.shape
```

Out[5]:
```
(319073, 17)
```

In [6]:
```
crime.duplicated().sum()
```

Out[6]:
```
23
```

In [7]:
```
crime.drop_duplicates(inplace = True)
```

In [8]:
```
crime.shape
```

Out[8]:
```
(319050, 17)
```

Exploring the dataset

In [9]:
```
crime.head()
```

Out[9]:

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

In [10]:
```
crime.tail()
```

Out[10]:

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

In [11]:
```
crime
```

Out[11]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHC
0	I182070945	619	Larceny	LARCENY ALL OTHERS	D14	808	
1	I182070943	1402	Vandalism	VANDALISM	C11	347	
2	I182070941	3410	Towed	TOWED MOTOR VEHICLE	D4	151	
3	I182070940	3114	Investigate Property	INVESTIGATE PROPERTY	D4	272	
4	I182070938	3114	Investigate Property	INVESTIGATE PROPERTY	B3	421	
...	...	...	...	...	...	...	
319068	I050310906-00	3125	Warrant Arrests	WARRANT ARREST	D4	285	
319069	I030217815-08	111	Homicide	MURDER, NON-NEGLIGIENT MANSLAUGHTER	E18	520	
319070	I030217815-08	3125	Warrant Arrests	WARRANT ARREST	E18	520	
319071	I010370257-00	3125	Warrant Arrests	WARRANT ARREST	E13	569	
319072	142052550	3125	Warrant Arrests	WARRANT ARREST	D4	903	

319050 rows × 17 columns

In [12]:
```
crime.info()
```

```
<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 [13]:
```
# Convert data type of "OCCURRED_ON_DATE" column to extract time information easily
crime.OCCURRED_ON_DATE = pd.to_datetime(crime.OCCURRED_ON_DATE)
```

In [14]:
```
crime.info()
```

```
<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 [15]:
```
crime.OCCURRED_ON_DATE.dt.month
```

Out[15]:
```
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 [16]:
```
crime.describe()
```

Out[16]:

	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 [17]:
```
crime.describe(include = 'object') # summary information includes non-numeric columns
```

Out[17]:

	INCIDENT_NUMBER	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	DAY_OF_
count	319050	319050	319050	317285	319050	1019	3
unique	282517	67	244	12	879	1	
top	I162030584	Motor Vehicle Accident Response	SICK/INJURED/MEDICAL - PERSON	B2		Y	
freq	13	37132	18783	49940	20250	1019	

In [18]:
```
crime.columns
```

Out[18]:
```
Index(['INCIDENT_NUMBER', 'OFFENSE_CODE', 'OFFENSE_CODE_GROUP',
       'OFFENSE_DESCRIPTION', 'DISTRICT', 'REPORTING_AREA', 'SHOOTING',
       'OCCURRED_ON_DATE', 'YEAR', 'MONTH', 'DAY_OF_WEEK', 'HOUR', 'UCR_PART',
       'STREET', 'Lat', 'Long', 'Location'],
      dtype='object')
```

In [19]:
```
# Check columns with missing values
crime.columns[np.sum(crime.isnull()) != 0]
```

Out[19]:
```
Index(['DISTRICT', 'SHOOTING', 'UCR_PART', 'STREET', 'Lat', 'Long'], dtype='object')
```

In [20]:
```
# Check for number of unique values in each column
for col in crime.columns:
    unique_count = crime[col].nunique()
    print(col + ": " + str(unique_count) + " unique values")
```

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

In [21]:
```
# What are the most common crimes in terms of offense group?
crime.OFFENSE_CODE_GROUP.value_counts()
```

Out[21]:
```
Motor Vehicle Accident Response             37132
Larceny                                     25935
Medical Assistance                          23540
Investigate Person                          18749
Other                                       18073
                                            ...
HUMAN TRAFFICKING                               7
INVESTIGATE PERSON                              4
Biological Threat                               2
HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE       2
Burglary - No Property Taken                    2
Name: OFFENSE_CODE_GROUP, Length: 67, dtype: int64
```

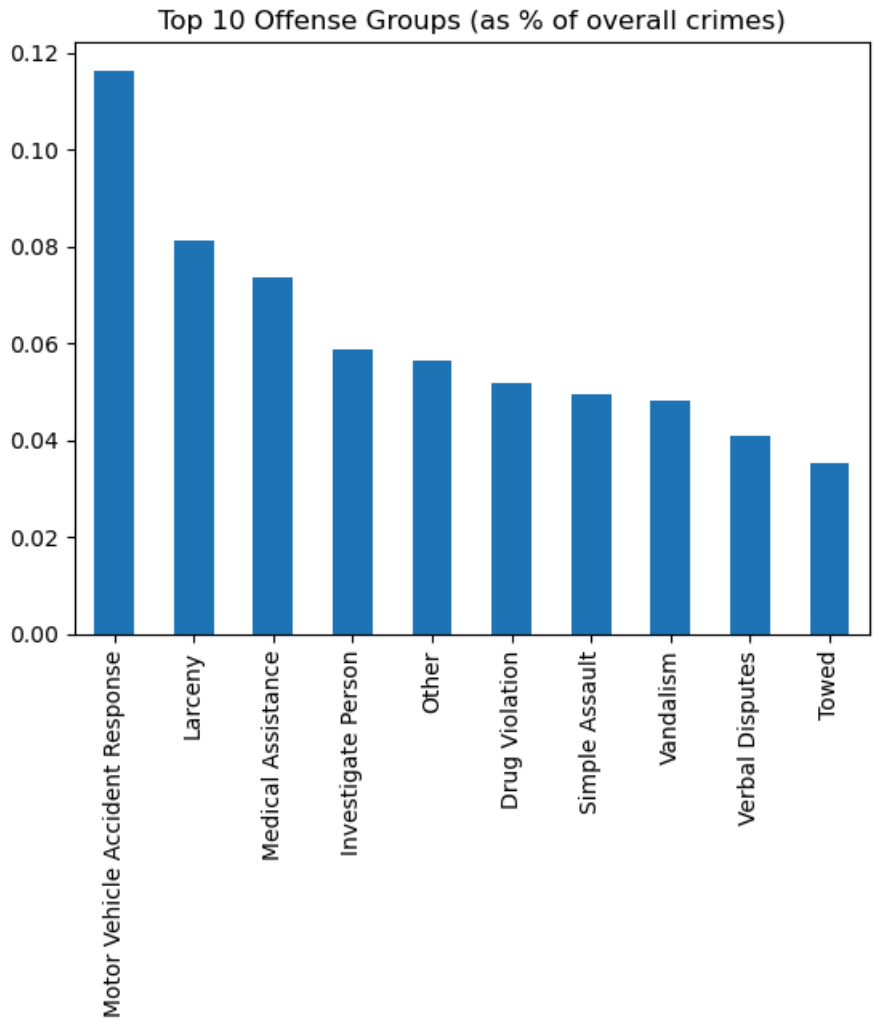
In [22]:
```
offense_group_vals = crime.OFFENSE_CODE_GROUP.value_counts()[:10]

display(offense_group_vals / crime.shape[0])

# Create a bar chart of the top 10 offense groups
(offense_group_vals / crime.shape[0]).plot(kind = 'bar')
plt.title("Top 10 Offense Groups (as % of overall 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
Name: OFFENSE_CODE_GROUP, dtype: float64
```

Out[22]:
```
Text(0.5, 1.0, 'Top 10 Offense Groups (as % of overall crimes)')
```

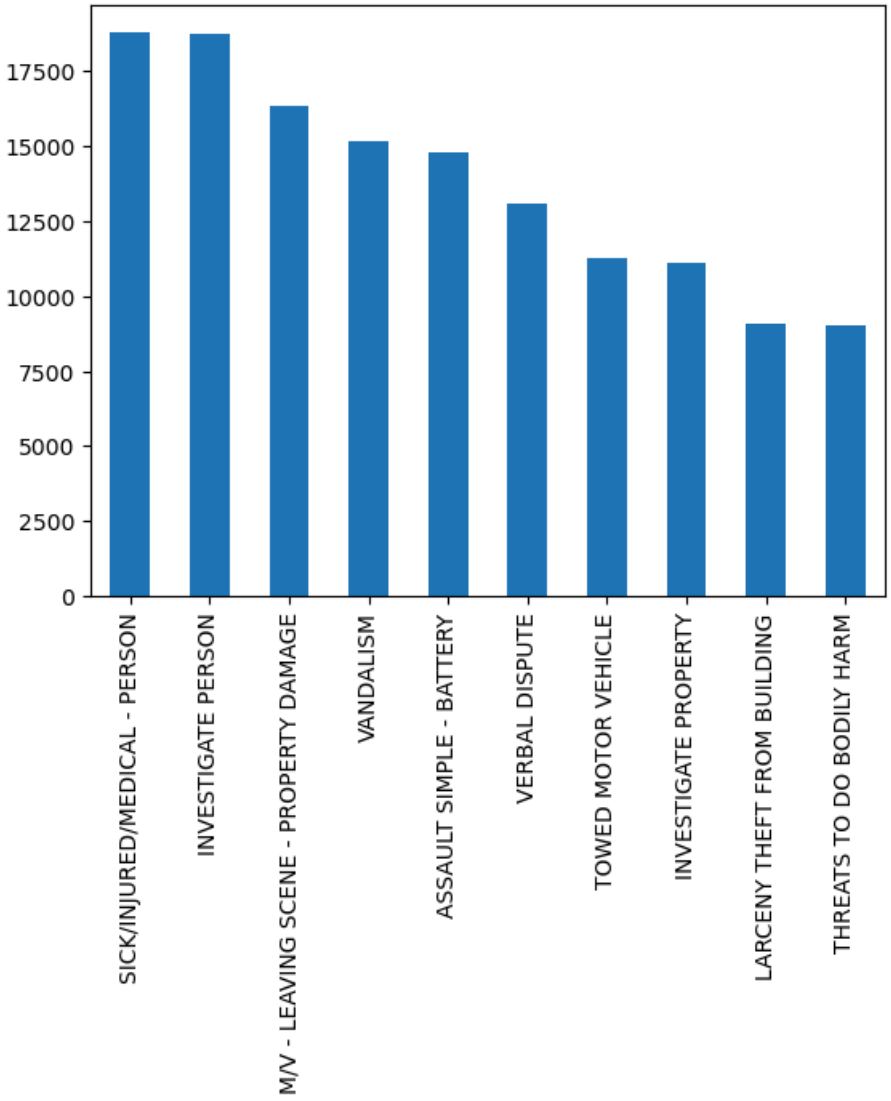


In [23]:
```
crime.OFFENSE_CODE_GROUP.value_counts().sort_values(ascending = True)[:10]
```

Out[23]:
```
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
Name: OFFENSE_CODE_GROUP, dtype: int64
```

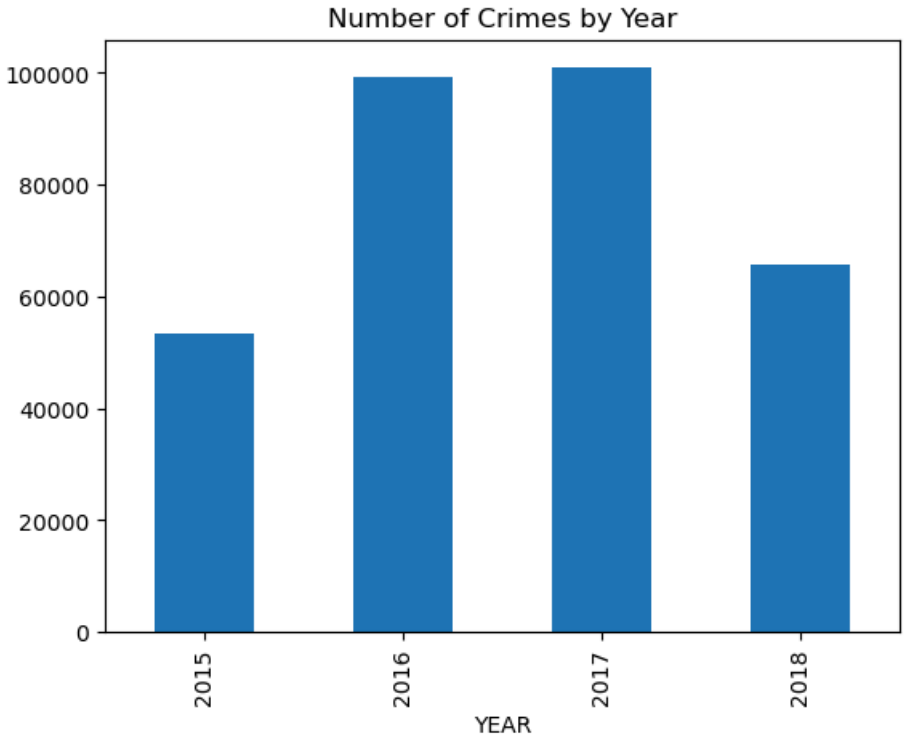
In [25]:
```
crime.OFFENSE_DESCRIPTION.value_counts()[:10].plot(kind = 'bar')
```

Out[25]:
```
<Axes: >
```



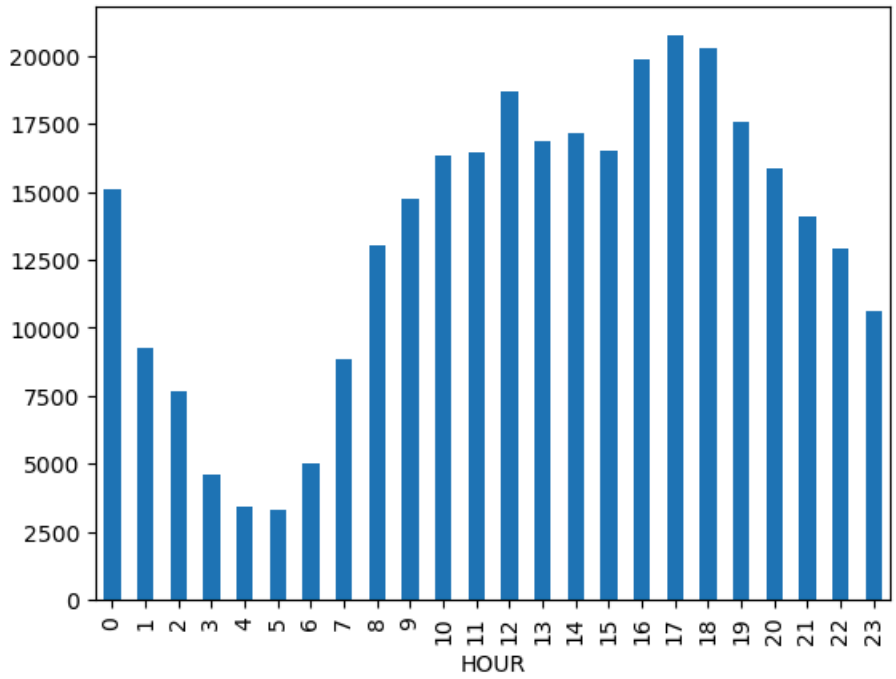
In [28]:
```
# Which year were most crimes committed
crime.groupby('YEAR')['INCIDENT_NUMBER'].count().plot(kind = 'bar')
plt.title("Number of Crimes by Year")
```

Out[28]:
```
Text(0.5, 1.0, 'Number of Crimes by Year')
```



In [30]:
```
crime.groupby('HOUR').count()['INCIDENT_NUMBER'].plot(kind = 'bar')
```

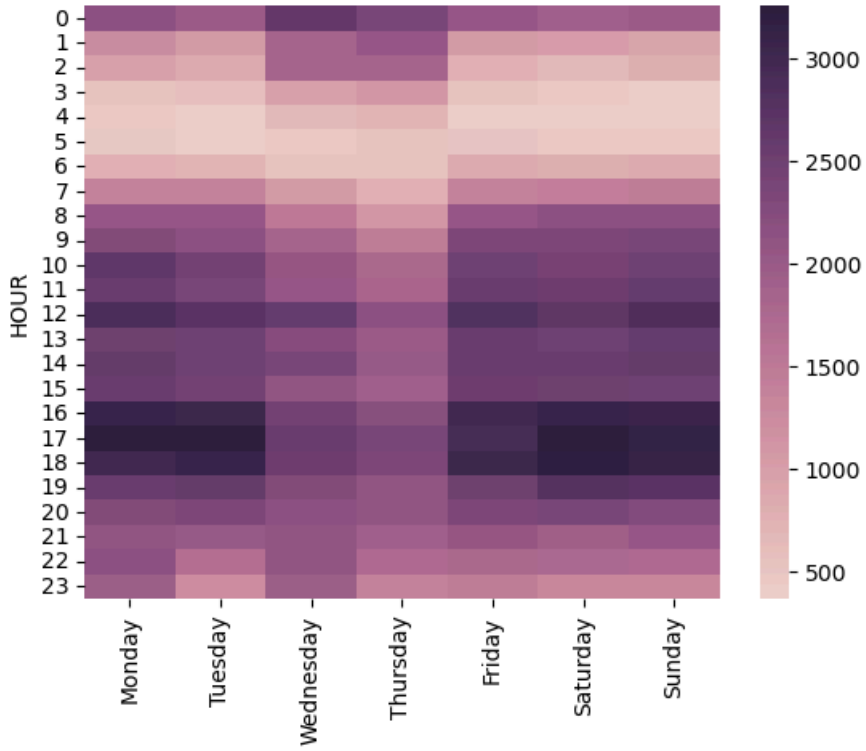
Out[30]:
```
<Axes: xlabel='HOUR'>
```



In [38]:
```
week_and_hour = crime.groupby(['HOUR', 'DAY_OF_WEEK']).count()['INCIDENT_NUMBER'].unstack()
```

In [39]:
```
week_and_hour.columns = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
```

In [40]:
```
sns.heatmap(week_and_hour, cmap = sns.cubehelix_palette(as_cmap = True));
```



In [41]:
```
# Which months were the number of crimes below average?
# Which months on average did most crimes occur?
```

In [55]:
```
# If value is less than average crime per month, highlight the value in blue
avg_crime = crime.groupby(['YEAR','MONTH']).count()['INCIDENT_NUMBER'].mean()
print("The average number of crimes is " + str(avg_crime))

year_and_month = crime.groupby(['MONTH', 'YEAR']).count()['INCIDENT_NUMBER'].unstack()

def style_negative(v, props=''):
```

```
    return props if v < avg_crime else None
s2 = year_and_month.style.applymap(style_negative, props='color:blue;')\
    .applymap(lambda v: 'opacity:20%;' if (v < 0.3) and (v > -0.3) else None)
s2
```

```
The average number of crimes is 7976.25
```

Out[55]:

YEAR	2015	2016	2017	2018
MONTH				
1	nan	7835.000000	7991.000000	7782.000000
2	nan	7307.000000	7408.000000	6937.000000
3	nan	8199.000000	8179.000000	7768.000000
4	nan	8101.000000	8069.000000	7916.000000
5	nan	8578.000000	8715.000000	8906.000000
6	4188.000000	8558.000000	8985.000000	8834.000000
7	8322.000000	8618.000000	9075.000000	8538.000000
8	8340.000000	8938.000000	9206.000000	8337.000000
9	8411.000000	8521.000000	8940.000000	667.000000
10	8305.000000	8582.000000	8846.000000	nan
11	7818.000000	7922.000000	7935.000000	nan
12	7987.000000	7951.000000	7535.000000	nan

In [57]:

```
# Highlight maximum value across each column
def highlight_max(s, props=''):
    return np.where(s == np.nanmax(s.values), props, '')
s2.apply(highlight_max, props = 'color:white;background-color:darkgreen', axis=0)
s2
```

Out[57]:

YEAR	2015	2016	2017	2018
MONTH				
1	nan	7835.000000	7991.000000	7782.000000
2	nan	7307.000000	7408.000000	6937.000000
3	nan	8199.000000	8179.000000	7768.000000
4	nan	8101.000000	8069.000000	7916.000000
5	nan	8578.000000	8715.000000	8906.000000
6	4188.000000	8558.000000	8985.000000	8834.000000
7	8322.000000	8618.000000	9075.000000	8538.000000
8	8340.000000	8938.000000	9206.000000	8337.000000
9	8411.000000	8521.000000	8940.000000	667.000000
10	8305.000000	8582.000000	8846.000000	nan
11	7818.000000	7922.000000	7935.000000	nan
12	7987.000000	7951.000000	7535.000000	nan

In [64]:

```
# Explore the number of crimes between different districts for each year
district_and_year = crime.groupby(['DISTRICT', 'YEAR']).count()['INCIDENT_NUMBER'].unstack()

district_and_year
```

Out[64]:

YEAR	2015	2016	2017	2018
DISTRICT				
A1	6010	10922	11374	7404
A15	1027	1986	2167	1325
A7	2426	4130	4264	2724
B2	8682	15706	15680	9872
B3	5616	11145	11195	7485
C11	7363	13602	13281	8282
C6	3939	7072	7247	5199
D14	3280	6279	6509	4059
D4	7203	12953	13157	8601
E13	2800	5558	5514	3662
E18	2740	5223	5611	3773
E5	2157	4017	4309	2755

In [ ]: