Montreal City Crime Data Analysis and Visualization

In this notebook, I analyzed Montreal city crime data (2015 - 2017) with a goal of providing answers to the following main questions among other findings noted in the analysis:

1. What are the top 3 prevalent crimes or offenses committed in 2015, 2016 and 2017 in Montreal City?
2. Which neighborhoods recorded the highest crime incidents in 2015, 2016 and 2017 and what are the crime types in these neighborhoods?
3. Which neighborhood has the highest cases of murder crime in 2015, 2016 and 2017?
4. What time of the day did most crime incidents occur in 2015, 2016 and 2017?
5. Which top 5 police stations (PDQ) got the most crime complaints in 2015, 2016 and 2017?
6. Which are the top 3 PDQs that got least crime complaints in 2015, 2016 and 2017?

Data source and descriptions

The dataset used in this analysis contains the criminal acts or crimes registered by the Police Department of the City of Montreal (SPVM) and made available on Montreal Open Data Portal (http://donnees.ville.montreal.qc.ca/dataset/actes-criminels (http://donnees.ville.montreal.qc.ca/dataset/actes-criminels)).

Import packages and define settings

In [134]:

```python
# import relevant packages
import pandas as pd
import numpy as np
from dateutil.parser import parse
import calendar

import folium
from folium import plugins
from IPython.display import display_html, HTML
from geopy.geocoders import Nominatim
import json


# Plotly packages
import plotly
from plotly import tools
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot

# settings
init_notebook_mode(connected=True)
pd.set_option('display.max_colwidth', 130)
```

In [147]:

```
#crimedata functions

# define chart marker colors in two lists
markercol = ['rgba(31, 119, 180, 0.5)', 'rgba(255, 127, 14, 0.5)',
              'rgba(50, 171, 96, 0.5)', 'rgba(214, 39, 40, 0.5)',
             'rgba(148, 103, 189, 0.5)', 'rgba(140, 86, 75, 0.5)']
linecol = ['rgba(31, 119, 180, 1.0)', 'rgba(255, 127, 14, 1.0)',
             'rgba(50, 171, 96, 1.0)', 'rgba(214, 39, 40, 1.0)',
             'rgba(148, 103, 189, 1.0)', 'rgba(140, 86, 75, 1.0)']


def map_data(mappings, x):
    '''a function to map columns descriptions in French to English'''
    for i, j in mappings:
        if i == x:
            return j



def embed_map(map):
    '''a function to embed map in notebook '''
    map.save(outfile="map.html")
    return HTML('<iframe src="{i}" style="width: 100%; height: 510px; border: none"></iframe>'



def generate_map(df, yr):
    '''a function for creating an interactive map'''

    # exclude null location values
    ref = df[(df['YEAR'] == yr) & (df['COORDS'] != (1.0, 1.0))].copy()

    # create base map
    crimemap = folium.Map(location=[ref['LAT'].mean(), ref['LON'].mean()], zoom_start=11)

    # create an instance of marker cluster for crimes in the dataset
    crimes = plugins.MarkerCluster().add_to(crimemap)

    # loop through the dataset and add each crime point to the marker cluster
    for lat, lon, category in zip(ref['LAT'], ref['LON'], ref['ADAPTED_CATEGORY']):
        folium.Marker(location=[lat, lon], icon=None, popup=category).add_to(crimes)

    return embed_map(crimemap)

def extract_address(col):
    '''a function to extract addresses from latitudes and longitudes'''
    coord = list(col)
    slist = []

    # Set a custom user_agent for the Nominatim geocoder
    geolocator = Nominatim(user_agent="my-custom-application")

    for i in coord:
        jlist = []
        location = geolocator.reverse(i, timeout=10)

        # Convert location to JSON string and then to a dictionary
        json_string = json.dumps(location.raw)
        dat = json.loads(json_string)

        # Extract neighborhood address
        for j in dat['address'].keys():
            if j not in ['house_number', 'city', 'region', 'state', 'postcode', 'country', 'co
                jlist.append(dat['address'][j])
        locstr = ", ".join(jlist)
        slist.append(locstr)
    return slist
```

```python
def plotchart(chdata, chlayout, titlelist, yearlist, subtitlelist):
    '''a function to plot a chart with 1 row and 3 columns figure'''

    # define subplots
    fig = tools.make_subplots(rows=1, cols=3,
        subplot_titles=(["<b>{}</b>".format(i) for i in subtitlelist]),
        shared_yaxes=True,horizontal_spacing=(0.05),print_grid=False)

    # an empty list to hold chart data definitions for each plot
    trace_list = []
    for i in range(3):
        data = chdata['trace_data'][chdata['trace_data']['YEAR'] == yearlist[i]]
        tracex = go.Bar(x=data[chdata['x']], y=data[chdata['y']], name=titlelist[i], width=0.7
            text=data[chdata['y']], textposition='outside', hoverinfo='text',
            outsidetextfont=dict(size=10), cliponaxis=False,
            marker=dict(color=markercol[:3][i], line=dict(color=linecol[:3][i], width=1)))
        trace_list.append(tracex)

        # define each subplot order of selection
        m = np.array([1, 1, 1, 2, 1, 3]).reshape(3, 2)

        # append each subplot data definitions to the figure instance
        fig.append_trace(trace_list[i], m[i][0], m[i][1])

    # define layout settings
    for i in fig['layout']['annotations']:
        i['font'] = dict(size=12)

    for i in range(1, 4):
        fig['layout']['yaxis' + '{}'.format(i)].update(title=chlayout['yaxistitle'],
            titlefont=dict(size=12, color='rgb(107, 107, 107)'), showticklabels=False, showgri

        fig['layout']['xaxis' + '{}'.format(i)].update(titlefont=dict(size=11, color='rgb(107,
            tickfont=dict(size=11, color='rgb(107, 107, 107)'), tickangle=chlayout['tickangle'

    # update layout settings
    fig['layout'].update(height=chlayout['height'], width=chlayout['width'], showlegend=False,
        autosize=False, title=chlayout['title'], titlefont=dict(size=14),
        paper_bgcolor='rgba(245, 246, 249, 1)', plot_bgcolor='rgba(245, 246, 249, 1)')

    return iplot(fig)
```

Load the data

In [148]:

```python
df = pd.read_csv('actes-criminels.csv')
df.sort_values(by=['DATE'], inplace=True)
df
```

Out[148]:

	CATEGORIE	DATE	QUART	PDQ	X	Y	LONGITUDE	LATITUDE
12763	Introduction	2015-01-01	soir	27.0	293203.472992	5.045436e+06	-73.648516	45.548740
41627	Introduction	2015-01-01	jour	16.0	299612.000006	5.036898e+06	-73.566352	45.471990
56091	Vol dans / sur véhicule à moteur	2015-01-01	jour	15.0	298038.765999	5.034604e+06	-73.586441	45.451332
1782	Vols qualifiés	2015-01-01	soir	16.0	299173.405992	5.035151e+06	-73.571941	45.456266
54900	Vol dans / sur véhicule à moteur	2015-01-01	nuit	20.0	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
225190	Vol dans / sur véhicule à moteur	2022-12-31	jour	10.0	287477.098001	5.043502e+06	-73.721784	45.531224
225191	Vol de véhicule à moteur	2022-12-31	jour	31.0	295686.990996	5.045723e+06	-73.616715	45.551361
225192	Vol dans / sur véhicule à moteur	2022-12-31	jour	21.0	300335.540996	5.040250e+06	-73.557129	45.502162
238211	Vols qualifiés	2022-12-31	nuit	22.0	300687.888991	5.043679e+06	-73.552649	45.533012
229328	Méfait	2022-12-31	jour	48.0	301719.003998	5.048206e+06	-73.539476	45.573752

242483 rows × 8 columns

In [149]:

```python
# display the first five records of the data
df.head()
```

Out[149]:

	CATEGORIE	DATE	QUART	PDQ	X	Y	LONGITUDE	LATITUDE
12763	Introduction	2015-01-01	soir	27.0	293203.472992	5.045436e+06	-73.648516	45.548740
41627	Introduction	2015-01-01	jour	16.0	299612.000006	5.036898e+06	-73.566352	45.471990
56091	Vol dans / sur véhicule à moteur	2015-01-01	jour	15.0	298038.765999	5.034604e+06	-73.586441	45.451332
1782	Vols qualifiés	2015-01-01	soir	16.0	299173.405992	5.035151e+06	-73.571941	45.456266
54900	Vol dans / sur véhicule à moteur	2015-01-01	nuit	20.0	NaN	NaN	NaN	NaN

In [150]:

```python
# determine the number of records in the dataset
print('The dataset contains {0} rows and {1} columns.'.format(df.shape[0], df.shape[1]))
```

```
The dataset contains 242483 rows and 8 columns.
```

In [151]:

```
# check for missing values and data types of the columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 242483 entries, 12763 to 229328
Data columns (total 8 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   CATEGORIE  242483 non-null  object 
 1   DATE       242483 non-null  object 
 2   QUART      242483 non-null  object 
 3   PDQ        242478 non-null  float64
 4   X          201267 non-null  float64
 5   Y          201267 non-null  float64
 6   LONGITUDE  201267 non-null  float64
 7   LATITUDE   201267 non-null  float64
dtypes: float64(5), object(3)
memory usage: 16.6+ MB
```

Crime categories mappings

Columns such as 'CATEGORIE' for crime types as well as 'QUART' field defining the time of the day when the crime event was reported are in French. However, for the purpose of this analysis, these will be summarized in English.

In [152]:

```
# Map crime descriptions and day times in French to English

crime_mappings = list(zip([
        'Introduction', 'Vol dans / sur véhicule à moteur',
        'Vol de véhicule à moteur', 'Méfait', 'Vols qualifiés',
        'Infractions entrainant la mort'
    ], [
        'Burglary', 'Vehicle contents or parts theft', 'Vehicle theft',
        'Misdemeanor', 'Robbery', 'Offenses causing death'
    ]))

day_mappings = list(zip(['jour', 'soir', 'nuit'], ['day', 'evening', 'night']))

# create a new column 'ADAPTED_CATEGORY' for crime descriptions in English
df['ADAPTED_CATEGORY'] = df['CATEGORIE'].apply(
    lambda x: map_data(crime_mappings, x))

# modify 'QUART' column
df['QUART'] = df['QUART'].apply(lambda x: map_data(day_mappings, x))
```

Define new columns and drop non-useful columns

The date field included in the dataset was parsed and separated into 'YEAR' and 'MONTH' columns. The PDQ column is converted to string and the alphabets 'PDQ' are appended to each value. Columns 'X' and 'Y' are not relevant to the analysis and hence dropped.

In [153]:

```
# turn date field from object to date data type
df['DATE'] = df['DATE'].apply(lambda x: parse(x))

# define a new column 'YEAR'
df['YEAR'] = df['DATE'].apply(lambda x: x.year).astype(str)

# define a new column 'MONTH'
df['MONTH'] = df['DATE'].apply(lambda x: x.month)

# modify 'PDQ' column
df['PDQ'] = df['PDQ'].apply(lambda x: 'PDQ '+ str(x))

# drop X and Y columns
df.drop(['X', 'Y'], axis=1, inplace=True)

df.rename(columns = {'LATITUDE':'LAT','LONGITUDE':'LON'}, inplace = True)

df.dropna(inplace = True)
```

Define data scope for the analysis

For the purpose of this analysis, the scope of data will be restricted to period between 2015 and 2017.

In [154]:

```
# extract only '2015 - 2017' records from the dataset
xdf = df[df['YEAR'] != '2018'].copy()
```

Generate crime map

To generate the map, the latitude and longitude fields in the dataset were first chained together and defined as a new column. This was then used to summarize and aggregate crime events in the same neighborhoods to enable easy mapping.

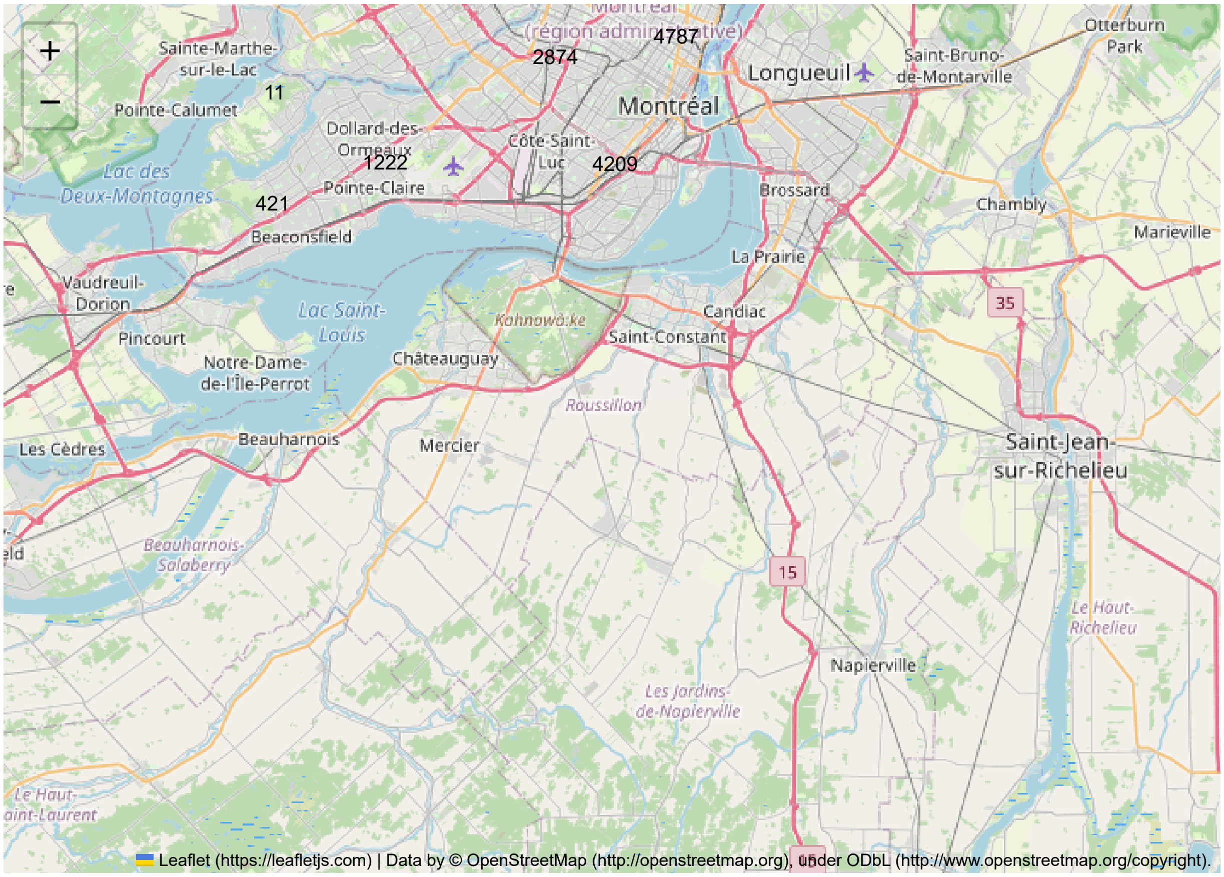
In [155]:

```
# chain latitudes and longitudes together as a new column
xdf['COORDS'] = list(zip(xdf['LAT'], xdf['LON']))

# summarize data
aggcrime = xdf.groupby(['COORDS', 'ADAPTED_CATEGORY', 'YEAR']).agg({'ADAPTED_CATEGORY':'count

# split 'COORDS' into two columns
aggcrime['LAT'] = aggcrime['COORDS'].apply(lambda x: str(x).split(",")[0].replace("(", "")).as
aggcrime['LON'] = aggcrime['COORDS'].apply(lambda x: str(x).split(",")[1].replace(")", "")).as
```

In [156]:

```python
# a crime map of 2015 is shown here. Maps of 2016 and 2017 can be displayed by changing the ye
generate_map(aggcrime, '2015')
```

Out[156]:



Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

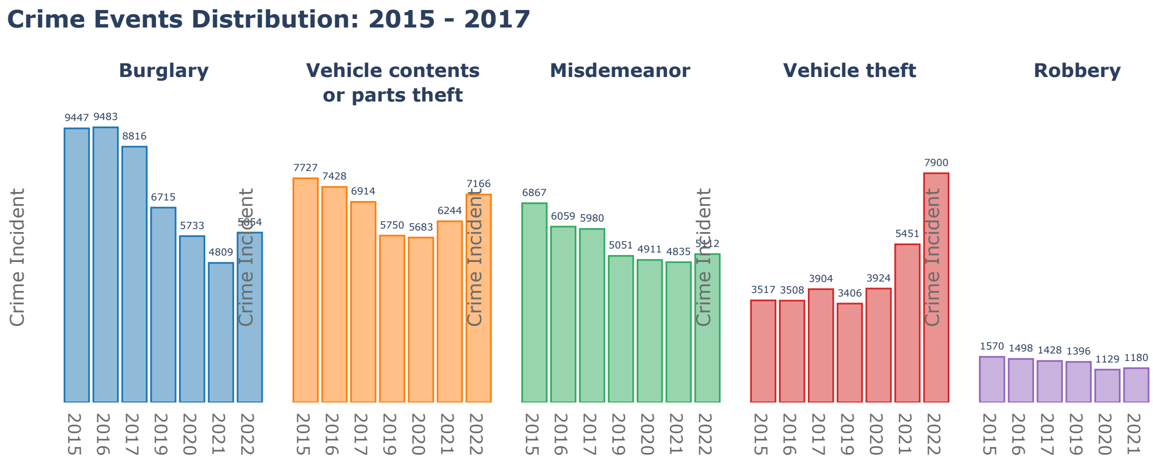
Plot crime categories distribution

In [157]:

```python
# summarize data by year and crime categories
aggdf = xdf.groupby(['YEAR', 'ADAPTED_CATEGORY']).agg({'ADAPTED_CATEGORY':'count'}).rename(col

# define lists to hold crime types and sorted years.
titlelist = [
    'Burglary', 'Vehicle contents or parts theft', 'Misdemeanor',
    'Vehicle theft', 'Robbery', 'Offenses causing death'
]
yearlist = sorted(list(set(xdf['YEAR'])))
```

In [169]:

```
# create an empty list to hold chart data definitions for each plot
trace_list = []

# define subplots
fig = tools.make_subplots(rows=1, cols=6, subplot_titles=(["<b>{}</b>".format(i) for i in titl
    shared_yaxes=True, horizontal_spacing=(0.02), print_grid=False)

# a matrix for subplot selection order
m = np.array([1, 1, 1, 2, 1, 3, 1, 4, 1, 5, 1, 6]).reshape(6, 2)

# define chart data
for i in range(6):
    data = aggdf[aggdf['ADAPTED_CATEGORY'] == titlelist[i]]
    tracex = go.Bar(x=data['YEAR'], y=data['INCIDENT_COUNT'], text=data['INCIDENT_COUNT'],
        textposition='outside', hoverinfo='text', outsidetextfont=dict(size=10),
        cliponaxis=False, name='', showlegend=False, width=0.85,
        marker=dict(color=markercol[i], line=dict(color=linecol[i], width=1)))
    trace_list.append(tracex)

    # append each subplot data definitions to the figure instance
    fig.append_trace(trace_list[i], m[i][0], m[i][1])

# define layout settings
for i in fig['layout']['annotations']:
    i['font'] = dict(size=11)
    i['y'] = 1.2
    i['yanchor'] = 'top'

fig['layout']['annotations'][1]['text'] = '<b>Vehicle contents<br>or parts theft</b>'

for i in range(1, 7):
    fig['layout']['yaxis' + '{}'.format(i)].update(title='Crime Incident',
        titlefont=dict(size=11, color='rgb(107, 107, 107)'), tickfont=dict(size=10, color='rgb
        showticklabels=False, showgrid=True)

    fig['layout']['xaxis' + '{}'.format(i)].update(titlefont=dict(size=11, color='rgb(107, 10
        tickfont=dict(size=10, color='rgb(107, 107, 107)'))

# update layout settings
fig['layout'].update(height=350, width=950, showlegend=False, autosize=False,
    title="<b>Crime Events Distribution: 2015 - 2017</b>", titlefont=dict(size=14),
    paper_bgcolor='rgba(245, 246, 249, 1)', plot_bgcolor='rgba(245, 246, 249, 1)')

iplot(fig)
```



Observation:

- *Besides vehicle theft that increased by 9.4% between 2016 and 2017, all other crime types showed a downward trend in the 3-years period reviewed.*

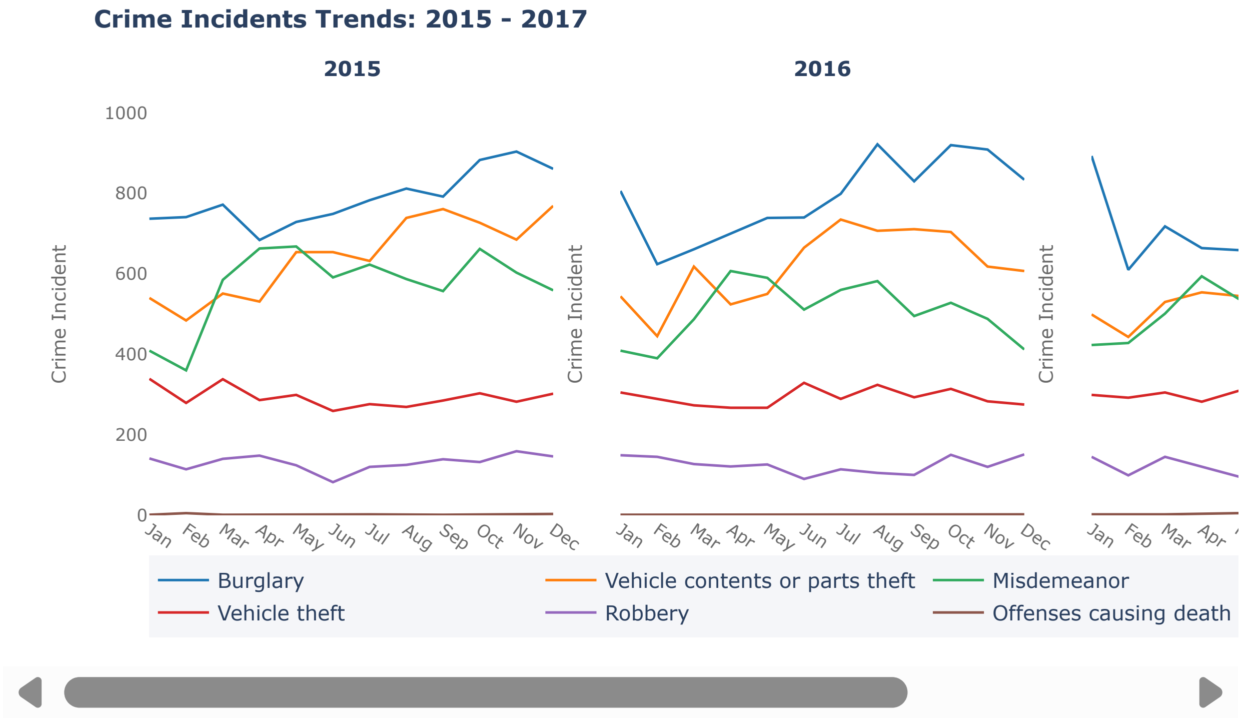
Plot crime incidents trends

In [159]:

```
# summarize data by year, month and crime categories
mon_trend = xdf.groupby(['YEAR', 'MONTH', 'ADAPTED_CATEGORY']).agg({'ADAPTED_CATEGORY': 'count

# sort month column and change months in figures to names
mon_trend['MONTH'] = mon_trend['MONTH'].sort_values().apply(lambda x: calendar.month_abbr[x])
```

In [168]:

```
# define subplots
fig = tools.make_subplots(rows=1, cols=3, subplot_titles=(["<b>{}</b>".format(i) for i in year
    shared_yaxes=True, horizontal_spacing=(0.05), print_grid=False)

# a matrix for subplot selection order
m = np.array([1, 1, 1, 2, 1, 3]).reshape(3, 2)

# define marker colors
scattcol = ['rgb(31, 119, 180)', 'rgb(255, 127, 14)', 'rgb(50, 171, 96)',
            'rgb(214, 39, 40)', 'rgb(148, 103, 189)', 'rgb(140, 86, 75)']

# define chart data
for i in range(3):
    trace_list = []
    data = mon_trend[mon_trend['YEAR'] == yearlist[i]]
    for j in range(6):
        tracex = go.Scatter(x=data[data['ADAPTED_CATEGORY'] == titlelist[j]]['MONTH'],
            y=data[data['ADAPTED_CATEGORY'] == titlelist[j]]['INCIDENT_COUNT'], mode='lines',
            marker=dict(color=scattcol[j]), line=dict(width=1.5), showlegend=False, name=title

        if i == 0:  # show legend for only the first subplot.
            tracex.showlegend = True

        trace_list.append(tracex)

        # append each subplot data definitions to the figure instance
        fig.append_trace(trace_list[j], m[i][0], m[i][1])

# define layout settings
for i in fig['layout']['annotations']:
    i['font'] = dict(size=12)
    i['y'] = 1.07

fig['layout']['legend'] = dict(orientation="h")

for i in range(1, 4):
    fig['layout']['yaxis' + '{}'.format(i)].update(title='Crime Incident',
        titlefont=dict(size=11, color='rgb(107, 107, 107)'), tickfont=dict(size=10, color='rgb
        range=[0, 1000], showgrid=True)

    fig['layout']['xaxis' + '{}'.format(i)].update(titlefont=dict(size=11, color='rgb(107, 107
        tickfont=dict(size=10, color='rgb(107, 107, 107)'), tickangle=35, showgrid=False)

# update layout settings
fig['layout'].update(height=420, width=950, showlegend=True, autosize=False,
    title="<b>Crime Incidents Trends: 2015 - 2017</b>",
    titlefont=dict(size=14), paper_bgcolor='rgba(245, 246, 249, 1)', plot_bgcolor='rgba(245, 2

iplot(fig)
```



Observations:

1. Apart from from few exceptions noted in certain months, almost all the crime categories exhibited the same trend patterns over the 3-years period.

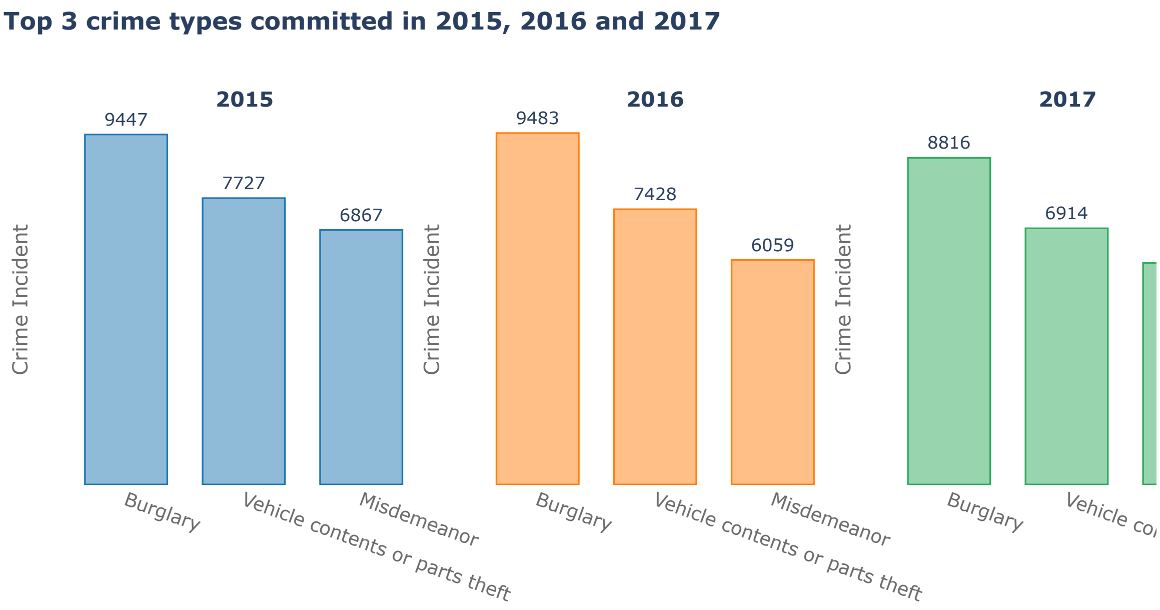
2. Burglary recorded a significant decrease (31%) from 917 in January to 629 in February in 2017.

What are the top 3 prevalent crimes or offenses committed in 2015, 2016 and 2017 in Montreal City?

In [170]:

```
# summarize data
tot =[aggdf[aggdf['YEAR']==year].sort_values('INCIDENT_COUNT', ascending=False)[:3] for year
prev = pd.concat(tot, ignore_index=True)

# display result in crosstab
display(pd.crosstab(index=prev['YEAR'], columns=prev['ADAPTED_CATEGORY'], values=prev['INCIDEN

# define chart data and plot
data = prev
chdata = {'trace_data':data, 'x':'ADAPTED_CATEGORY', 'y':'INCIDENT_COUNT'}
chlayout= {'height':400, 'width':850, 'title':"<b>Top 3 crime types committed in 2015, 2016 a

plotchart(chdata, chlayout, titlelist, yearlist, subtitlelist=yearlist)
```

ADAPTED_CATEGORY	Burglary	Misdemeanor	Vehicle contents or parts theft	Vehicle theft
YEAR				
2015	9447.0	6867.0	7727.0	NaN
2016	9483.0	6059.0	7428.0	NaN
2017	8816.0	5980.0	6914.0	NaN
2019	6715.0	5051.0	5750.0	NaN
2020	5733.0	4911.0	5683.0	NaN
2021	NaN	4835.0	6244.0	5451.0
2022	5854.0	NaN	7166.0	7900.0



Observation:

- *Burglary, Vehicle contents or parts theft and Misdemeanor are the top three prevalent crimes in the 3-years period.*

Which neighborhoods recorded the highest crime incidents in 2015, 2016 and 2017 and what are the crime types in these neighborhoods?

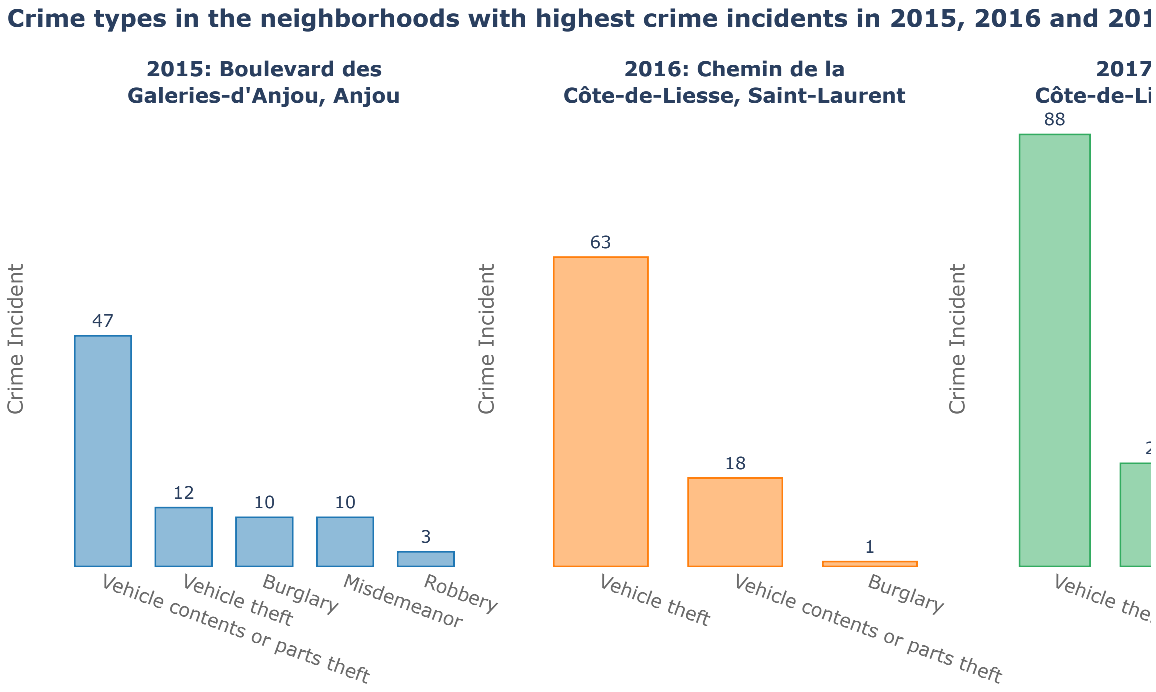
In [162]:

```
# exclude records with no location information
top_neighb = xdf[xdf['LAT']!=1.000000]

# summarize data
top = top_neighb.groupby(['YEAR', 'COORDS']).agg({'ADAPTED_CATEGORY': 'count'}).rename(columns

tot =[top[top['YEAR']==year].sort_values('CRIME_INCIDENT', ascending=False).iloc[:1] for year
topdf = pd.concat(tot, ignore_index=True)

# extract neighborhood addresses as a new column using 'extract_address' function.
topdf['NEIGHBORHOOD'] = extract_address(topdf['COORDS'])

# define list to format subplot headers
klist = ["<b>2015: Boulevard des<br>Galeries-d'Anjou, Anjou</b>",
 '<b>2016: Chemin de la<br>Côte-de-Liesse, Saint-Laurent</b>',
 '<b>2017: Chemin de la<br>Côte-de-Liesse, Saint-Laurent</b>']
```

In [163]:

```
# merge dataframes
tcp = pd.merge(topdf, top_neighb, on=['COORDS','YEAR'])

# summarize data and define a new dataframe
grptcp = tcp.groupby(['YEAR', 'ADAPTED_CATEGORY']).agg({'ADAPTED_CATEGORY': 'count'}).rename(
tot =[grptcp[grptcp['YEAR']==year].sort_values('CRIME_INCIDENT', ascending=False) for year in
prevcrime = pd.concat(tot, ignore_index=True)

# display result
display(topdf[['YEAR', 'CRIME_INCIDENT', 'NEIGHBORHOOD']])

# define chart data and plot
data = prevcrime
chdata = {'trace_data':data, 'x':'ADAPTED_CATEGORY', 'y':'CRIME_INCIDENT'}
chlayout= {'height':450, 'width':950, 'title':"<b>Crime types in the neighborhoods with highes

plotchart(chdata, chlayout, titlelist, yearlist, subtitlelist=klist)
```

	YEAR	CRIME_INCIDENT	NEIGHBORHOOD
0	2015	82	Boulevard des Galeries-d'Anjou, Anjou, Agglomération de Montréal, CA-QC
1	2016	82	Chemin de la Côte-de-Liesse, Saint-Laurent, Agglomération de Montréal, CA-QC
2	2017	118	Chemin de la Côte-de-Liesse, Saint-Laurent, Agglomération de Montréal, CA-QC
3	2019	53	Boulevard des Galeries-d'Anjou, Anjou, Agglomération de Montréal, CA-QC
4	2020	68	Boulevard Robert-Bourassa et rue Sainte-Catherine, Boulevard Robert-Bourassa, Quartier des Spectacles, Ville-Marie, Agglomérat...
5	2021	81	Voie de Service Nord, Pointe-Claire, Agglomération de Montréal, CA-QC
6	2022	142	Voie de Service Nord, Pointe-Claire, Agglomération de Montréal, CA-QC



Observations:

1. Boulevard des Galeries-d'Anjou, Anjou and Chemin de la Côte-de-Liesse, Saint-Laurent are neighborhoods with the highest number of crimes in 2015, 2016 and 2017.

2. Vehicle theft and vehicle contents or parts theft were the main common crime types in Boulevard des Galeries-d'Anjou, Anjou and Chemin de la Côte-de-Liesse, Saint-Laurent neighborhoods within the period reviewed.

3. Vehicle theft was particularly prominent in Chemin de la Côte-de-Liesse, Saint-Laurent in 2016 and in 2017. The neighborhood has about 40% increase in vehicle theft from 2016 to 2017. Vehicle contents or parts theft recorded 22%

Which neighborhood has the highest cases of murder crime in 2015, 2016 and 2017?

In [164]:

```
# extract and summarize data
ddf = top_neighb[top_neighb['ADAPTED_CATEGORY']=="Offenses causing death"]
dtop = ddf.groupby(['YEAR', 'COORDS']).agg({'COORDS':'count'}).rename(columns={'COORDS':'DEATH

# sort and extract neighborhood addresses as a new column using 'extract_address' function.
topdf = dtop.sort_values('DEATH_INCIDENT', ascending=False).reset_index(drop=True).iloc[:1]
topdf['NEIGHBORHOOD'] = extract_address(topdf['COORDS'])

# display result
display(topdf)
```

	YEAR	COORDS	DEATH_INCIDENT	NEIGHBORHOOD
0	2016	(45.511716, -73.562202)	2	Pavillon Sainte-Catherine, Rue Sainte-Catherine Est, Quartier des Spectacles, Ville-Marie, Agglomération de Montréal, CA-QC

Observation:

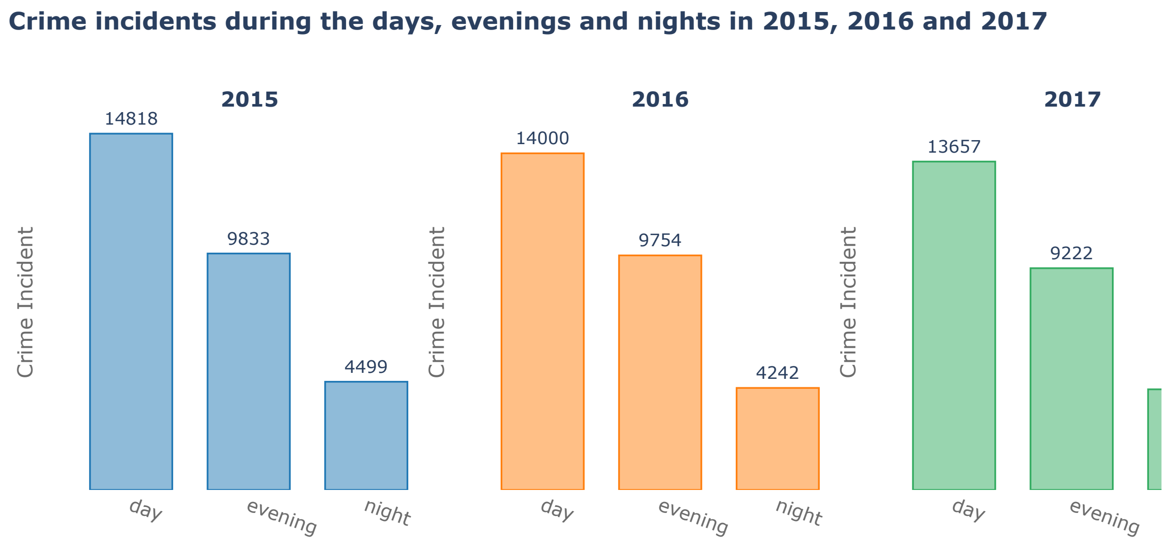
- *Two cases of murder occurred in 2016 at Pavillon Sainte-Catherine, Rue Sainte-Catherine Est, Quartier des Spectacles, Centre-Ville, Ville-Marie. These are the highest murder cases within the 3-years period.*

What time of the day did most crime incidents occur in 2015, 2016 and 2017?

In [165]:

```
# summarize and sort data
crime_time = xdf.groupby(['YEAR', 'QUART']).agg({'QUART': 'count'}).rename(columns={'QUART':'C

tot =[crime_time[crime_time['YEAR']==year].sort_values('COUNT', ascending=False) for year in y
timedf = pd.concat(tot, ignore_index=True)

# display result in crosstab
display(pd.crosstab(index=timedf['YEAR'], columns=timedf['QUART'], values=timedf['COUNT'], agg

# define chart data and plot
data = timedf
chdata = {'trace_data':data, 'x':'QUART', 'y':'COUNT'}
chlayout= {'height':400, 'width':850, 'title':"<b>Crime incidents during the days, evenings ar

plotchart(chdata, chlayout, titlelist, yearlist, subtitlelist=yearlist)
```

QUART	**day**	**evening**	**night**
YEAR			
2015	14818	9833	4499
2016	14000	9754	4242
2017	13657	9222	4184
2019	11311	7510	3517
2020	11141	6713	3548
2021	12045	6807	3693
2022	14417	8081	4994



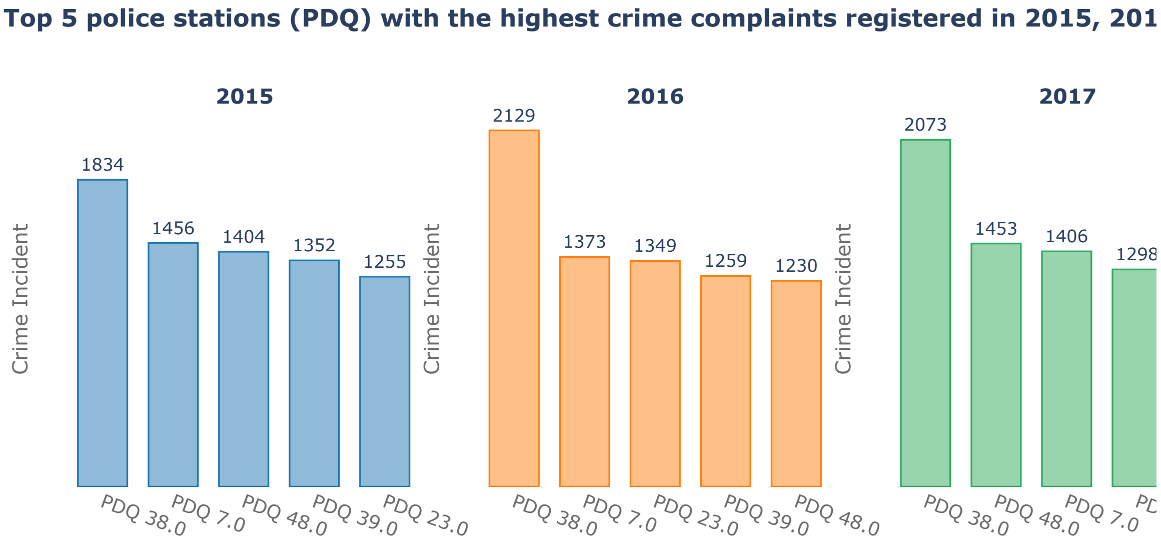
Observation:

- *Crimes committed during the days (jour) in each year were about twice the total crimes registered in the evenings (soir) and in the nights (nuit) for the same year. However daytime crimes recorded 5.4% decrease from 2015 to*

Which top 5 police stations (PDQ) got the most crime complaints in 2015, 2016 and 2017?

In [166]:

```
# summarize and sort data
crime_time = xdf.groupby(['YEAR', 'PDQ']).agg({'PDQ': 'count'}).rename(columns={'PDQ':'COUNT'}

tot =[crime_time[crime_time['YEAR']==year].sort_values('COUNT', ascending=False).iloc[:5] for
pdqdf = pd.concat(tot, ignore_index=True)

# define chart data and plot
data = pdqdf
chdata = {'trace_data':data, 'x':'PDQ', 'y':'COUNT'}
chlayout= {'height':400, 'width':850, 'title':"<b>Top 5 police stations (PDQ) with the highest

plotchart(chdata, chlayout, titlelist, yearlist, subtitlelist=yearlist)
```



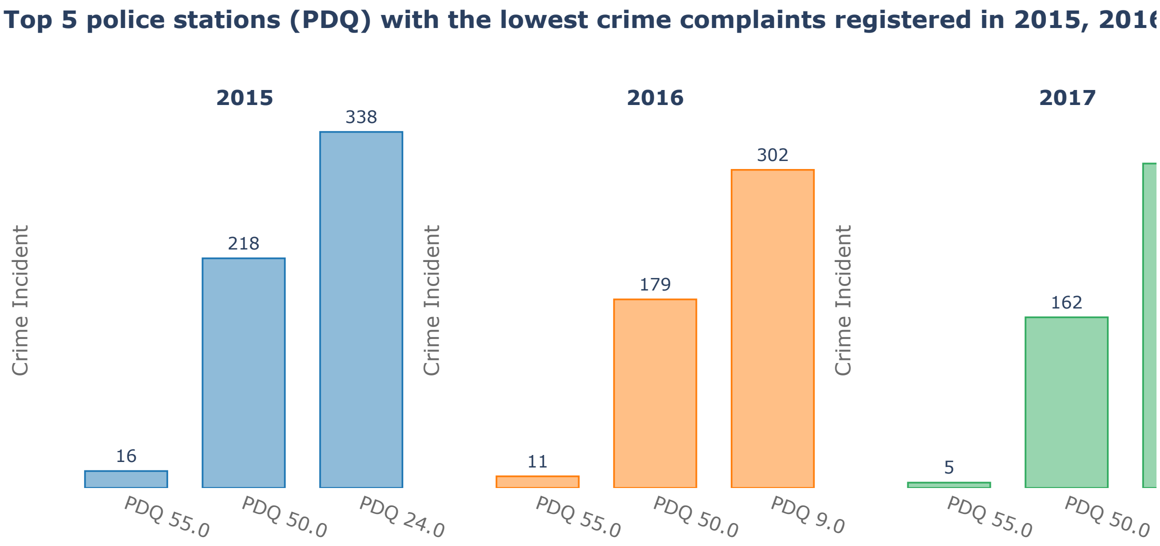
Observation:

- *PDQ38 and PDQ7 are the prominent among the top stations that registered most crime incidents in 2015, 2016 and 2017.*

Which are the top 3 PDQs that got least crime complaints in 2015, 2016 and 2017?

In [167]:

```
# sort and define a new dataframe
tot =[crime_time[crime_time['YEAR']==year].sort_values('COUNT', ascending=True).iloc[:3] for y
pdqdf = pd.concat(tot, ignore_index=True)

# define chart data and plot
data = pdqdf
chdata = {'trace_data':data, 'x':'PDQ', 'y':'COUNT'}
chlayout= {'height':400, 'width':850, 'title':"<b>Top 5 police stations (PDQ) with the lowest

plotchart(chdata, chlayout, titlelist, yearlist, subtitlelist=yearlist)
```



Observation:

- *PDQ0 and PDQ55 recorded the least crime cases among the 3 top stations with low crime registrations in 2015, 2016 and 2017.*

Conclusion

- Burglary, Vehicle contents or parts theft and Misdemeanor are the three most prevalent crimes in the 3-years period.
- Besides vehicle theft that increased by 9.4% between 2016 and 2017, all other crime types showed a downward trend in the 3-years period reviewed.
- Offenses causing death are the least crime incidents noted during the 3-years period.
- Burglary recorded a significant decrease (31%) from 917 in January to 629 in February in 2017.
- Boulevard des Galeries-d'Anjou, Anjou and Chemin de la Côte-de-Liesse, Saint-Laurent are the neighborhoods with the highest number of crimes in 2015, 2016 and 2017. Vehicle theft and vehicle contents or parts theft are the

main common crime types noted in these neighborhoods.

- Vehicle theft was particularly prominent in Chemin de la Côte-de-Liesse, Saint-Laurent in 2016 and in 2017. The neighbourhood has about 40% increase in vehicle theft from 2016 to 2017. Vehicle contents or parts theft recorded 22% increase within the same period.
- Two cases of murder occurred in 2016 at Pavillon Sainte-Catherine, Rue Sainte-Catherine Est, Quartier des Spectacles, Centre-Ville, Ville-Marie. These are the highest murder cases within the 3-years period based on the dataset.
- Crimes committed during the days (jour) in each year were about twice the total crimes registered in the evenings (soir) and in the nights (nuit) for the same year. However daytime crimes recorded 5.4% decrease from 2015 to 2016 and 2.2% decrease between 2016 and 2017.
- PDQ38 and PDQ7 are the prominent among the top stations that registered most crime incidents in 2015, 2016 and 2017.
- PDQ0 and PDQ55 recorded the least crime cases among the 3 top stations with low crime registrations in 2015, 2016 and 2017.