In [2]:
```
from plotly.offline import init_notebook_mode, iplot
from sqlalchemy import create_engine
from IPython.display import display
import chart_studio.plotly as py
import matplotlib.pyplot as plt
import plotly.graph_objs as go
import dask.dataframe as dd
from datetime import datetime
import pandas as pd
import hvplot.dask
import os.path
import pathlib
import pyodbc

init_notebook_mode()
disk_engine = create_engine('sqlite:///311.db')
db_file=pathlib.Path("311.db")
chunksize=100000
required_columns=['Agency Name','Complaint Type','Borough']
file='311_Service_Requests_from_2010_to_Present.csv'
server_name = 'cd767c5d883f'
database_name = 'master'
```

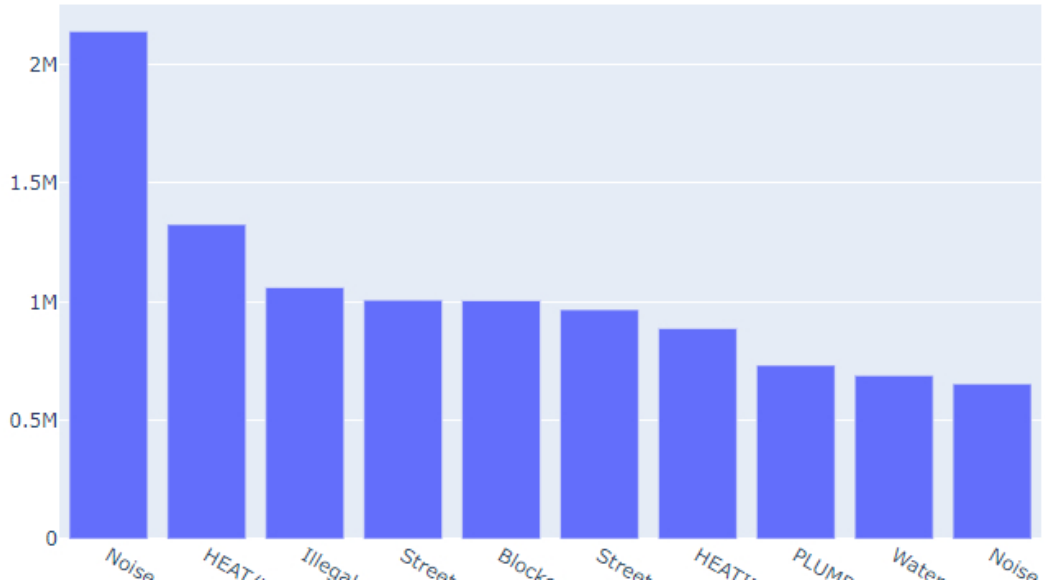
Wykorzystanie biblioteki Pandas

Załadowanie pliku oraz przygotowanie danych.

In [3]:
```
df = pd.read_csv(file,usecols=required_columns, chunksize=chunksize)
chunk_list = []
for data_chunk in df:
    data_chunk = data_chunk.rename(columns={c: c.replace(' ', '') for c in data_chunk.columns})
    filtered_df = data_chunk.dropna()
    chunk_list.append(filtered_df)
chunk_data = pd.concat(chunk_list)
```

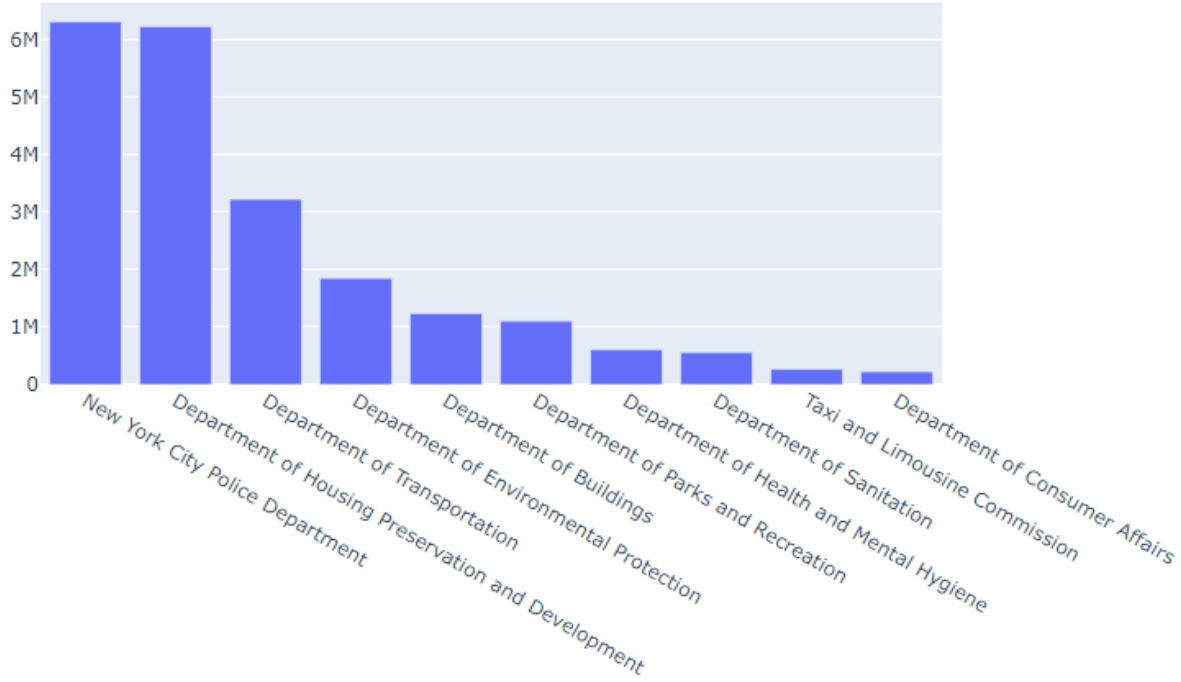
Najczęściej zgłaszane skargi.

In [4]:
```
series = chunk_data['ComplaintType'].value_counts().nlargest(10)
iplot([go.Bar(x=series.index, y=series.values)], filename='Najczesciej zglaszane skargi')
```



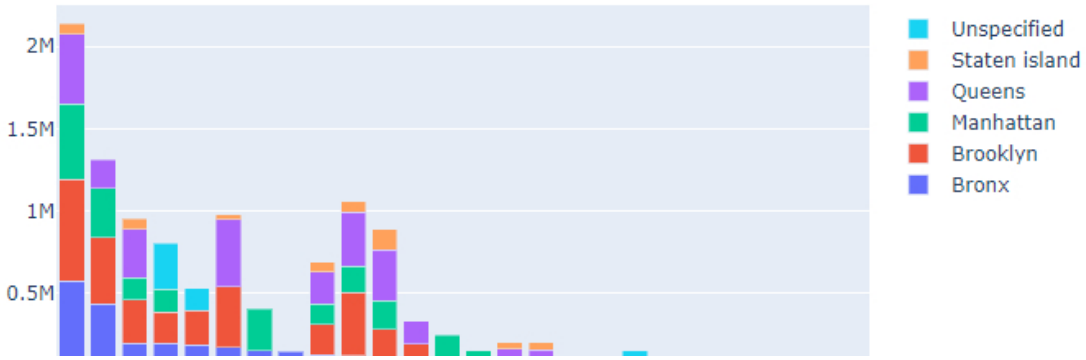
Urzędy, do których najczęściej zgłaszano skargi.

In [5]:
```
series = chunk_data['AgencyName'].value_counts().nlargest(10)
iplot([go.Bar(x=series.index, y=series.values)], filename='Najczestsze urzedy')
```



Najczęściej zgłaszane skargi w każdej dzielnicy.

In [37]:
```
borough_groups = chunk_data.groupby(['Borough', 'ComplaintType']).count().groupby(level=0)

traces = []
for name, value in borough_groups:
    df = value.sort_values(by='AgencyName', ascending=False).head(10).reset_index()
    traces.append(go.Bar(x=df['ComplaintType'], y=df.AgencyName, name=name.capitalize()))

iplot({'data': traces, 'layout': go.Layout(barmode='stack', xaxis={'tickangle': 40}, margin={'b': 150})}, filena
```



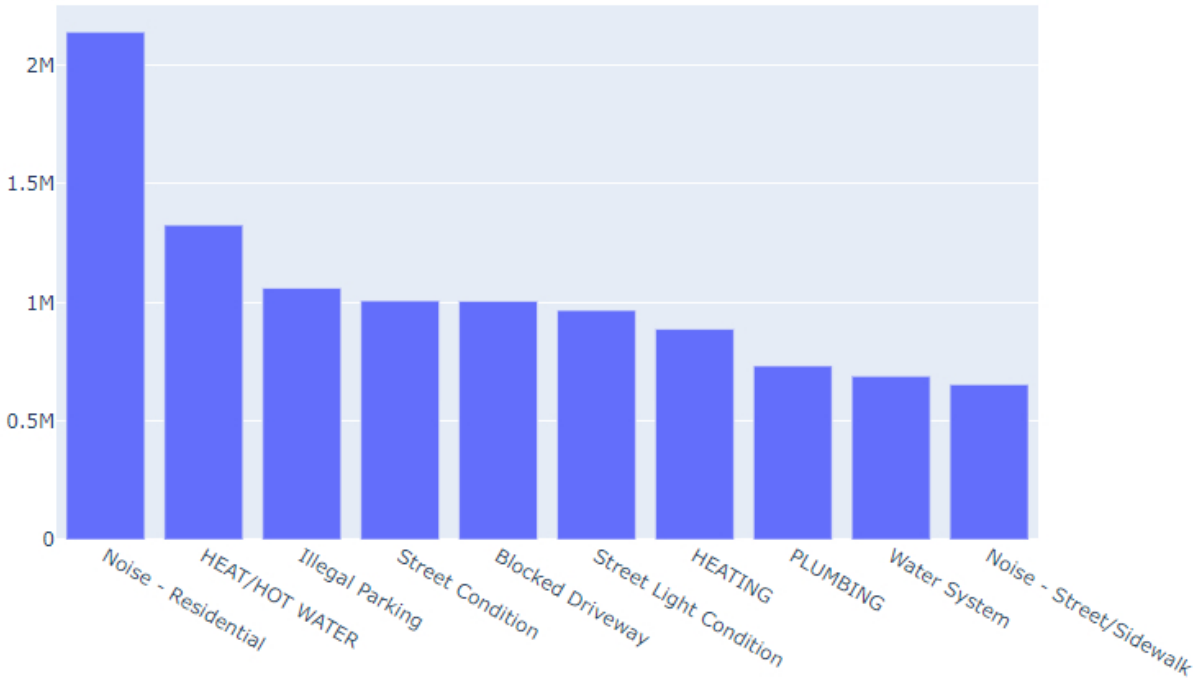
Wykorzystanie implementacji modułów Pandas z biblioteki Dask

Załadowanie pliku oraz przygotowanie danych.

In [8]:
```
dask_df = dd.read_csv(file, usecols=required_columns, dtype='str')
dask_df = dask_df.rename(columns={c: c.replace(' ', '') for c in dask_df.columns})
dask_df = dask_df.dropna()
```

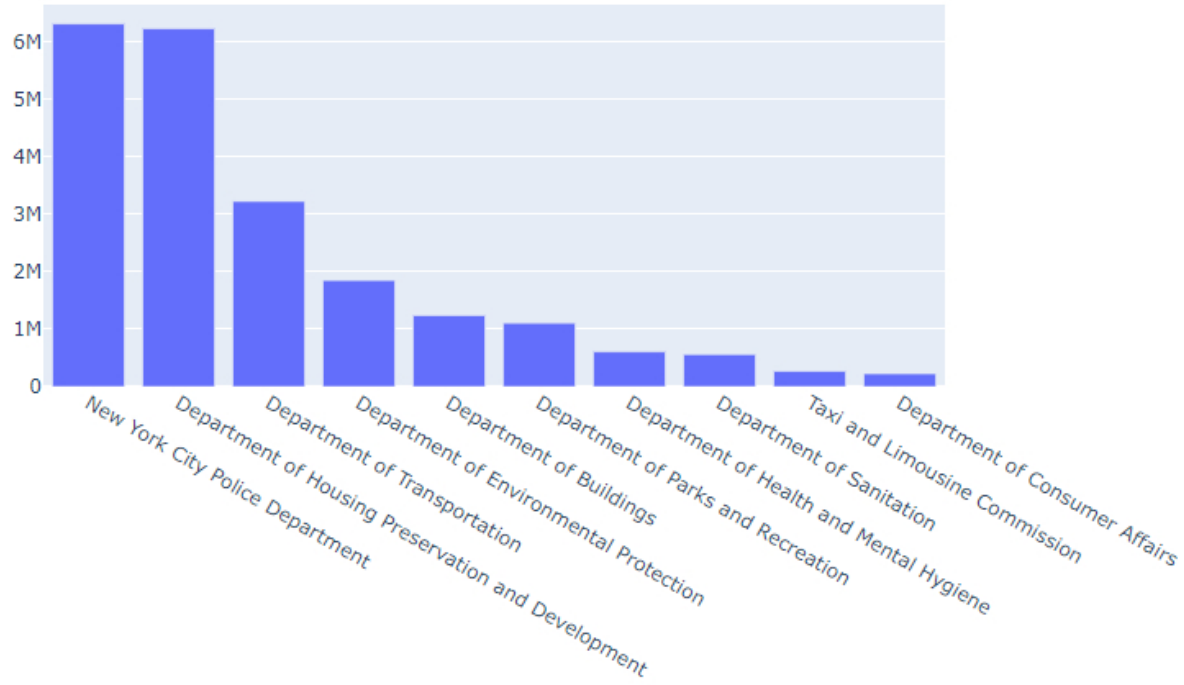
Najczęściej zgłaszane skargi.

In [39]:
```
series = dask_df['ComplaintType'].value_counts().nlargest(10).compute()
iplot([go.Bar(x=series.index, y=series.values)], filename='Najczesciej zglaszane skargi')
```



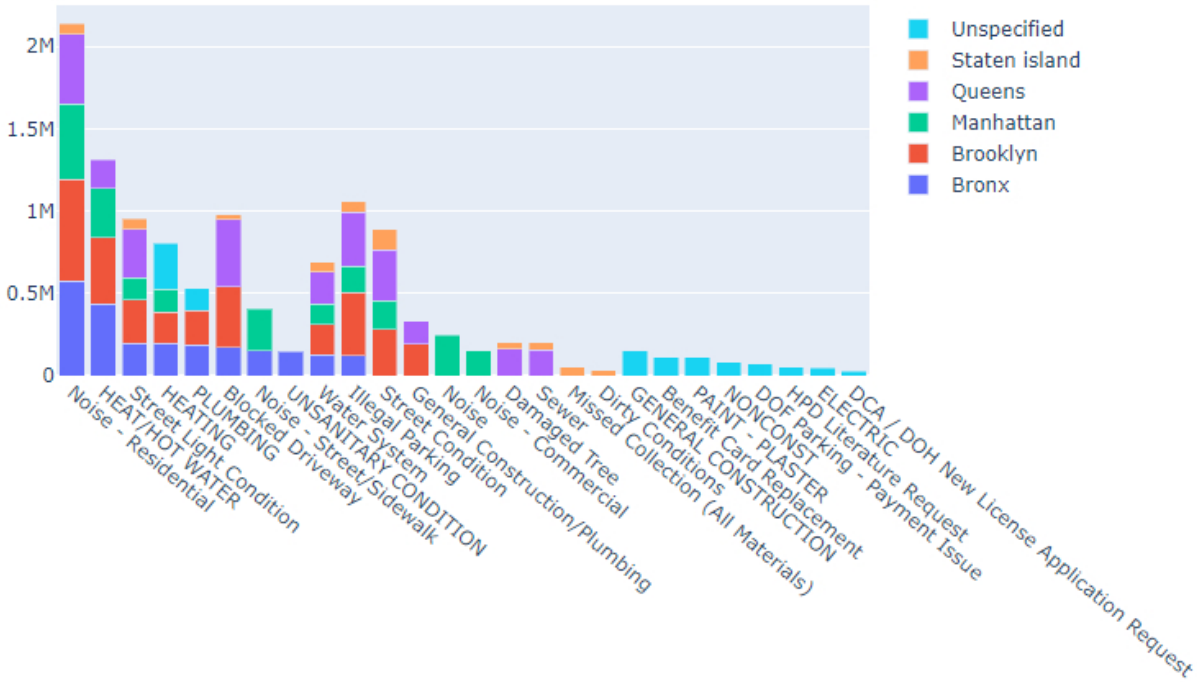
Urzędy, do których najczęściej zgłaszano skargi.

In [40]:
```
series = dask_df['AgencyName'].value_counts().nlargest(10).compute()
iplot([go.Bar(x=series.index, y=series.values)], filename='Najczestsze urzedy')
```



Najczęściej zgłaszane skargi w każdej dzielnicy.

In [38]:
```
dask_groups = dask_df.groupby(['Borough', 'ComplaintType']).count().compute().groupby(level=[0])

traces = []
for name, value in dask_groups:
    df = value.sort_values(by='AgencyName', ascending=False).head(10).reset_index()
    traces.append(go.Bar(x=df['ComplaintType'], y=df.AgencyName, name=name.capitalize()))

iplot({'data': traces, 'layout': go.Layout(barmode='stack', xaxis={'tickangle': 40}, margin={'b': 150})}, filena
```



Wykorzystanie silnika bazodanowego SQLite

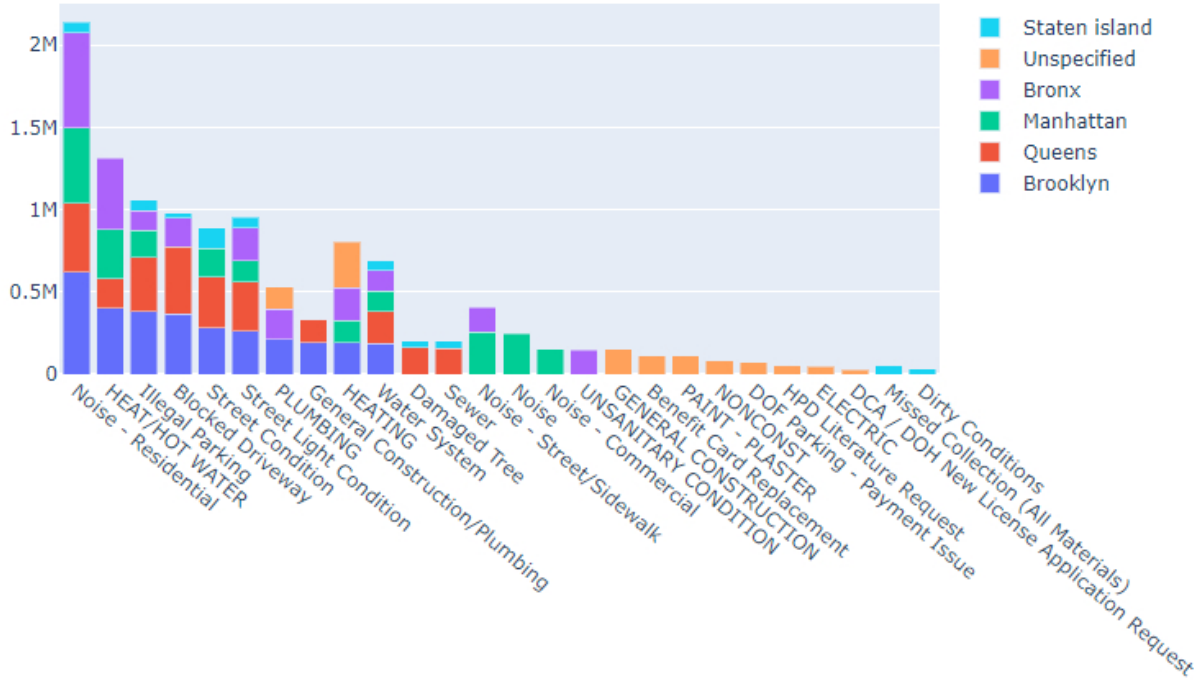
Wykorzystanie silnika bazodanowego SQLite

Przygotowanie plikowej bazy danych

In [18]:
```
if not db_file.exists():
    j = 0
    index_start = 1
    for df_sqlite3 in pd.read_csv(file, chunksize=chunksize,usecols=required_columns, iterator=True, encoding='u
        df_sqlite3 = df_sqlite3.rename(columns={c: c.replace(' ', '') for c in df_sqlite3.columns})
        df_sqlite3.index += index_start
        j+=1
        df_sqlite3.to_sql('data', disk_engine, if_exists='append')
        index_start = df_sqlite3.index[-1] + 1
```

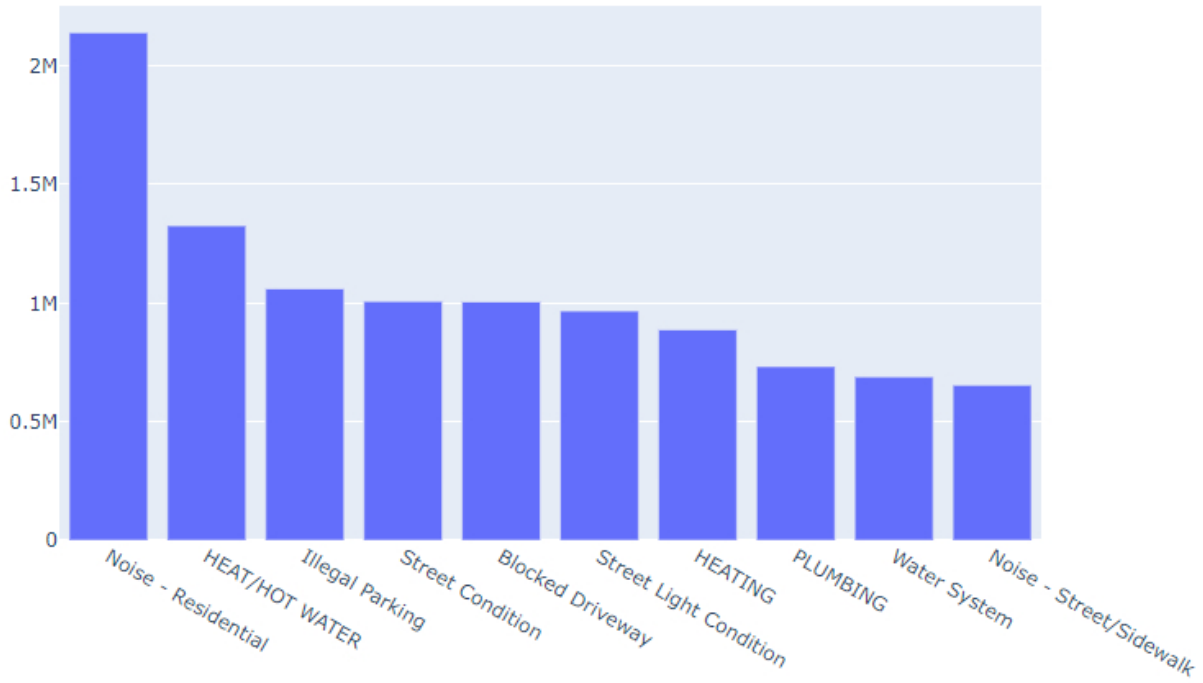
Najczęściej zgłaszane skargi w każdej dzielnicy.

In [14]:
```
df = pd.read_sql_query('SELECT Borough, COUNT(*) as `numbersOf` '
                       'FROM data '
                       'GROUP BY `Borough` '
                      'COLLATE NOCASE '
                      'ORDER BY -numbersOf '
                      'LIMIT 10 ', disk_engine)

boroughs = list(df.Borough)
if None in boroughs:
    boroughs.remove(None)

traces = []
for borough in boroughs:
    df = pd.read_sql_query('SELECT ComplaintType, COUNT(*) as `numbersOf` '
                           'FROM data '
                           'WHERE Borough = "{}" COLLATE NOCASE '
                           'GROUP BY `ComplaintType` '
                           'ORDER BY -numbersOf LIMIT 10'.format(borough), disk_engine)

    traces.append(go.Bar(x=df['ComplaintType'], y=df.numbersOf, name=borough.capitalize()))

iplot({'data': traces, 'layout': go.Layout(barmode='stack', xaxis={'tickangle': 40}, margin={'b': 150})}, filena
```



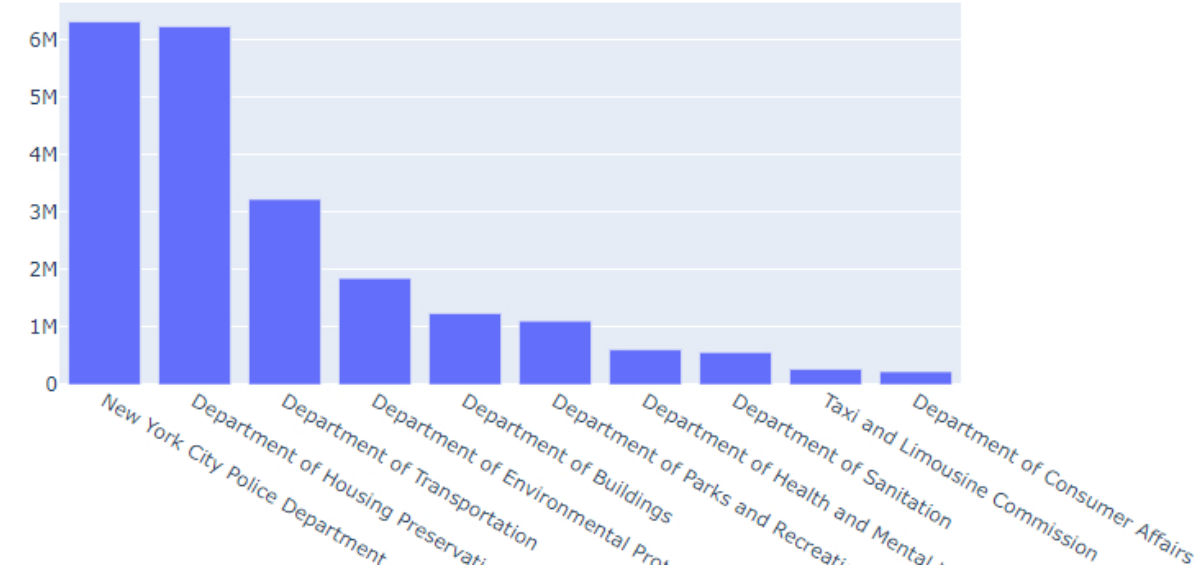
Najczęściej zgłaszane skargi.

In [15]:
```
df = pd.read_sql_query('SELECT ComplaintType, COUNT(*) as `numbersOf`'
                       'FROM data '
                       'GROUP BY ComplaintType '
                       'ORDER BY -numbersOf LIMIT 10', disk_engine)
iplot([go.Bar(x=df.ComplaintType, y=df.numbersOf)], filename='Najczesciej zglaszane skargi')
```



Urzędy, do których najczęściej zgłaszano skargi.

In [20]:
```
df = pd.read_sql_query('SELECT AgencyName, COUNT(*) as `numbersOf`'
                       'FROM data '
                       'GROUP BY AgencyName '
                       'ORDER BY -numbersOf LIMIT 10', disk_engine)
iplot([go.Bar(x=df.AgencyName, y=df.numbersOf)], filename='Najczestsze urzedy')
```



Wykorzystanie silnika bazodanowego MSSQL

Stworzenie bazy danych dla klienta MSSQL.

In [25]:
```
setup_conn = pyodbc.connect('Driver={SQL Server};'
                      'Server=localhost,1433;'
                      'Database=master;'
                      'UID=SA;'
                      'PWD=Abc12345678;', autocommit=True)

setup_cursor = setup_conn.cursor()
create_db_query = '''
IF NOT EXISTS(SELECT 1 FROM sys.databases WHERE name='PRequest')
BEGIN
    CREATE DATABASE PRequest
END;'''
setup_cursor.execute(create_db_query)
setup_cursor.close()
setup_conn.close()
```

```
Stworzenie bazy danych dla klienta MSSQL. Czas wykoniania: 0:00:01.684973
```

Połączenie się z bazą danych.

In [41]:
```
conn = pyodbc.connect('Driver={SQL Server};'
                      'Server=localhost,1433;'
                      'Database=PRequest;'
                      'UID=SA;'
                      'PWD=Abc12345678;')

cursor = conn.cursor()
```

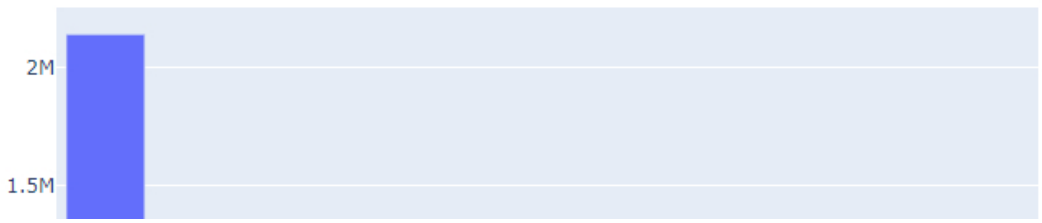
Stworzenie pliku 'export_dataframe.csv' z danymi do importu do bazy danych

In [27]:
```
df = pd.read_csv(file,usecols=required_columns, chunksize=chunksize)
chunk_list = []
for data_chunk in df:
   data_chunk = data_chunk.rename(columns={c: c.replace(' ', '') for c in data_chunk.columns})
   filtered_df = data_chunk.dropna()
   chunk_list.append(filtered_df)
chunk_data = pd.concat(chunk_list)
chunk_data.to_csv (r'export_dataframe.csv', index = False, header=True)
```

⚠ W tym miejscu należy zaimportować dane z pliku 'export_dataframe.csv' do bazy 'PRequest'

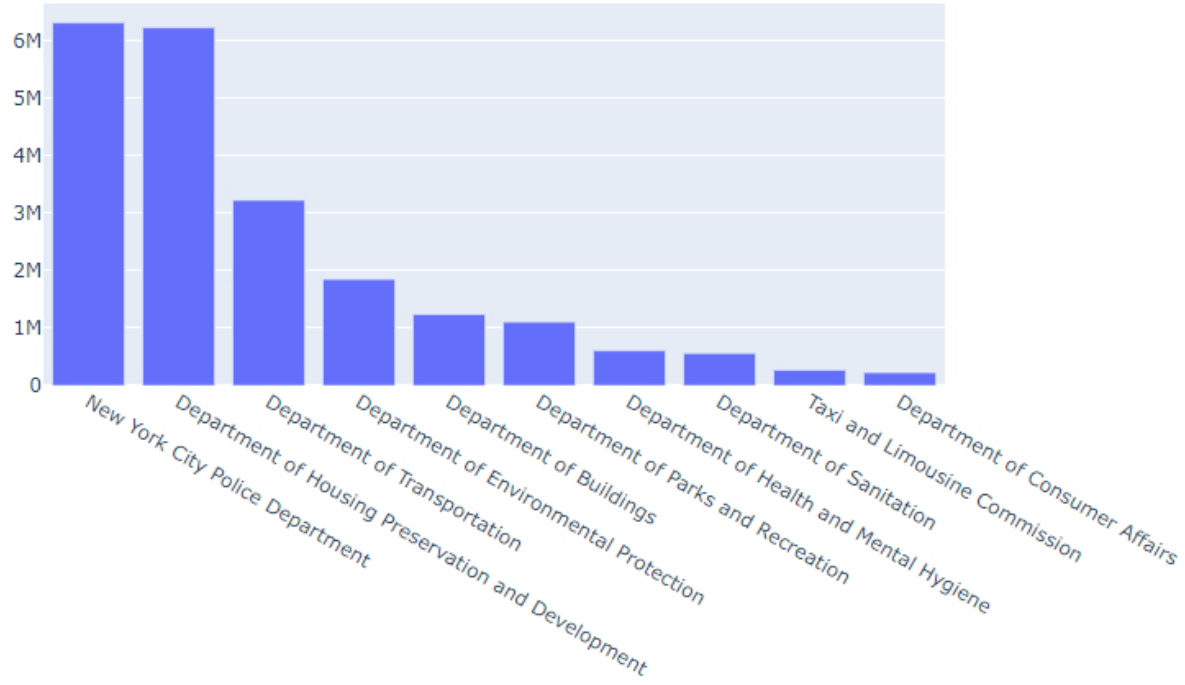
Najczęściej zgłaszane skargi.

In [42]:
```
sql_df = pd.read_sql_query('SELECT TOP 10 ComplaintType, COUNT(*) as numbersOf FROM export_dataframe_csv GROUP E
iplot([go.Bar(x=sql_df.ComplaintType, y=sql_df.numbersOf)], filename='Najczesciej zglaszane skargi')
```



Urzędy, do których najczęściej zgłaszano skargi.

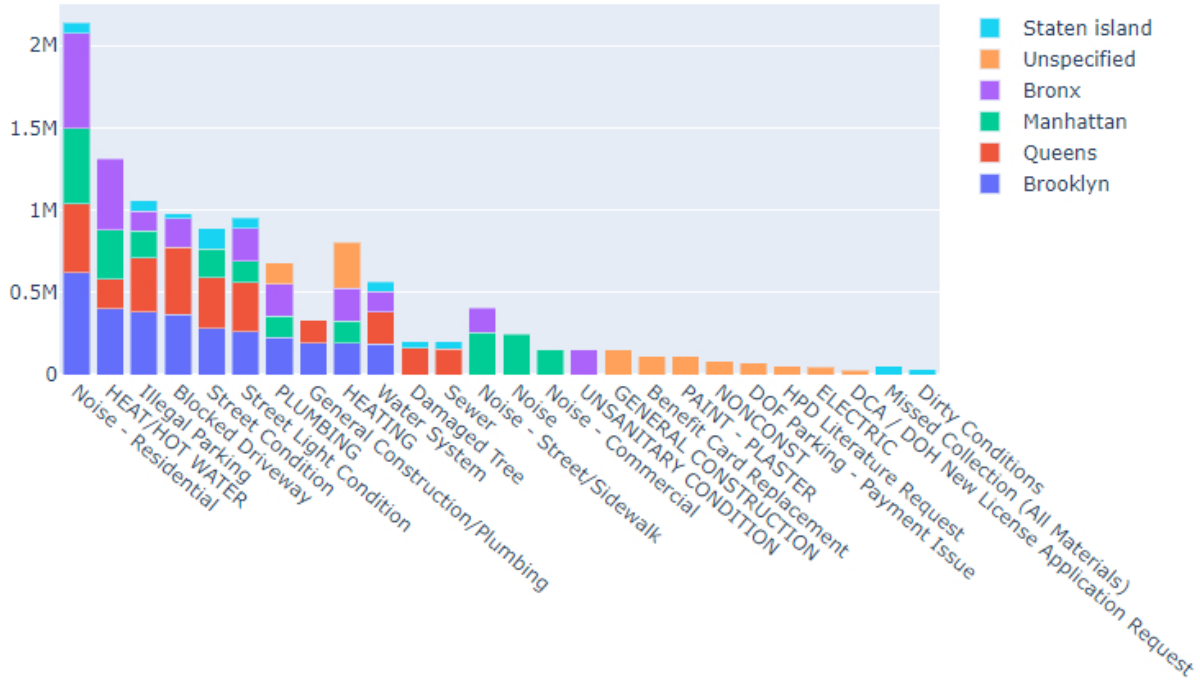
In [43]:
```
sql_df = pd.read_sql_query('SELECT TOP 10 AgencyName, COUNT(*) as numbersOf FROM export_dataframe_csv GROUP BY A
iplot([go.Bar(x=sql_df.AgencyName, y=sql_df.numbersOf)], filename='Najczesciej zglaszane skargi')
```



Najczęściej zgłaszane skargi w każdej dzielnicy.

In [44]:
```
sql_df = pd.read_sql_query('SELECT Borough, COUNT(*) as numbersOf '
                           'FROM export_dataframe_csv '
                           'GROUP BY Borough '
                           'ORDER BY numbersOf DESC', conn)
sql_df
boroughs = list(sql_df.Borough)
if None in boroughs:
    boroughs.remove(None)

traces = []
for borough in boroughs:
    df = pd.read_sql_query('SELECT TOP 10 ComplaintType, COUNT(*) as numbersOf '
                           'FROM export_dataframe_csv '
                           'WHERE Borough like \'{}\' '
                           'GROUP BY ComplaintType '
```

```
                            'ORDER BY numbersOf DESC'.format(borough), conn)

    traces.append(go.Bar(x=df['ComplaintType'], y=df.numbersOf, name=borough.capitalize()))

iplot({'data': traces, 'layout': go.Layout(barmode='stack', xaxis={'tickangle': 40}, margin={'b': 150})}, filena
```



In [ ]:
```
cursor.close()
conn.close()
```

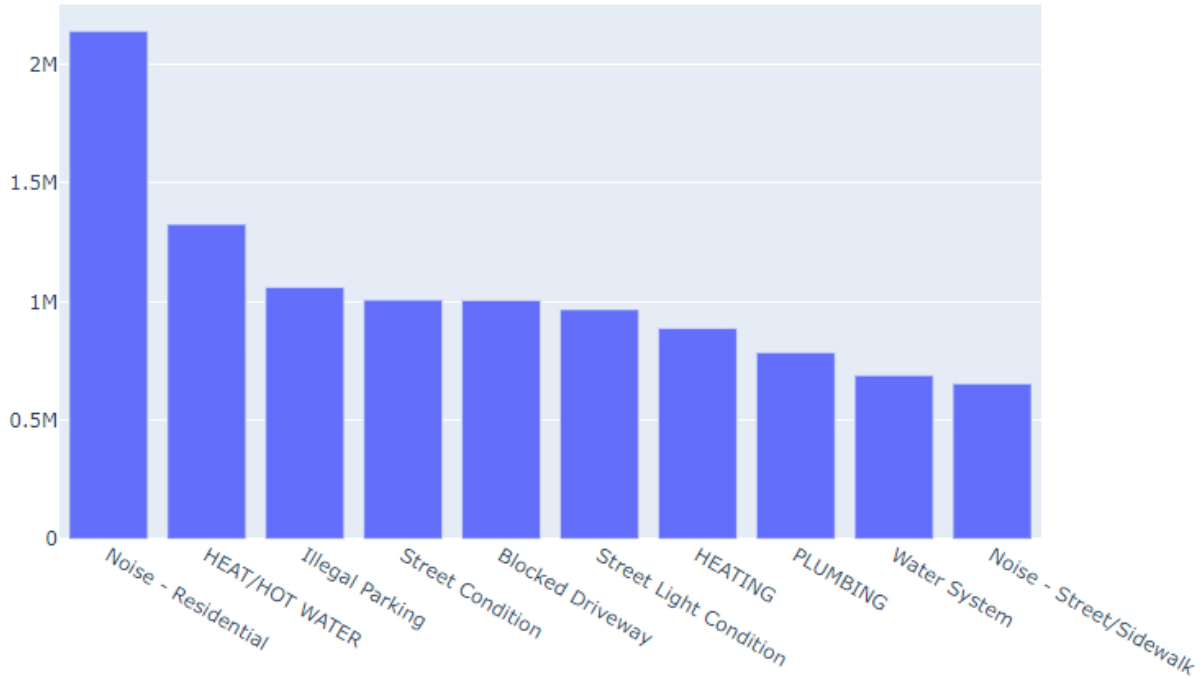
Zoptymalizowane kwerendy MSSQL

Utworzenie kolumny z id oraz stworzenie na niej indeksu

In [36]:
```
conn = pyodbc.connect('Driver={SQL Server};'
                      'Server=localhost,1433;'
                      'Database=PRequest;'
                      'UID=SA;'
                      'PWD=Abc12345678;', autocommit=True)
cursor = conn.cursor()

alter_table = '''
IF NOT EXISTS(SELECT 1 FROM sys.columns WHERE Name = N'id' AND Object_ID = Object_ID(N'export_dataframe_csv'))
BEGIN
        ALTER TABLE export_dataframe_csv
        ADD id BIGINT IDENTITY NOT NULL
END
'''
cursor.execute(alter_table).commit()

create_index = '''
IF NOT EXISTS(SELECT * FROM sys.indexes WHERE name = 'index_id' AND object_id = OBJECT_ID('export_dataframe_csv'
BEGIN
        CREATE UNIQUE INDEX index_id ON export_dataframe_csv (id)
END
'''

cursor.execute(create_index).commit()
```

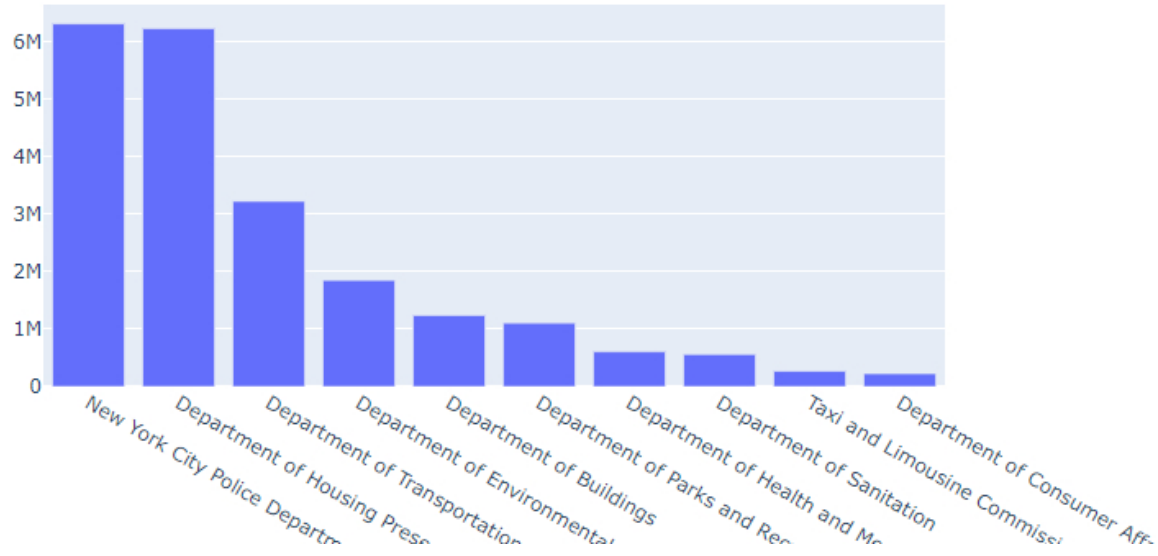
Optymalizacja: Najczęściej zgłaszane skargi

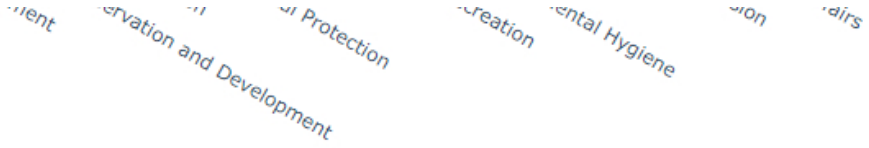
In [46]:
```
sql_df = pd.read_sql_query('SELECT TOP 10 ComplaintType, COUNT(*) as numbersOf '
                           'FROM export_dataframe_csv WITH(INDEX(index_id)) '
                           'GROUP BY ComplaintType '
                           'ORDER BY numbersOf desc', conn)
iplot([go.Bar(x=sql_df.ComplaintType, y=sql_df.numbersOf)], filename='Najczesciej zglaszane skargi')
```



Optymalizacja: Urzędy, do których najczęściej zgłaszano skargi.

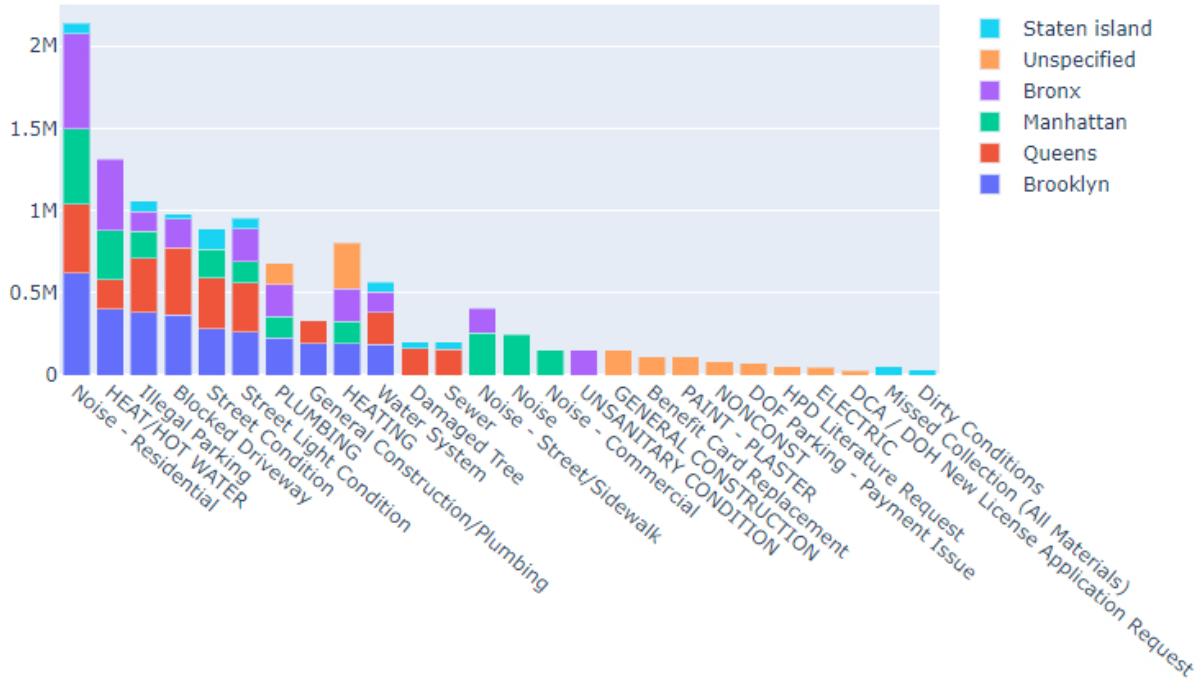
In [47]:
```
sql_df = pd.read_sql_query('SELECT TOP 10 AgencyName, COUNT(*) as numbersOf '
                           'FROM export_dataframe_csv WITH(INDEX(index_id)) '
                           'GROUP BY AgencyName '
                           'ORDER BY numbersOf desc', conn)
iplot([go.Bar(x=sql_df.AgencyName, y=sql_df.numbersOf)], filename='Najczesciej zglaszane skargi')
```





Optymalizacja: Najczęściej zgłaszane skargi w każdej dzielnicy.

In [48]:
```
sql_df = pd.read_sql_query('SELECT Borough, COUNT(*) as numbersOf '
                           'FROM export_dataframe_csv WITH(INDEX(index_id))'
                           'GROUP BY Borough '
                           'ORDER BY numbersOf DESC', conn)
sql_df
boroughs = list(sql_df.Borough)
if None in boroughs:
    boroughs.remove(None)

traces = []
for borough in boroughs:
    df = pd.read_sql_query('SELECT TOP 10 ComplaintType, COUNT(*) as numbersOf '
                           'FROM export_dataframe_csv WITH(INDEX(index_id))'
                           'WHERE Borough like \'{}\' '
                           'GROUP BY ComplaintType '
                           'ORDER BY numbersOf DESC'.format(borough), conn)

    traces.append(go.Bar(x=df['ComplaintType'], y=df.numbersOf, name=borough.capitalize()))

iplot({'data': traces, 'layout': go.Layout(barmode='stack', xaxis={'tickangle': 40}, margin={'b': 150})}, filena
```



In [ ]:
```
cursor.close()
conn.close()
```