In [1]:
```
import pandas as pd
import numpy as np
import os
import io

import matplotlib.pyplot as plt
import matplotlib
%matplotlib inline
```

In [2]:
```
import boto3
import sagemaker
```

In [24]:
```
s3_client = boto3.client('s3')
bucket_name='mvc-crashes'
```

In [25]:
```
obj_list=s3_client.list_objects(Bucket=bucket_name)

files=[]
for contents in obj_list['Contents']:
    files.append(contents['Key'])

print(files)
```

```
['crashes_by_street.csv']
```

In [26]:
```
file_name=files[0]

print(file_name)
```

```
crashes_by_street.csv
```

In [27]:
```
data_object = s3_client.get_object(Bucket=bucket_name, Key=file_name)
display(data_object)
```

```
{'ResponseMetadata': {'RequestId': '7ZFS0SFG0J1KEP1R',
  'HostId': 'UKHbo5+zI/mZoYQZe3O79yk3RQlS2jp6SdAy+HCeEY9sLa38y5DCrD24XvVt
KC72IivOgOSfIIA=',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amz-id-2': 'UKHbo5+zI/mZoYQZe3O79yk3RQlS2jp6SdAy+HCe
EY9sLa38y5DCrD24XvVtKC72IivOgOSfIIA=',
   'x-amz-request-id': '7ZFS0SFG0J1KEP1R',
   'date': 'Mon, 13 Jul 2020 03:26:17 GMT',
   'last-modified': 'Mon, 13 Jul 2020 02:19:08 GMT',
   'etag': '"f952f6edfc8ce8f4287340bdbc37d241"',
   'accept-ranges': 'bytes',
   'content-type': 'text/csv',
   'content-length': '1661051',
   'server': 'AmazonS3'},
  'RetryAttempts': 0},
 'AcceptRanges': 'bytes',
 'LastModified': datetime.datetime(2020, 7, 13, 2, 19, 8, tzinfo=tzutc
()),
 'ContentLength': 1661051,
 'ETag': '"f952f6edfc8ce8f4287340bdbc37d241"',
 'ContentType': 'text/csv',
 'Metadata': {},
 'Body': <botocore.response.StreamingBody at 0x7fe73c83d2b0>}
```

In [28]:
```
data_body = data_object["Body"].read()
```

In [29]:
```
data_stream = io.BytesIO(data_body)

streets_df = pd.read_csv(data_stream, header=0, delimiter=",")
streets_df
```

Out[29]:

	street_city	number_of_persons_injured	number_of_persons_killed	number_of_pedestrians_injure
0	100th Avenue, Queens, NY	97	2	
1	100th Drive, Queens, NY	4	0	
2	100th Road, Queens, NY	2	0	
3	100th Street, Kings, NY	8	0	
4	100th Street, Queens, NY	189	0	4
...	...	...	...	.
7985	Zoe Street, Richmond, NY	8	0	
7986	Zoller Road, Queens, NY	18	0	
7987	Zulette Avenue, Bronx, NY	28	0	
7988	Zwicky Avenue, Richmond, NY	12	0	
7989	ramp to Belt Parkway West, Nassau, NY	0	0	

7990 rows × 78 columns

In [20]:
```
print(streets_df.describe())
```

```
       number_of_persons_injured  number_of_persons_killed  \
count                7990.000000               7990.000000
mean                   77.245307                  0.340426
std                   242.904127                  1.296173
min                     0.000000                  0.000000
25%                     1.000000                  0.000000
50%                    10.000000                  0.000000
75%                    57.000000                  0.000000
max                  5485.000000                 37.000000

       number_of_pedestrians_injured  number_of_pedestrians_killed  \
count                    7990.000000                   7990.000000
mean                       17.032541                      0.197997
std                        54.473108                      0.822055
min                         0.000000                      0.000000
25%                         0.000000                      0.000000
50%                         1.000000                      0.000000
75%                        12.000000                      0.000000
max                      1195.000000                     17.000000

       number_of_cyclist_injured  number_of_cyclist_killed  \
count                7990.000000               7990.000000
mean                    6.874593                  0.026033
std                    25.427344                  0.180605
min                     0.000000                  0.000000
25%                     0.000000                  0.000000
50%                     0.000000                  0.000000
75%                     4.000000                  0.000000
max                   663.000000                  3.000000

       number_of_motorist_injured  number_of_motorist_killed  \
count                 7990.000000                7990.000000
mean                    53.338173                   0.115645
std                    186.824097                   0.645923
min                      0.000000                   0.000000
25%                      1.000000                   0.000000
50%                      7.000000                   0.000000
75%                     37.000000                   0.000000
max                   5427.000000                  29.000000

       factor__driver_inattention_distraction  \
count                             7990.000000
mean                                55.719274
std                                185.228996
min                                  0.000000
25%                                  1.000000
50%                                  7.000000
75%                                 38.000000
max                               4450.000000

       factor__failure_to_yield_right_of_way  ...          may         ju
ne  \
count                            7990.000000  ...  7990.000000  7990.0000
00
mean                               18.598874  ...    24.217021    24.3843
```

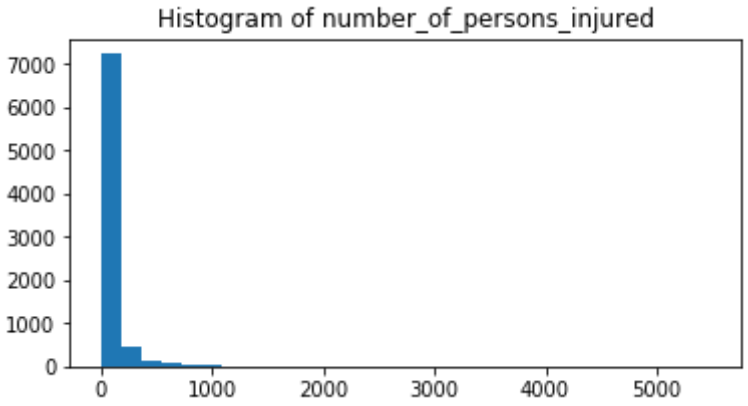
```
55
std                                  51.937144  ...    74.807837    74.4711
67
min                                   0.000000  ...     0.000000     0.0000
00
25%                                   0.000000  ...     1.000000     1.0000
00
50%                                   2.000000  ...     4.000000     4.0000
00
75%                                  14.000000  ...    19.000000    19.0000
00
max                                1030.000000  ...  1676.000000  1681.0000
00

                july       august    september      october     november  \
count  7990.000000  7990.000000  7990.000000  7990.000000  7990.000000
mean     26.292240    25.604506    25.733917    26.564205    25.715519
std      80.537048    79.788389    80.401488    83.398826    80.132129
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       1.000000     1.000000     1.000000     1.000000     1.000000
50%       4.000000     4.000000     4.000000     4.000000     4.000000
75%      20.000000    20.000000    20.000000    20.000000    20.000000
max    1741.000000  1761.000000  1875.000000  1924.000000  1855.000000

          december  is_intersection  is_not_intersection
count  7990.000000      7990.000000          7990.000000
mean     26.089237       234.863705            59.146308
std      80.782696       707.010971           356.909328
min       0.000000         0.000000             0.000000
25%       1.000000         6.000000             1.000000
50%       4.000000        36.000000             5.000000
75%      20.000000       189.000000            31.000000
max    1760.000000     19150.000000         16181.000000

[8 rows x 77 columns]
```

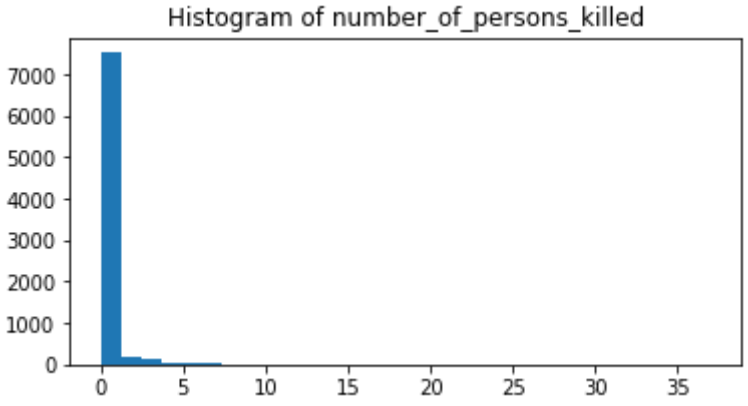
In [21]:
```
streets_df.columns
```

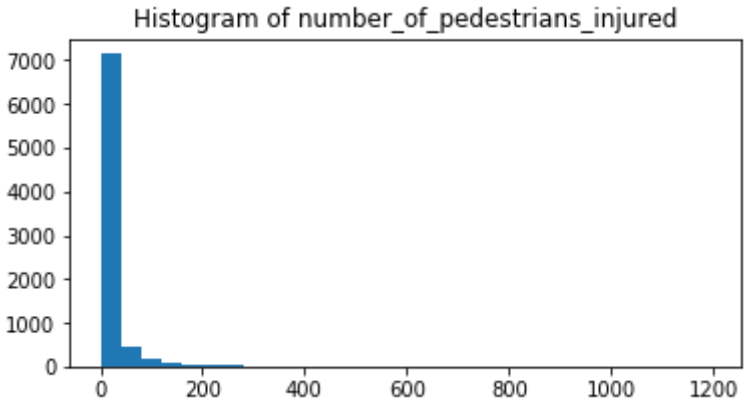
Out[21]:
```
Index(['street_city', 'number_of_persons_injured', 'number_of_persons_kil
led',
       'number_of_pedestrians_injured', 'number_of_pedestrians_killed',
       'number_of_cyclist_injured', 'number_of_cyclist_killed',
       'number_of_motorist_injured', 'number_of_motorist_killed',
       'factor__driver_inattention_distraction',
       'factor__failure_to_yield_right_of_way',
       'factor__following_too_closely', 'factor__backing_unsafely',
       'factor__other_vehicular', 'factor__fatigued_drowsy',
       'factor__turning_improperly', 'factor__passing_or_lane_usage_impro
per',
       'factor__passing_too_closely', 'factor__unsafe_lane_changing',
       'factor__traffic_control_disregarded', 'factor__driver_inexperienc
e',
       'vehicle__passenger_vehicle', 'vehicle__suv', 'vehicle__sedan',
       'vehicle__taxi', 'vehicle__van', 'vehicle__pick_up_truck',
       'vehicle__bus', 'vehicle__bicycle', 'vehicle__ambulance',
       'vehicle__tractor', 'vehicle__motorcycle', 'hour_0', 'hour_1', 'ho
ur_2',
       'hour_3', 'hour_4', 'hour_5', 'hour_6', 'hour_7', 'hour_8', 'hour_
9',
       'hour_10', 'hour_11', 'hour_12', 'hour_13', 'hour_14', 'hour_15',
       'hour_16', 'hour_17', 'hour_18', 'hour_19', 'hour_20', 'hour_21',
       'hour_22', 'hour_23', 'monday', 'tuesday', 'wednesday', 'thursda
y',
       'friday', 'saturday', 'sunday', 'is_weekend', 'january', 'februar
y',
       'march', 'april', 'may', 'june', 'july', 'august', 'september',
       'october', 'november', 'december', 'is_intersection',
       'is_not_intersection'],
      dtype='object')
```

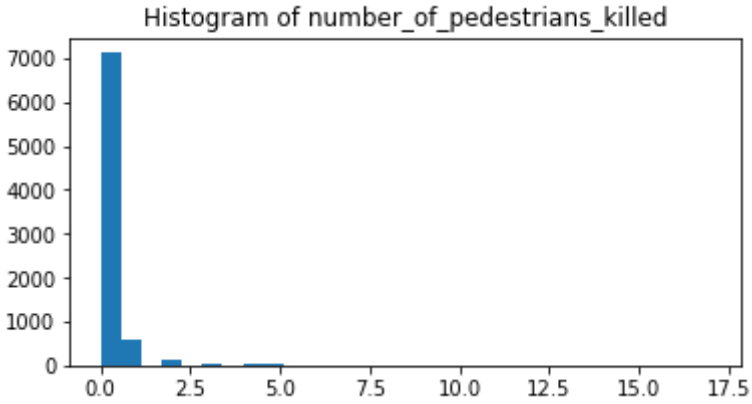
In [24]:
```
def plot_histograms(data, columns_list, n_bins = 30):
    for column_name in columns_list:
        ax = plt.subplots(figsize=(6,3))
        ax = plt.hist(data[column_name], bins=n_bins)
        title="Histogram of " + column_name
        plt.title(title, fontsize=12)
        plt.show()
```

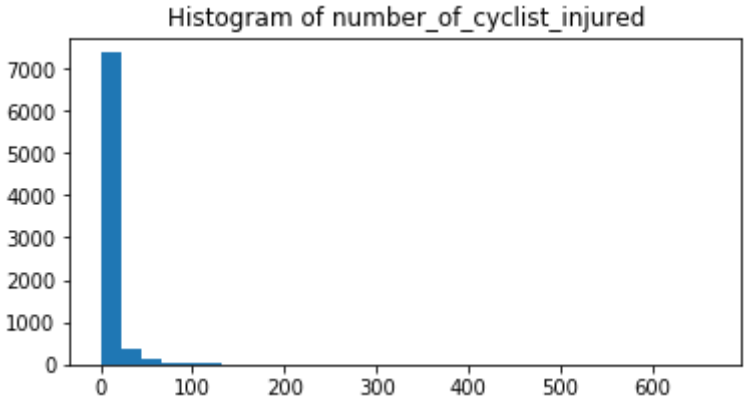
In [25]:
```
# Injured and killed persons
persons_list = ['number_of_persons_injured', 'number_of_persons_killed',
    'number_of_pedestrians_injured', 'number_of_pedestrians_killed',
    'number_of_cyclist_injured', 'number_of_cyclist_killed',
    'number_of_motorist_injured', 'number_of_motorist_killed']
plot_histograms(streets_df, persons_list)
```

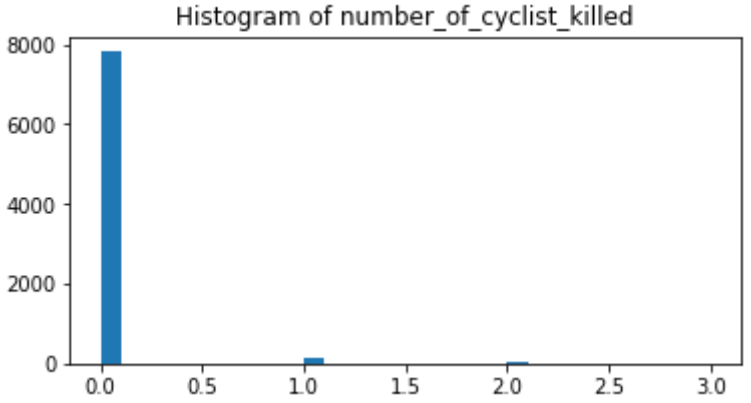


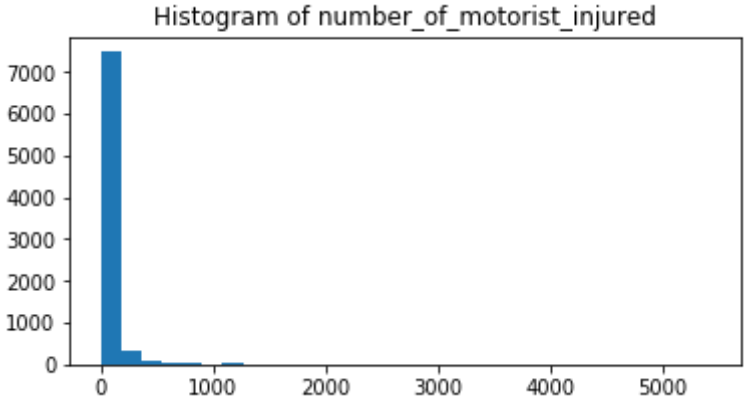


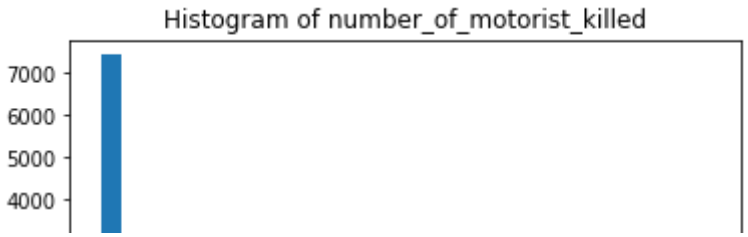



Histogram of number_of_pedestrians_killed
7000
6000
5000
4000
3000
2000
1000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

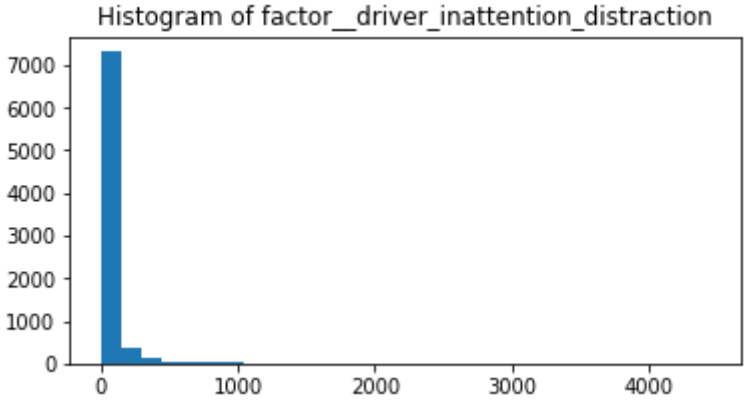

Histogram of number_of_cyclist_injured
7000
6000
5000
4000
3000
2000
1000
0
0
100
200
300
400
500
600

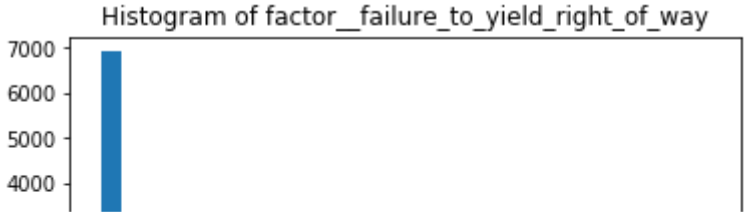

Histogram of number_of_cyclist_killed
8000
6000
4000
2000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0


Histogram of number_of_motorist_injured
7000
6000
5000
4000
3000
2000
1000
0
0
1000
2000
3000
4000
5000

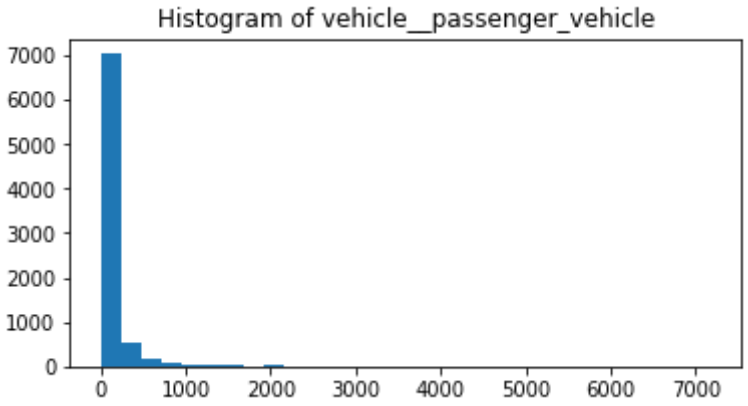


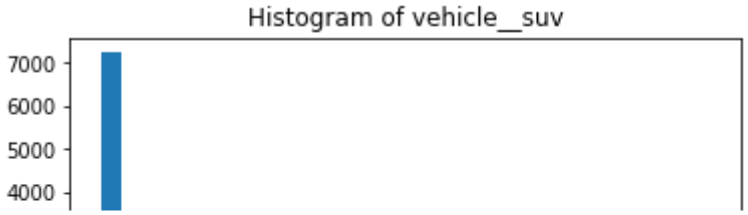
In [26]:
```
factors_list = [
    'factor__driver_inattention_distraction',
    'factor__failure_to_yield_right_of_way',
    'factor__following_too_closely', 'factor__backing_unsafely',
    'factor__other_vehicular', 'factor__fatigued_drowsy',
    'factor__turning_improperly', 'factor__passing_or_lane_usage_improper',
    'factor__passing_too_closely', 'factor__unsafe_lane_changing',
    'factor__traffic_control_disregarded', 'factor__driver_inexperience']
plot_histograms(streets_df, factors_list)
```



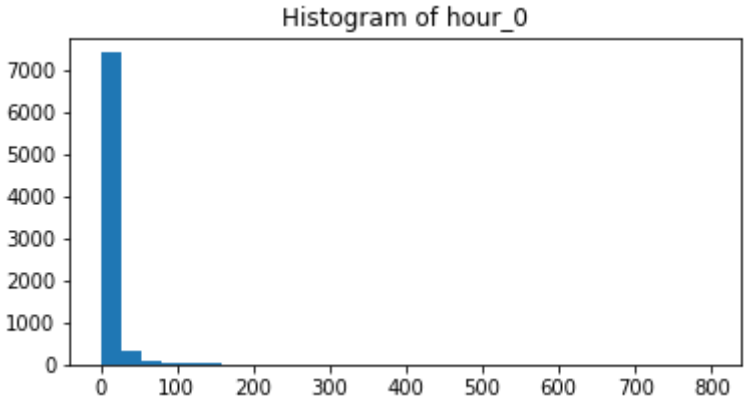


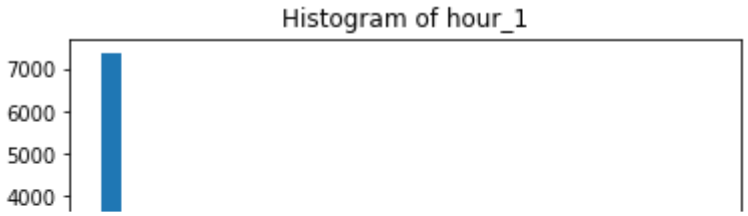
In [27]:
```
vehicles_list = [
    'vehicle__passenger_vehicle', 'vehicle__suv', 'vehicle__sedan',
    'vehicle__taxi', 'vehicle__van', 'vehicle__pick_up_truck',
    'vehicle__bus', 'vehicle__bicycle', 'vehicle__ambulance',
    'vehicle__tractor', 'vehicle__motorcycle']
plot_histograms(streets_df, vehicles_list)
```





In [28]:
```
datetime_list = [
    'hour_0', 'hour_1', 'hour_2',
    'hour_3', 'hour_4', 'hour_5', 'hour_6', 'hour_7', 'hour_8', 'hour_9',
    'hour_10', 'hour_11', 'hour_12', 'hour_13', 'hour_14', 'hour_15',
    'hour_16', 'hour_17', 'hour_18', 'hour_19', 'hour_20', 'hour_21',
    'hour_22', 'hour_23', 'monday', 'tuesday', 'wednesday', 'thursday',
    'friday', 'saturday', 'sunday', 'is_weekend', 'january', 'february',
    'march', 'april', 'may', 'june', 'july', 'august', 'september',
    'october', 'november', 'december']
plot_histograms(streets_df, datetime_list)
```





In [30]:
```
streets_df.index = streets_df['street_city']
del streets_df['street_city']
```

In [31]:
```
streets_df.head()
```

Out[31]:

street_city	number_of_persons_injured	number_of_persons_killed	number_of_pedestrians_injured	nur
100th Avenue, Queens, NY	97	2	3	
100th Drive, Queens, NY	4	0	0	
100th Road, Queens, NY	2	0	0	
100th Street, Kings, NY	8	0	2	
100th Street, Queens, NY	189	0	48	

5 rows × 77 columns

In [32]:
```
from sklearn import preprocessing
scaler = preprocessing.MinMaxScaler()
scaler.fit(streets_df)
columns = streets_df.columns
temp = streets_df.copy()
temp[columns] = scaler.transform(streets_df[columns])
streets_df_scaled = temp
```

In [33]:
```
streets_df_scaled.head()
```

Out[33]:

street_city	number_of_persons_injured	number_of_persons_killed	number_of_pedestrians_injured	nur
100th Avenue, Queens, NY	0.017685	0.054054	0.002510	
100th Drive, Queens, NY	0.000729	0.000000	0.000000	
100th Road, Queens, NY	0.000365	0.000000	0.000000	
100th Street, Kings, NY	0.001459	0.000000	0.001674	
100th Street, Queens, NY	0.034458	0.000000	0.040167	

5 rows × 77 columns

In [34]:
```
print(streets_df_scaled.describe())
```

```
       number_of_persons_injured  number_of_persons_killed  \
count                7990.000000               7990.000000
mean                    0.014083                  0.009201
std                     0.044285                  0.035032
min                     0.000000                  0.000000
25%                     0.000182                  0.000000
50%                     0.001823                  0.000000
75%                     0.010392                  0.000000
max                     1.000000                  1.000000

       number_of_pedestrians_injured  number_of_pedestrians_killed  \
count                    7990.000000                   7990.000000
mean                        0.014253                      0.011647
std                         0.045584                      0.048356
min                         0.000000                      0.000000
25%                         0.000000                      0.000000
50%                         0.000837                      0.000000
75%                         0.010042                      0.000000
max                         1.000000                      1.000000

       number_of_cyclist_injured  number_of_cyclist_killed  \
count                7990.000000               7990.000000
mean                    0.010369                  0.008678
std                     0.038352                  0.060202
min                     0.000000                  0.000000
25%                     0.000000                  0.000000
50%                     0.000000                  0.000000
75%                     0.006033                  0.000000
max                     1.000000                  1.000000

       number_of_motorist_injured  number_of_motorist_killed  \
count                 7990.000000                7990.000000
mean                     0.009828                   0.003988
std                      0.034425                   0.022273
min                      0.000000                   0.000000
25%                      0.000184                   0.000000
50%                      0.001290                   0.000000
75%                      0.006818                   0.000000
max                      1.000000                   1.000000

       factor__driver_inattention_distraction  \
count                             7990.000000
mean                                 0.012521
std                                  0.041624
min                                  0.000000
25%                                  0.000225
50%                                  0.001573
75%                                  0.008539
max                                  1.000000

       factor__failure_to_yield_right_of_way  ...          may         ju
ne  \
count                           7990.000000  ...  7990.000000  7990.0000
00
mean                               0.018057  ...     0.014449     0.0145
```

```
06
std                                  0.050424  ...     0.044635     0.0443
02
min                                  0.000000  ...     0.000000     0.0000
00
25%                                  0.000000  ...     0.000597     0.0005
95
50%                                  0.001942  ...     0.002387     0.0023
80
75%                                  0.013592  ...     0.011337     0.0113
03
max                                  1.000000  ...     1.000000     1.0000
00
```

	july	august	september	october	november	\
count	7990.000000	7990.000000	7990.000000	7990.000000	7990.000000	
mean	0.015102	0.014540	0.013725	0.013807	0.013863	
std	0.046259	0.045309	0.042881	0.043347	0.043198	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000574	0.000568	0.000533	0.000520	0.000539	
50%	0.002298	0.002271	0.002133	0.002079	0.002156	
75%	0.011488	0.011357	0.010667	0.010395	0.010782	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

	december	is_intersection	is_not_intersection
count	7990.000000	7990.000000	7990.000000
mean	0.014823	0.012264	0.003655
std	0.045899	0.036920	0.022057
min	0.000000	0.000000	0.000000
25%	0.000568	0.000313	0.000062
50%	0.002273	0.001880	0.000309
75%	0.011364	0.009869	0.001916
max	1.000000	1.000000	1.000000

[8 rows x 77 columns]

Data Modeling

In [51]:
```
from sagemaker import get_execution_role

session = sagemaker.Session()
role = get_execution_role()
print(role)
```

```
arn:aws:iam::006275120779:role/service-role/AmazonSageMaker-ExecutionRole
-20200401T204101
```

In [52]:
```
bucket_name2 = session.default_bucket()
print(bucket_name2)
```

```
sagemaker-us-east-1-006275120779
```

In [53]:
```
prefix = 'streets'
output_path='s3://{}/{}/'.format(bucket_name2, prefix)
```

In [54]:
```
from sagemaker import PCA

# 77 - 1
N_COMPONENTS=76

pca_SM = PCA(role=role,
             train_instance_count=1,
             train_instance_type='ml.c4.xlarge',
             output_path=output_path,
             num_components=N_COMPONENTS,
             sagemaker_session=session)
```

In [57]:
```
# convert df to np array
train_data_np = streets_df_scaled.values.astype('float32')

# convert to RecordSet format
formatted_train_data = pca_SM.record_set(train_data_np)
```

In [58]:
```
%%time

# train the PCA mode on the formatted data
pca_SM.fit(formatted_train_data)
```

```
'get_image_uri' method will be deprecated in favor of 'ImageURIProvide
r' class in SageMaker Python SDK v2.
's3_input' class will be renamed to 'TrainingInput' in SageMaker Python
SDK v2.
'get_image_uri' method will be deprecated in favor of 'ImageURIProvide
r' class in SageMaker Python SDK v2.
```

In [41]:
```
# Use created model

training_job_name='pca-2020-07-13-03-00-44-434'

# where the model is saved, by default
model_key = os.path.join(prefix, training_job_name, 'output/model.tar.gz')
print(model_key)

# download and unzip model
boto3.resource('s3').Bucket(bucket_name2).download_file(model_key, 'model.t

# unzipping as model_algo-1
os.system('tar -zxvf model.tar.gz')
os.system('unzip model_algo-1')
```

```
streets/pca-2020-07-13-03-00-44-434/output/model.tar.gz
```

Out[41]:
```
2304
```

In [6]:
```
import mxnet as mx

# loading the unzipped artifacts
pca_model_params = mx.ndarray.load('model_algo-1')

# what are the params
print(pca_model_params)
```

```
{'s':
[           nan            nan 1.61837239e-03 2.15460802e-03
 7.68422261e-02 1.04500875e-01 1.73892722e-01 1.84098914e-01
 1.93696454e-01 2.03601733e-01 2.15352640e-01 2.28391692e-01
 2.37656131e-01 2.48948261e-01 2.54175484e-01 2.59164184e-01
 2.62757212e-01 2.69009054e-01 2.77671605e-01 2.89450198e-01
 2.91878551e-01 2.99390614e-01 3.05442780e-01 3.08743745e-01
 3.13480824e-01 3.21597338e-01 3.24649662e-01 3.33393246e-01
 3.40419382e-01 3.43046516e-01 3.49858493e-01 3.54105979e-01
 3.58261049e-01 3.63609254e-01 3.69137257e-01 3.83826435e-01
 3.91432166e-01 3.91978085e-01 3.97146702e-01 4.28536564e-01
 4.51678574e-01 4.62648153e-01 4.79570627e-01 5.00418127e-01
 5.14542878e-01 5.30971646e-01 5.40328801e-01 5.71223617e-01
 5.98456621e-01 6.18360758e-01 6.34087682e-01 6.57845974e-01
 6.87686443e-01 7.17521966e-01 7.28587568e-01 8.07570577e-01
 8.39793086e-01 9.32936788e-01 9.81311202e-01 1.05359292e+00
 1.10795975e+00 1.23650944e+00 1.29207957e+00 1.40506709e+00
 1.50001490e+00 1.70452201e+00 1.77418506e+00 1.97279155e+00
 2.20480990e+00 2.30336046e+00 2.77320051e+00 2.90516949e+00
 3.92182779e+00 5.04642296e+00 5.60239744e+00 3.10696869e+01]
<NDArray 76 @cpu(0)>, 'v':
[[-2.39437491e-01 -4.98337656e-01  1.39463961e-01 ... -6.72872588e-02
   1.53611898e-01  1.20797202e-01]
 [-4.75900561e-05  5.76116981e-05  2.57609536e-05 ... -1.35817677e-01
   1.13361344e-01  8.41522366e-02]
 [ 5.21663427e-02  1.08563505e-01 -3.03885750e-02 ...  3.04747820e-02
  -2.55745500e-01  1.15788981e-01]
 ...
 [ 1.35612205e-01 -9.04593689e-05  1.44306466e-01 ...  1.19618354e-02
   1.83542818e-02  1.31415620e-01]
 [-3.80834132e-01  5.17988443e-01  8.01923573e-02 ...  4.28751856e-02
  -1.66905314e-01  9.95390266e-02]
 [-3.21794659e-01  4.37684000e-01  6.77695870e-02 ... -3.84404883e-02
   2.21221820e-01  4.26605307e-02]]
<NDArray 77x76 @cpu(0)>, 'mean':
[[0.01408301 0.00920069 0.01425317 0.01164691 0.01036892 0.00867751
  0.0098283  0.00398774 0.01252118 0.01805716 0.00375032 0.01650473
  0.00479393 0.00331835 0.0108529  0.01345924 0.01578456 0.00614963
  0.02182937 0.01078247 0.01700342 0.01362125 0.01540171 0.005129
  0.00840146 0.01057519 0.00951654 0.00801217 0.00952406 0.00518505
  0.0136849  0.01065271 0.01141754 0.01119827 0.01223712 0.01143695
  0.00876475 0.00848677 0.01189411 0.01561558 0.01330122 0.01245842
  0.01432022 0.0153713  0.01541784 0.01548167 0.01645774 0.01525182
  0.01565803 0.01617205 0.01503047 0.01361453 0.01336425 0.01207921
  0.01221644 0.01582743 0.01425073 0.0145228  0.01354602 0.01381356
  0.01444321 0.01570977 0.01500784 0.01450543 0.01565738 0.015088
  0.0142402  0.0144493  0.01450586 0.01510181 0.01453975 0.01372476
  0.01380676 0.01386281 0.01482343 0.01226442 0.00365529]]
<NDArray 1x77 @cpu(0)>}
```

In [7]:
```
# get selected params
s=pd.DataFrame(pca_model_params['s'].asnumpy())
v=pd.DataFrame(pca_model_params['v'].asnumpy())
```

In [8]:
```
# 77 - 1
N_COMPONENTS=76
```

In [9]:
```
# looking at top 5 components
n_principal_components = 5

start_idx = N_COMPONENTS - n_principal_components  # 76-n

# print a selection of s
print(s.iloc[start_idx:, :])
```

```
            0
71   2.905169
72   3.921828
73   5.046423
74   5.602397
75  31.069687
```

In [10]:
```
def explained_variance(s, n_top_components):
    '''Calculates the approx. data variance that n_top_components captures.
       :param s: A dataframe of singular values for top components;
           the top value is in the last row.
       :param n_top_components: An integer, the number of top components to
       :return: The expected data variance covered by the n_top_components.
    return s[-n_top_components:].pow(2).sum()/s.pow(2).sum()
```

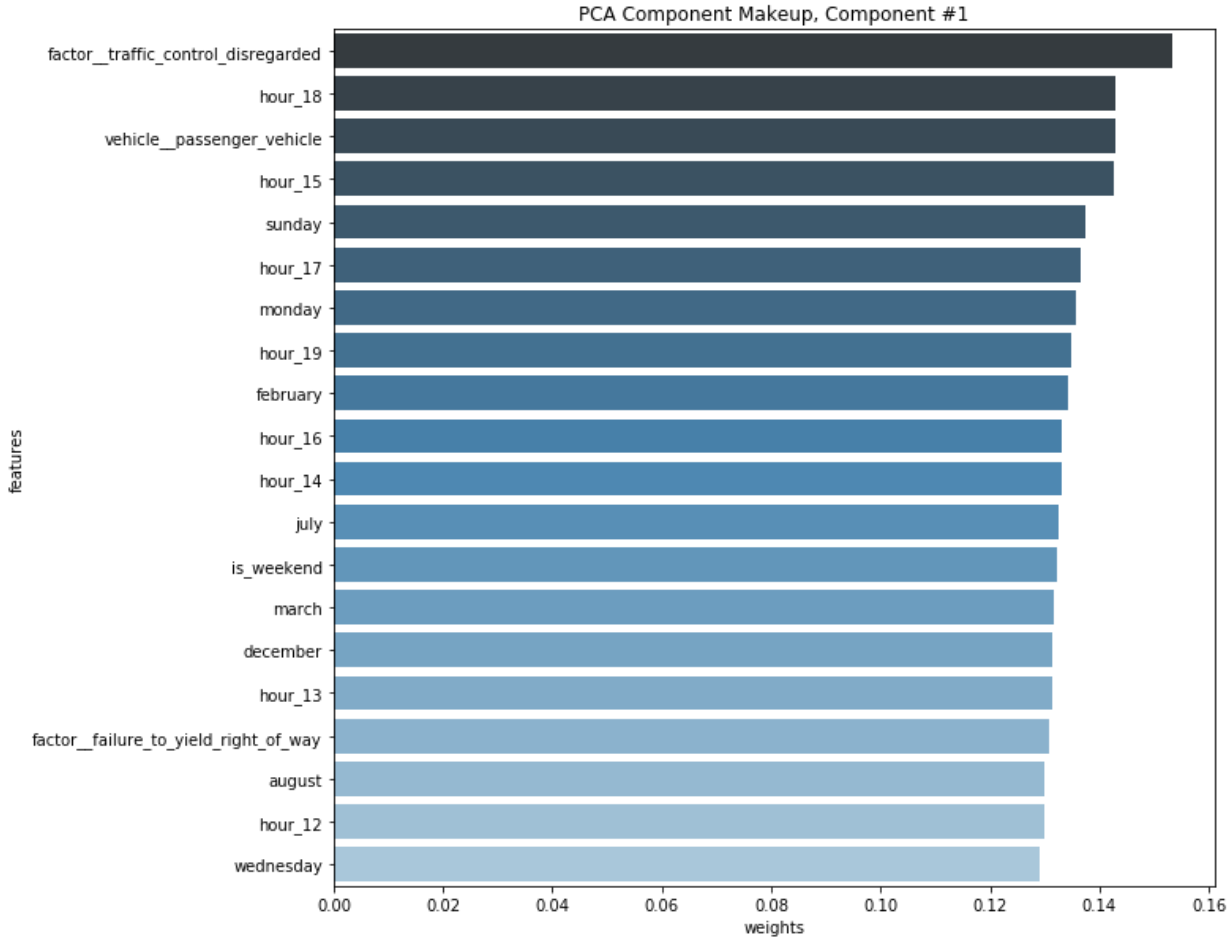
In [166]:
```
# test cell
n_top_components = 5 # select a value for the number of top components

# calculate the explained variance
exp_variance = explained_variance(s, n_top_components)
print('Explained variance: ', exp_variance)
```

```
Explained variance:  0    0.955097
dtype: float32
```

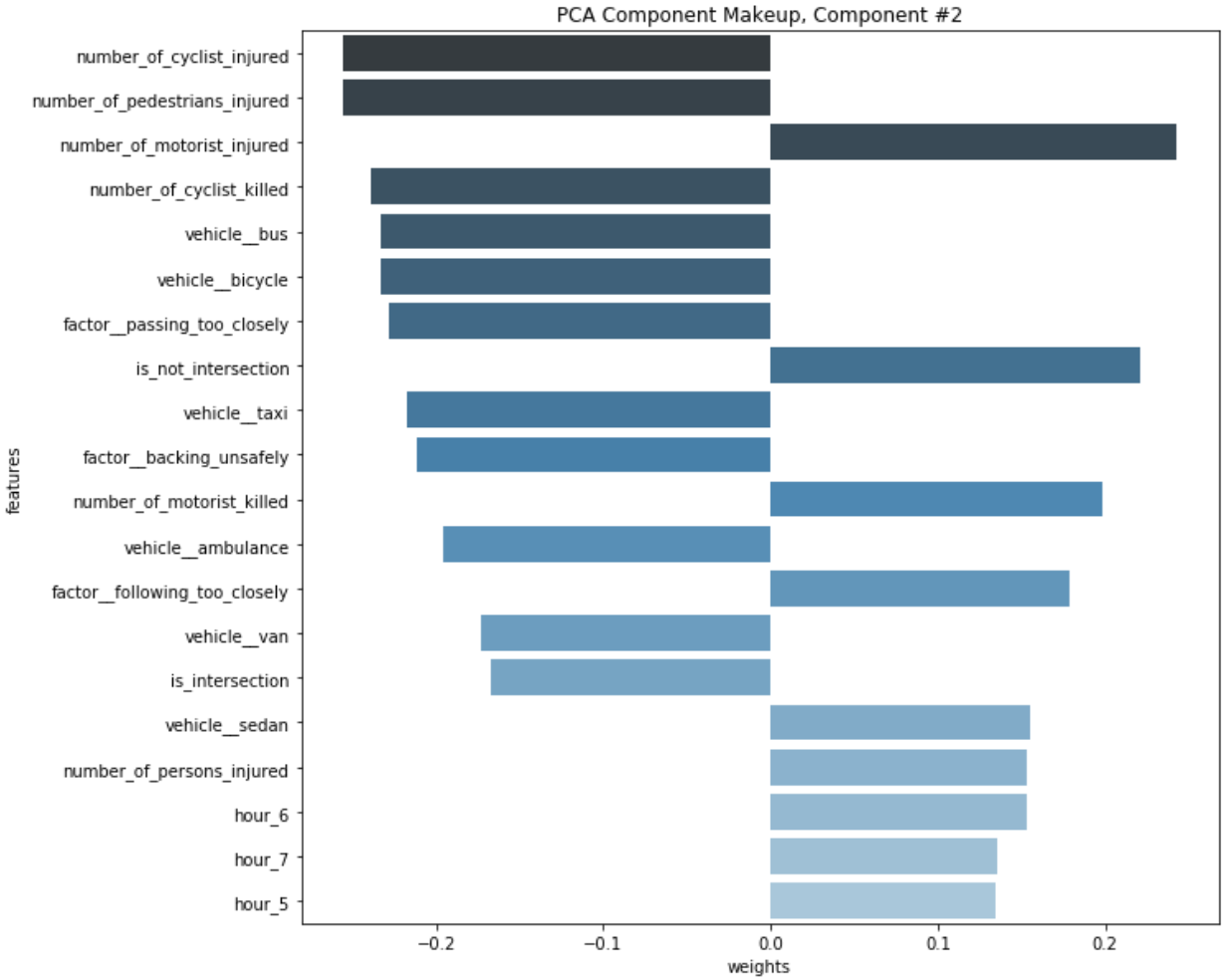
In [43]:

```
import seaborn as sns

def display_component(v, features_list, component_num, n_weights=10):

    # get index of component (last row - component_num)
    row_idx = N_COMPONENTS-component_num

    # get the list of weights from a row in v, dataframe
    v_1_row = v.iloc[:, row_idx]
    v_1 = np.squeeze(v_1_row.values)

    # match weights to features in counties_scaled dataframe, using list co
    comps = pd.DataFrame(list(zip(v_1, features_list)),
                         columns=['weights', 'features'])

    # we'll want to sort by the largest n_weights
    # weights can be neg/pos and we'll sort by magnitude
    comps['abs_weights']=comps['weights'].apply(lambda x: np.abs(x))
    sorted_weight_data = comps.sort_values('abs_weights', ascending=False).

    # display using seaborn
    ax=plt.subplots(figsize=(10,10))
    ax=sns.barplot(data=sorted_weight_data,
                   x="weights",
                   y="features",
                   palette="Blues_d")
    ax.set_title("PCA Component Makeup, Component #" + str(component_num))
    plt.show()
```

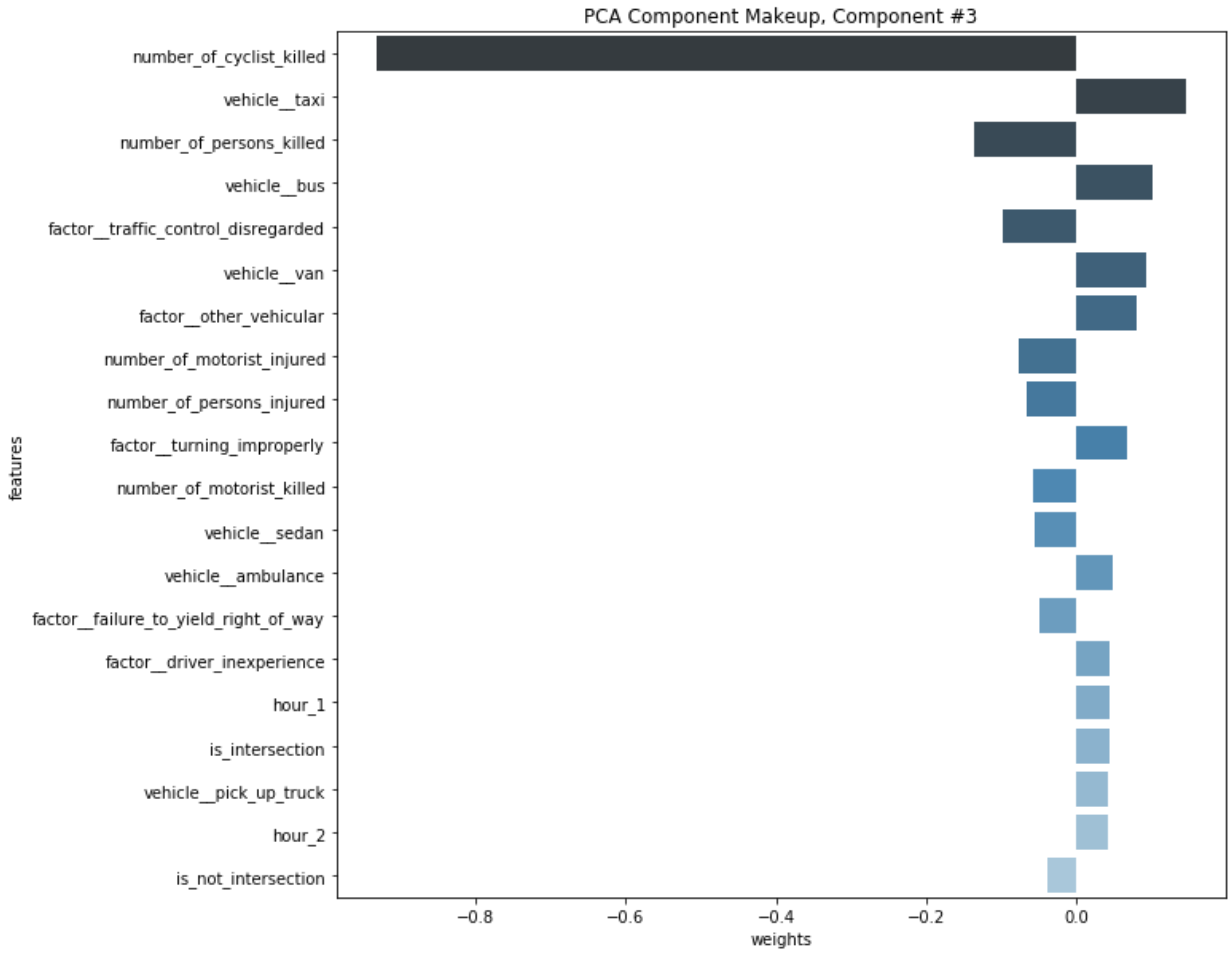
In [44]:
```
# display makeup of first component
num=1
display_component(v, streets_df_scaled.columns.values, component_num=num, n
```



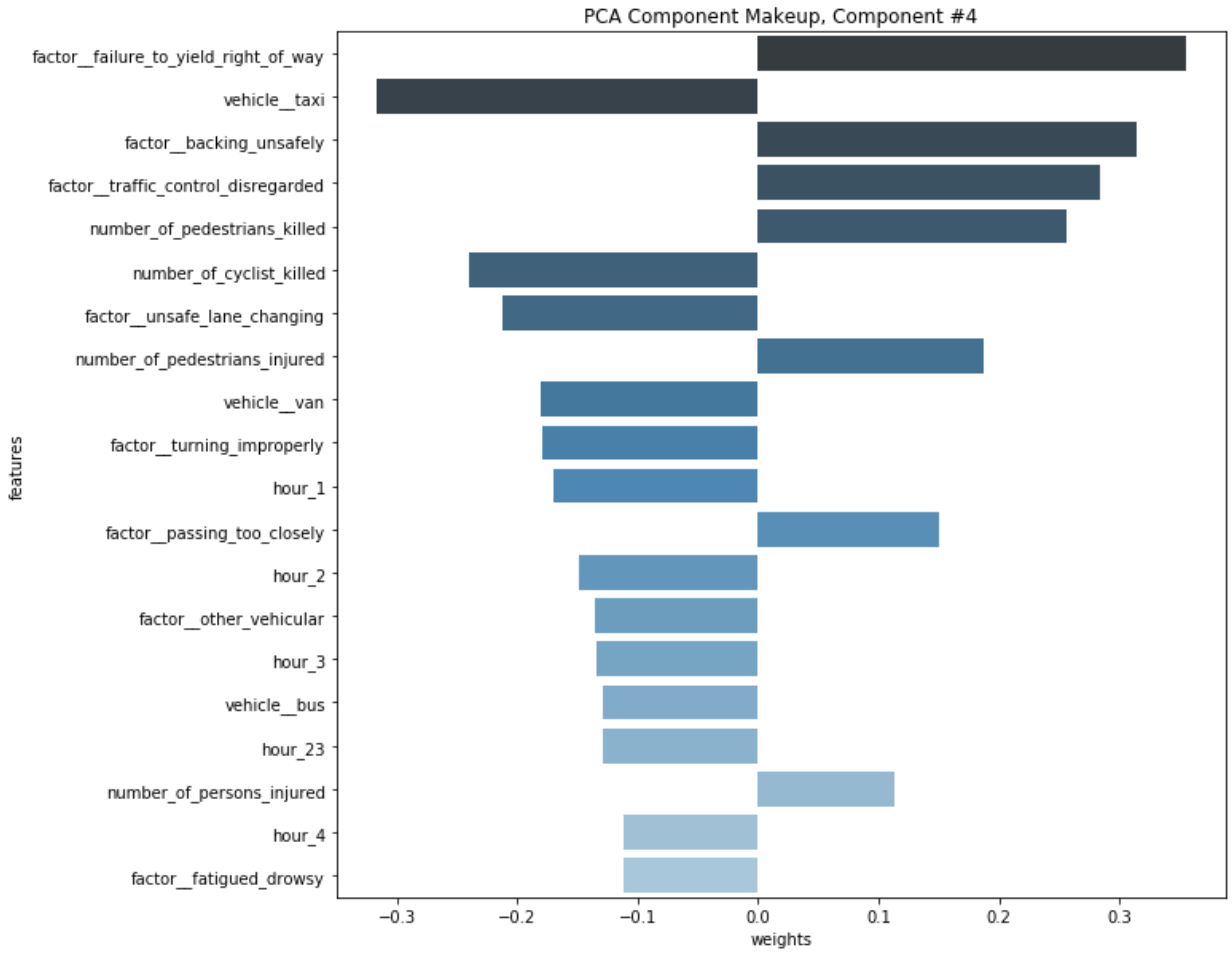
In [45]:
```
# display makeup of first component
num=2
display_component(v, streets_df_scaled.columns.values, component_num=num, n
```



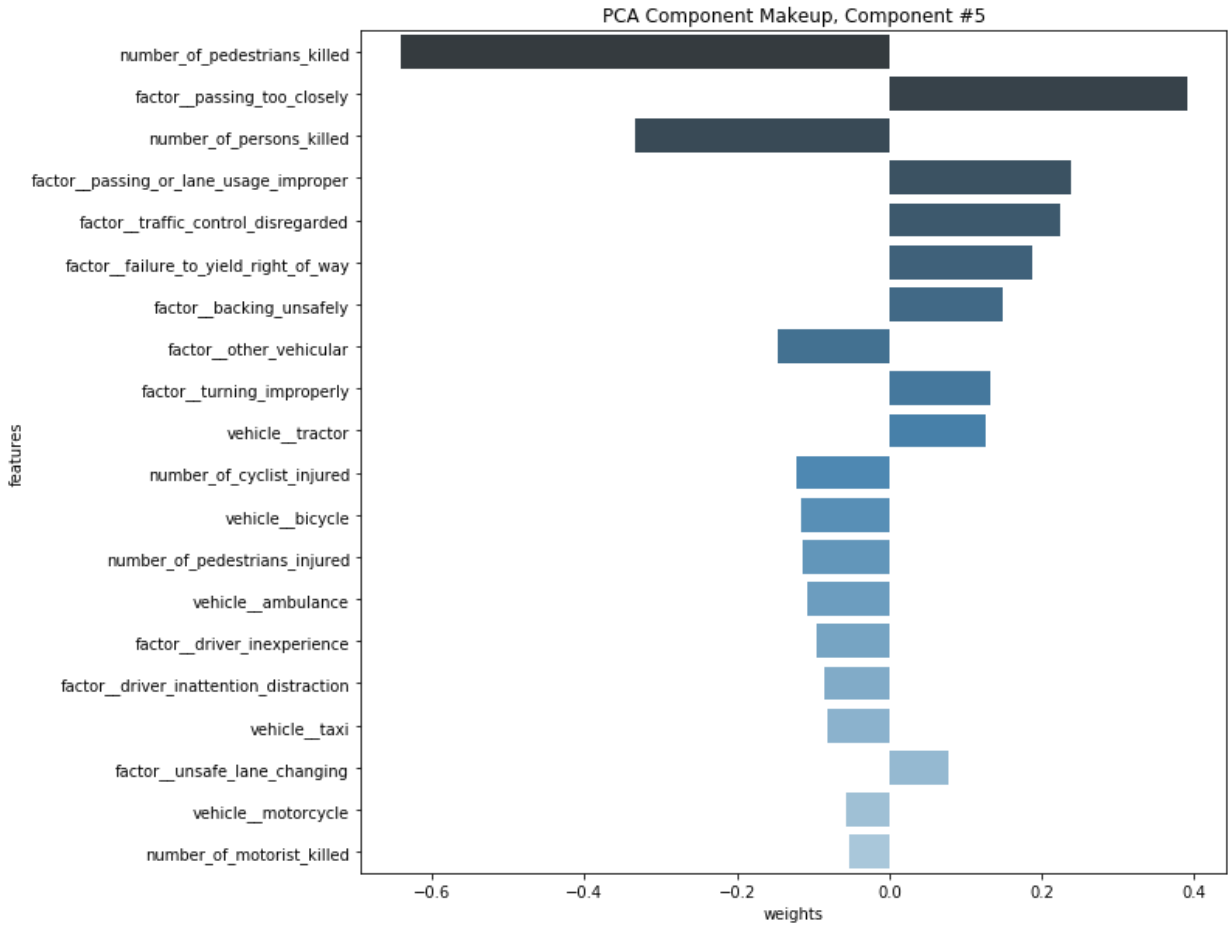
In [46]:
```
# display makeup of first component
num=3
display_component(v, streets_df_scaled.columns.values, component_num=num, n
```



In [47]:
```
# display makeup of first component
num=4
display_component(v, streets_df_scaled.columns.values, component_num=num, n
```



In [48]:
```
# display makeup of first component
num=5
display_component(v, streets_df_scaled.columns.values, component_num=num, n
```



Deploy PCA model

In [60]:
```
%%time

pca_predictor = pca_SM.deploy(initial_instance_count=1,
                              instance_type='ml.t2.medium')
```

```
Parameter image will be renamed to image_uri in SageMaker Python SDK v2.
Using already existing model: pca-2020-07-13-03-40-31-685

-----------------!CPU times: user 292 ms, sys: 23.5 ms, total: 315 ms
Wall time: 8min 33s
```

In [61]:
```
train_pca = pca_predictor.predict(train_data_np)
```

In [62]:
```
data_idx = 0
print(train_pca[data_idx])
```

```
label {
  key: "projection"
  value {
    float32_tensor {
      values: 5.76805234686617e-07
      values: 6.698915200331612e-08
      values: 2.992140366586682e-07
      values: 1.5649349904833798e-07
      values: -0.000299029954476282
      values: -0.00033904460724443197
      values: 0.0008863899856805801
      values: 0.0018049577483907342
      values: 0.001904990989714861
      values: -0.0036464992444962263
      values: -0.0012257921043783426
      values: 0.004078459460288286
      values: -0.001398534863255918
      values: 0.0017021134262904525
      values: -0.002795390086248517
      values: 0.004608123563230038
      values: -0.0021518520079553127
      values: 0.00021773693151772022
      values: -3.674867912195623e-05
      values: 0.001865985686890781
      values: -0.0007005211664363742
      values: -0.0012943560723215342
      values: -5.8722798712551594e-05
      values: 0.0030160327441990376
      values: 0.0008709158282727003
      values: -0.002565366681665182
      values: 0.0010897178435698152
      values: -0.00031977181788533926
      values: 0.00134608568623662
      values: -0.00032960085081867874
      values: 0.004466602113097906
      values: -0.00552052166312933
      values: -0.0013866052031517029
      values: -0.00016887200763449073
      values: -0.0038374343421310186
      values: 0.00165913428645581
      values: 9.324507118435577e-05
      values: 0.0009145416552200913
      values: -0.00429937057197094
      values: -0.0009746442665345967
      values: -0.0019056766759604216
      values: 0.0038097528740763664
      values: -0.0009622120996937156
      values: 0.00010718405246734619
      values: -0.005922803189605474 5
      values: 0.002000147011131048
      values: -0.008501190692186356
      values: -0.0009515900164842606
      values: 0.005201519466936588
      values: 0.0009683672687970102
```

```
          values: 0.011858307756483555
          values: -0.009651820175349712
          values: -0.00012620344932656735
          values: 0.0027150725945830345
          values: 0.008640092797577381
          values: -0.0034109146799892187
          values: -0.004304901696741581
          values: -0.001988197909668088
          values: -0.0005872835754416883
          values: -0.0018489856738597155
          values: -8.321687346324325e-05
          values: -0.026840969920158386
          values: 0.014779385179281235
          values: 0.048220183700323105
          values: 0.04137240722775459
          values: 0.022646132856607437
          values: -0.006832472048699856
          values: -0.0203497763723135
          values: -0.009036961942911148
          values: 0.00045483605936169624
          values: 8.447864092886448e-05
          values: -0.011278853751718998
          values: 0.015339664183557034
          values: -0.009544269181787968
          values: 0.03507256507873535
          values: -0.022777453064918518
        }
      }
    }
```

In [63]:

```
def create_transformed_df(train_pca, df_scaled, n_top_components):
    ''' Return a dataframe of data points with component features.
        The dataframe should be indexed by State-County and contain compone
        :param train_pca: A list of pca training data, returned by a PCA mo
        :param counties_scaled: A dataframe of normalized, original feature
        :param n_top_components: An integer, the number of top components t
        :return: A dataframe, indexed by State-County, with n_top_component
    '''
    data = {}
    size = len(train_pca)
    for i in range(n_top_components):
        data['c_{}'.format(i+1)] = [train_pca[j].label['projection'].float3
    transformed_df = pd.DataFrame(data=data, index=df_scaled.index)
    return transformed_df
```

In [64]:
```
create_transformed_df(train_pca, streets_df_scaled, 5).head()
```

Out[64]:

street_city	c_1	c_2	c_3	c_4	c_5
100th Avenue, Queens, NY	-0.022777	0.035073	-0.009544	0.015340	-0.011279
100th Drive, Queens, NY	-0.107436	0.000347	0.003553	-0.008035	-0.003171
100th Road, Queens, NY	-0.113299	0.002167	0.002954	-0.008789	-0.003356
100th Street, Kings, NY	-0.085527	0.002889	0.005174	-0.008929	0.001665
100th Street, Queens, NY	0.136673	0.025241	0.008200	0.025324	0.013837

In [65]:
```
## Specify top n
top_n = 5

# call your function and create a new dataframe
streets_transformed = create_transformed_df(train_pca, streets_df_scaled, n

streets_transformed.head()
```

Out[65]:

street_city	c_1	c_2	c_3	c_4	c_5
100th Avenue, Queens, NY	-0.022777	0.035073	-0.009544	0.015340	-0.011279
100th Drive, Queens, NY	-0.107436	0.000347	0.003553	-0.008035	-0.003171
100th Road, Queens, NY	-0.113299	0.002167	0.002954	-0.008789	-0.003356
100th Street, Kings, NY	-0.085527	0.002889	0.005174	-0.008929	0.001665
100th Street, Queens, NY	0.136673	0.025241	0.008200	0.025324	0.013837

In [66]:
```
session.delete_endpoint(pca_predictor.endpoint)
```

K-means

In [ ]:
```
# XXX: Find optimal K
# https://aws.amazon.com/blogs/machine-learning/k-means-clustering-with-ama
```

In [67]:
```
import sagemaker
# Try different k
kmeans = sagemaker.KMeans(role=role, train_instance_count=1, train_instance
print(kmeans)
```

```
<sagemaker.amazon.kmeans.KMeans object at 0x7fe71a1a26d8>
```

In [68]:
```
streets_transformed_np = streets_transformed.values.astype('float32')
streets_rs = kmeans.record_set(streets_transformed_np)
```

In [69]:
```
kmeans.fit(streets_rs)
```

```
'get_image_uri' method will be deprecated in favor of 'ImageURIProvide
r' class in SageMaker Python SDK v2.
's3_input' class will be renamed to 'TrainingInput' in SageMaker Python
SDK v2.
'get_image_uri' method will be deprecated in favor of 'ImageURIProvide
r' class in SageMaker Python SDK v2.
```

In [96]:
```
from scipy.spatial.distance import cdist
```

In [104]:
```
streets_transformed_tmp = streets_transformed[['c_1', 'c_2', 'c_3', 'c_4',
```

In [106]:
```
def run_kmeans_fit(rs, k, role):
    bn = 'sagemaker-us-east-1-006275120779'
    output_path = 's3://{}/{}'.format(bn, k)
    kmeans = sagemaker.KMeans(
        role=role, train_instance_count=1, train_instance_type='ml.c4.xlarg
        k=k, output_path=output_path)
    kmeans.fit(rs)
    return '{}/{}/output/model.tar.gz'.format(k, kmeans.latest_training_job
```

In [107]:
```
run_kmeans_fit(streets_rs, 2, role)  # Trial run for k=2
```

```
'get_image_uri' method will be deprecated in favor of 'ImageURIProvide
r' class in SageMaker Python SDK v2.
's3_input' class will be renamed to 'TrainingInput' in SageMaker Python
SDK v2.
'get_image_uri' method will be deprecated in favor of 'ImageURIProvide
r' class in SageMaker Python SDK v2.
```

In [118]:
```
# WARNING: A very long calculation
model_keys = {}
for k in range(2, 11):
    model_keys[k] = run_kmeans_fit(streets_rs, k, role)
```

```
'get_image_uri' method will be deprecated in favor of 'ImageURIProvide
r' class in SageMaker Python SDK v2.
's3_input' class will be renamed to 'TrainingInput' in SageMaker Python
SDK v2.
'get_image_uri' method will be deprecated in favor of 'ImageURIProvide
r' class in SageMaker Python SDK v2.
```

In [116]:
```
kmeans.latest_training_job.name
```

Out[116]:
```
'kmeans-2020-07-14-01-27-33-712'
```

In [121]:
```
# model_keys = {
#     2: '2/kmeans-2020-07-14-22-53-24-572/output/model.tar.gz',
#     3: '3/kmeans-2020-07-14-22-57-38-266/output/model.tar.gz',
#     4: '4/kmeans-2020-07-14-23-01-50-715/output/model.tar.gz',
#     5: '5/kmeans-2020-07-14-23-06-33-412/output/model.tar.gz',
#     6: '6/kmeans-2020-07-14-23-10-45-976/output/model.tar.gz',
#     7: '7/kmeans-2020-07-14-23-14-28-068/output/model.tar.gz',
#     8: '8/kmeans-2020-07-14-23-18-10-316/output/model.tar.gz',
#     9: '9/kmeans-2020-07-14-23-22-22-728/output/model.tar.gz',
#     10: '10/kmeans-2020-07-14-23-26-35-359/output/model.tar.gz',
# }
```

In [122]:
```
model_keys
```

Out[122]:
```
{2: '2/kmeans-2020-07-14-22-53-24-572/output/model.tar.gz',
 3: '3/kmeans-2020-07-14-22-57-38-266/output/model.tar.gz',
 4: '4/kmeans-2020-07-14-23-01-50-715/output/model.tar.gz',
 5: '5/kmeans-2020-07-14-23-06-33-412/output/model.tar.gz',
 6: '6/kmeans-2020-07-14-23-10-45-976/output/model.tar.gz',
 7: '7/kmeans-2020-07-14-23-14-28-068/output/model.tar.gz',
 8: '8/kmeans-2020-07-14-23-18-10-316/output/model.tar.gz',
 9: '9/kmeans-2020-07-14-23-22-22-728/output/model.tar.gz',
 10: '10/kmeans-2020-07-14-23-26-35-359/output/model.tar.gz'}
```

In [103]:
```
kmeans_model = mx.ndarray.load('model_algo-1')
kmeans_numpy = kmeans_model[0].asnumpy()
distortion = sum(np.min(cdist(streets_transformed_tmp, kmeans_numpy, 'eucli
print(distortion)  # 0.05455268327216128
# print(kmeans_numpy)
# print(streets_transformed.shape)
```

```
0.05455268327216128
```

In [102]:
```
streets_transformed.head()
```

Out[102]:

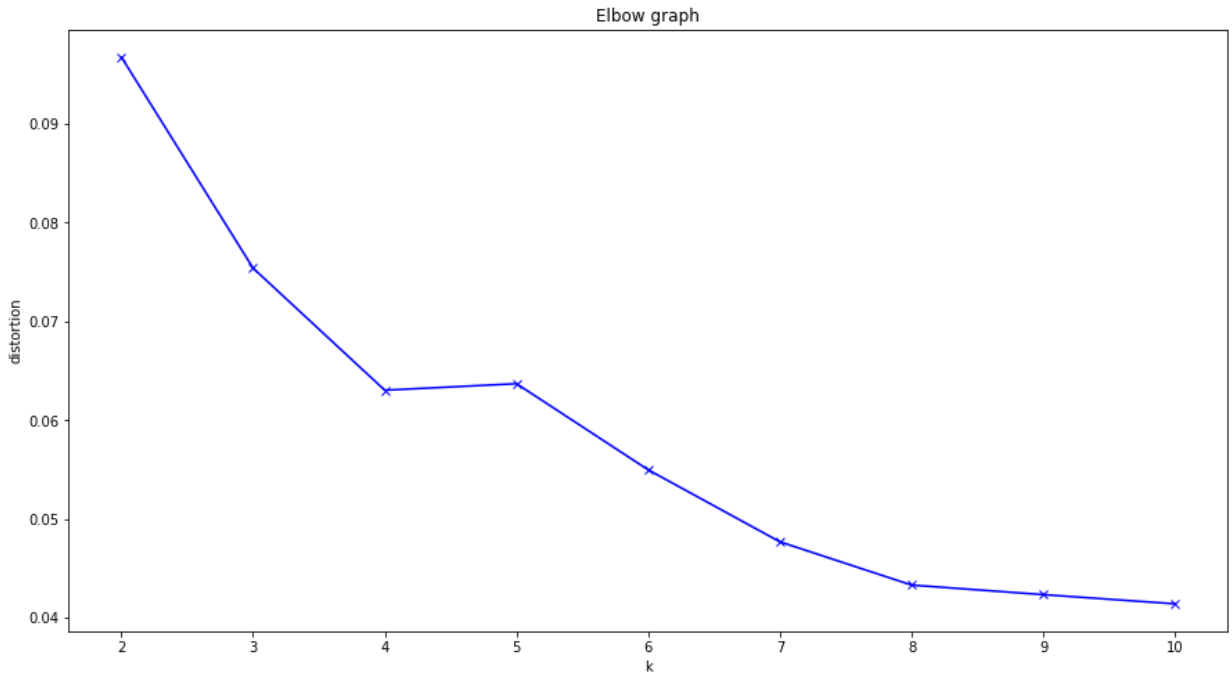
street_city	c_1	c_2	c_3	c_4	c_5	labels
100th Avenue, Queens, NY	-0.022777	0.035073	-0.009544	0.015340	-0.011279	4
100th Drive, Queens, NY	-0.107436	0.000347	0.003553	-0.008035	-0.003171	0
100th Road, Queens, NY	-0.113299	0.002167	0.002954	-0.008789	-0.003356	0
100th Street, Kings, NY	-0.085527	0.002889	0.005174	-0.008929	0.001665	0
100th Street, Queens, NY	0.136673	0.025241	0.008200	0.025324	0.013837	2

In [125]:
```
def get_distortion(bucket_name, model_key, df):
    # download and unzip model
    boto3.resource('s3').Bucket(bucket_name).download_file(model_key, 'mode
    # unzipping as model_algo-1
    os.system('tar -zxvf model.tar.gz')
    os.system('unzip model_algo-1')
    kmeans_model = mx.ndarray.load('model_algo-1')
    kmeans_numpy = kmeans_model[0].asnumpy()
    return sum(np.min(cdist(df, kmeans_numpy, 'euclidean'), axis=1)) / df.s
```

In [126]:
```
k_list = []
distortions = []
for k, model_key in model_keys.items():
    k_list.append(k)
    dist = get_distortion(bucket_name2, model_key, streets_transformed_tmp)
    distortions.append(dist)
print(distortions)
```

```
[0.09677154319398021, 0.07540288758219163, 0.06304325365721802, 0.0637031
1884653652, 0.054983099648486834, 0.04769420901940647, 0.0433169405314031
8, 0.042348865243549506, 0.041419052151975694]
```

In [164]:
```
# Plot the elbow
plt.figure(figsize=(15,8))
plt.plot(k_list, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('distortion')
plt.title('Elbow graph')
plt.show()
```



In [128]:
```
# Optimal K=4
```

In [129]:
```
kmeans = sagemaker.KMeans(role=role, train_instance_count=1, train_instance
```

In [131]:
```
kmeans.fit(streets_rs)
```

```
'get_image_uri' method will be deprecated in favor of 'ImageURIProvide
r' class in SageMaker Python SDK v2.
's3_input' class will be renamed to 'TrainingInput' in SageMaker Python
SDK v2.
'get_image_uri' method will be deprecated in favor of 'ImageURIProvide
r' class in SageMaker Python SDK v2.
```

In [132]:
```
%%time
# deploy the model to create a predictor
kmeans_predictor = kmeans.deploy(initial_instance_count=1,
                                 instance_type='ml.t2.medium')
```

```
Parameter image will be renamed to image_uri in SageMaker Python SDK v2.

------------------!CPU times: user 303 ms, sys: 18.1 ms, total: 321 ms
Wall time: 8min 32s
```

In [133]:
```
# get the predicted clusters for all the kmeans training data
cluster_info = kmeans_predictor.predict(streets_transformed_np)
```

In [134]:
```
# print cluster info for first data point
data_idx = 0

print('Street is: ', streets_transformed.index[data_idx])
print()
print(cluster_info[data_idx])
```

```
Street is:  100th Avenue, Queens, NY

label {
  key: "closest_cluster"
  value {
    float32_tensor {
      values: 1.0
    }
  }
}
label {
  key: "distance_to_cluster"
  value {
    float32_tensor {
      values: 0.0776713564991951
    }
  }
}
```

In [135]:
```
# get all cluster labels
cluster_labels = [c.label['closest_cluster'].float32_tensor.values[0] for c
```

In [136]:
```
# count up the points in each cluster
cluster_df = pd.DataFrame(cluster_labels)[0].value_counts()

print(cluster_df)
```

```
1.0    6407
3.0    1141
0.0     383
2.0      59
Name: 0, dtype: int64
```

In [137]:
```
# delete kmeans endpoint
session.delete_endpoint(kmeans_predictor.endpoint)
```

Model attributes

In [138]:
```
# download and unzip the kmeans model file
# use the name model_algo-1
training_job_name = kmeans.latest_training_job.name  # Example: 'kmeans-202

# where the model is saved, by default
model_key = os.path.join(training_job_name, 'output/model.tar.gz')
print(model_key)

# download and unzip model
boto3.resource('s3').Bucket(bucket_name2).download_file(model_key, 'model.t

# unzipping as model_algo-1
os.system('tar -zxvf model.tar.gz')
os.system('unzip model_algo-1')
```

```
kmeans-2020-07-15-00-05-20-176/output/model.tar.gz
```

Out[138]:
```
2304
```

In [139]:
```
# get the trained kmeans params using mxnet
# loading the unzipped artifacts
kmeans_model_params = mx.ndarray.load('model_algo-1')

print(kmeans_model_params)
```

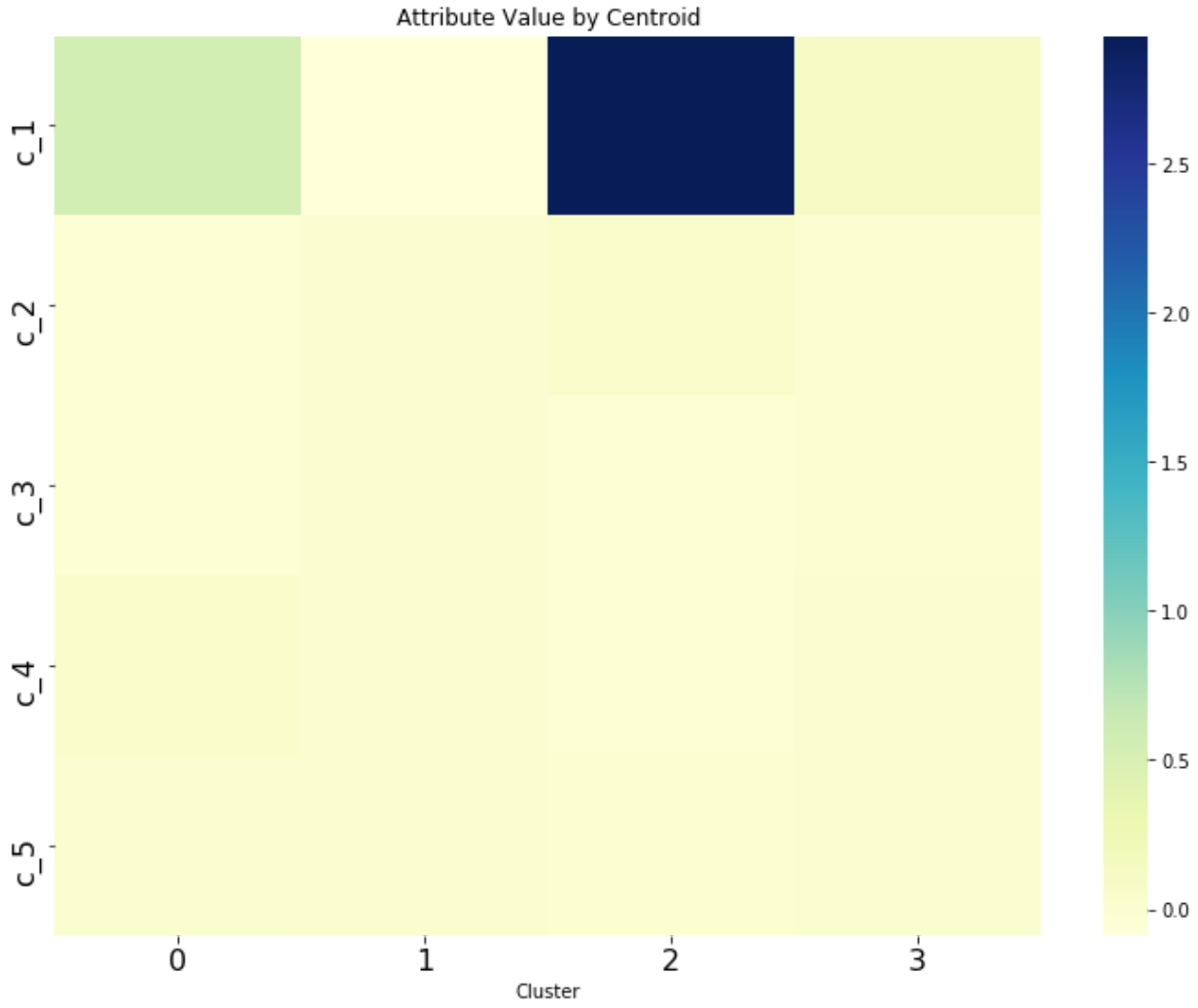
```
[
[[ 5.54219306e-01 -1.98705718e-02 -2.40580197e-02  4.46812361e-02
   1.00690164e-02]
 [-8.80557746e-02  2.08135997e-03  3.83516867e-03 -5.05146710e-03
  -1.87920011e-03]
 [ 2.92986131e+00  6.29404411e-02 -2.53199860e-02 -3.52037922e-02
  -1.05179232e-02]
 [ 1.12177595e-01 -7.91298971e-03 -1.29100708e-02  1.36005618e-02
   9.22227930e-03]]
<NDArray 4x5 @cpu(0)>]
```

In [141]:
```
# streets_transformed = streets_transformed_tmp
```

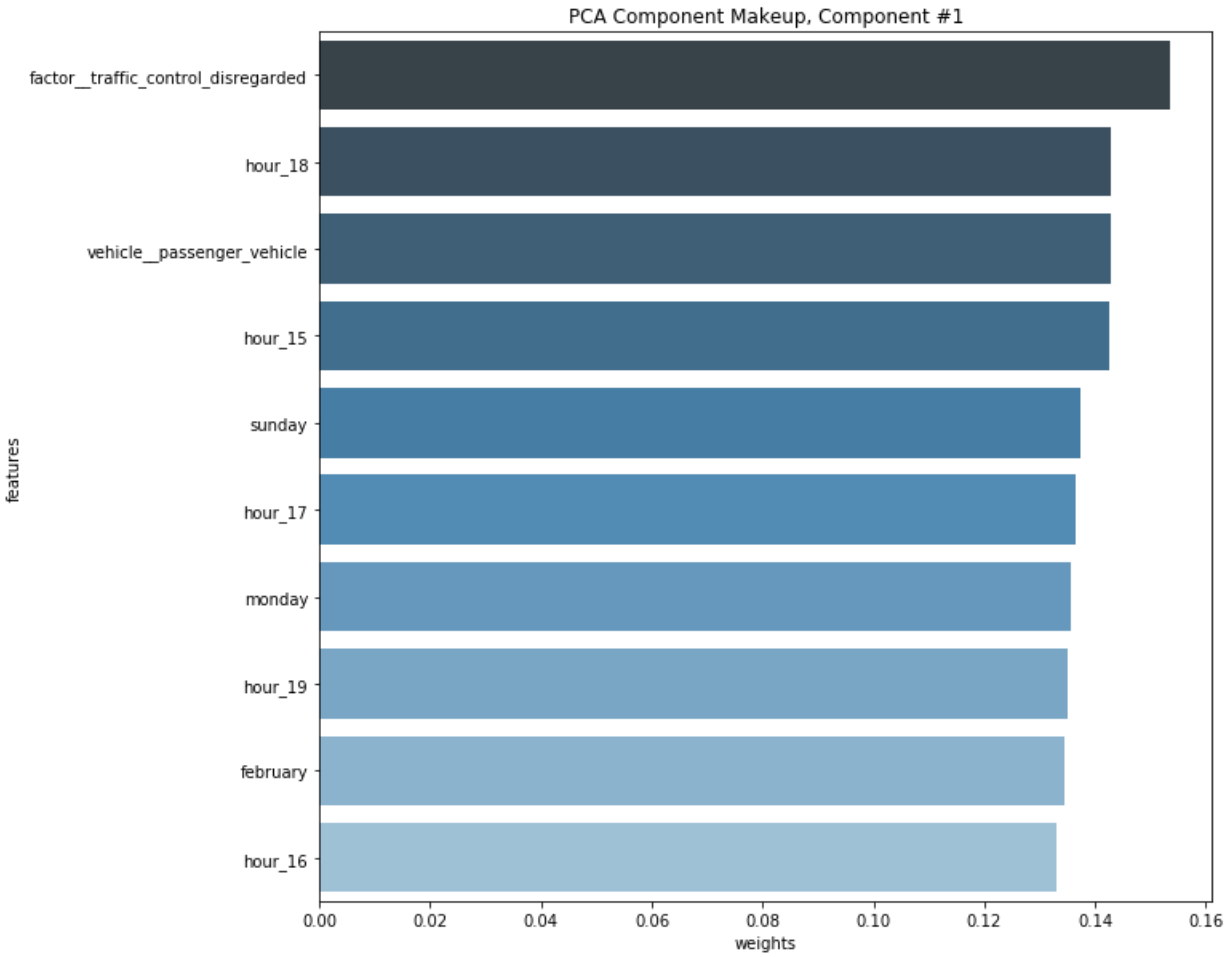
In [142]:
```
cluster_centroids=pd.DataFrame(kmeans_model_params[0].asnumpy())
cluster_centroids.columns=streets_transformed.columns

display(cluster_centroids)
```

	c_1	c_2	c_3	c_4	c_5
0	0.554219	-0.019871	-0.024058	0.044681	0.010069
1	-0.088056	0.002081	0.003835	-0.005051	-0.001879
2	2.929861	0.062940	-0.025320	-0.035204	-0.010518
3	0.112178	-0.007913	-0.012910	0.013601	0.009222

In [143]:
```
# generate a heatmap in component space, using the seaborn library
plt.figure(figsize = (12,9))
ax = sns.heatmap(cluster_centroids.T, cmap = 'YlGnBu')
ax.set_xlabel("Cluster")
plt.yticks(fontsize = 16)
plt.xticks(fontsize = 16)
ax.set_title("Attribute Value by Centroid")
plt.show()
```



In [144]:
```
component_num = 1
display_component(v, streets_df_scaled.columns.values, component_num=compon
```



In [145]:
```
# add a 'labels' column to the dataframe
streets_transformed['labels']=list(map(int, cluster_labels))

# sort by cluster label 0-6
sorted_streets = streets_transformed.sort_values('labels', ascending=True)
# view some pts in cluster 0
sorted_streets.head(20)
```

Out[145]:

street_city	c_1	c_2	c_3	c_4	c_5	labels
Carroll Street, Kings, NY	0.343678	-0.044849	0.030327	0.056601	-0.006047	0
Booth Memorial Avenue, Queens, NY	0.334320	0.001728	-0.008118	0.138228	0.056438	0
90th Avenue, Queens, NY	0.428128	0.019989	0.010274	0.117358	0.104571	0
225th Street, Queens, NY	0.346966	0.010751	-0.314830	-0.024157	0.033016	0
Borden Avenue, Queens, NY	0.541048	-0.145757	-0.916450	-0.245221	0.076035	0
Columbus Avenue, New York, NY	1.511552	-0.238995	0.183134	-0.015152	-0.059883	0
90th Street, Queens, NY	0.379852	-0.060337	-0.304423	0.014946	-0.045046	0
Flatlands Avenue, Kings, NY	0.428806	0.059037	0.018908	0.028268	-0.015706	0
91st Avenue, Queens, NY	0.535854	0.054913	0.011904	0.099323	0.056926	0
21st Street, Queens, NY	1.064310	-0.034216	-0.258374	-0.011698	-0.003880	0
Remsen Avenue, Kings, NY	1.024858	0.192849	-0.010237	0.140017	-0.139479	0
East 49th Street, New York, NY	0.357876	-0.044311	0.062823	-0.069488	-0.041282	0
West 207th Street, New York, NY	0.636619	0.014022	0.057855	-0.022418	0.113284	0
Flatbush Avenue Extension, Kings, NY:Kings, NY; New York, NY	0.352323	0.002624	0.037672	-0.024295	0.037822	0
92nd Street, Queens, NY	0.356895	-0.004100	0.021971	0.059462	-0.015004	0
Flushing Avenue, Queens, NY; Kings, NY	0.767804	-0.044015	0.039552	0.084707	-0.050363	0
Flushing Avenue, Queens, NY;Kings, NY	1.099580	-0.215500	-0.227173	-0.019764	0.075009	0
Bowery, New York, NY	1.309652	-0.185380	-0.141420	-0.266281	0.000464	0
East 42nd Street, New York, NY	0.824616	-0.143191	0.149571	-0.164777	-0.022014	0
College Point Boulevard, Queens, NY	1.290465	-0.176287	0.047598	0.396193	0.395847	0

In [88]:
```
# get all streets with label
label = 1
cluster = streets_transformed[streets_transformed['labels'] == label]
cluster.head(20)
```

Out[88]:

street_city	c_1	c_2	c_3	c_4	c_5	labels
10th Avenue, New York, NY	3.698854	-0.552950	0.150901	-0.410364	0.779525	1
11th Avenue, New York, NY	2.572565	-0.427033	-0.033742	-0.123142	0.590916	1
1st Avenue, New York, NY	5.012228	-0.831180	-0.067207	-0.367988	-0.491100	1
2nd Avenue, New York, NY	7.583614	-0.750211	0.970788	-0.866848	-0.323405	1
31st Street, Queens, NY	1.209643	0.075689	0.048117	0.100720	0.234549	1
34th Avenue, Queens, NY	1.416093	-0.037835	0.048293	0.242450	-0.062944	1
37th Avenue, Queens, NY	1.791203	-0.177043	-0.232613	0.270588	0.026712	1
3rd Avenue, Kings, NY	1.377077	-0.009382	-0.301234	0.144404	-0.004516	1
3rd Avenue, New York, NY	6.210366	-0.578836	0.718754	-0.437398	-0.441828	1
41st Avenue, Queens, NY	1.439340	-0.210372	-0.242559	0.220564	0.109863	1
4th Avenue, Kings, NY	1.674099	-0.113875	-0.232745	0.140385	0.017770	1
5th Avenue, Kings, NY	2.061547	-0.363421	0.129129	0.344241	-0.112422	1
5th Avenue, New York, NY	4.738774	-0.878272	0.313918	-0.377596	-0.283302	1
65th Street, Kings, NY	1.348302	0.076482	0.022003	0.270637	0.005871	1
6th Avenue, New York, NY	4.528898	-0.697028	0.640128	-0.560177	-0.190708	1
7th Avenue, Kings, NY	1.445145	-0.082877	0.081732	0.231874	0.046923	1
7th Avenue, New York, NY	3.565042	-0.742301	0.251810	-0.586870	0.063815	1
86th Street, Kings, NY	1.579399	-0.032842	0.044910	0.347432	-0.139526	1
8th Avenue, New York, NY	3.895898	-0.838191	-0.026639	-0.689205	0.176713	1
9th Avenue, New York, NY	3.622095	-0.465043	0.495452	-0.420841	0.457436	1

In [89]:
```
# get all streets with label
label = 2
cluster = streets_transformed[streets_transformed['labels'] == label]
cluster.head(20)
```

Out[89]:

street_city	c_1	c_2	c_3	c_4	c_5	labels
100th Street, Queens, NY	0.136673	0.025241	0.008200	0.025324	0.013837	2
101st Street, Queens, NY	0.121993	-0.005713	0.007017	0.045564	-0.029129	2
103-24 Roosevelt Avenue, Queens, NY	0.120114	-0.019387	0.006643	0.030362	-0.111885	2
103rd Street, Queens, NY	0.192047	-0.022458	0.024841	0.017981	0.018449	2
104th Avenue, Queens, NY	0.205218	0.063101	-0.009013	0.063680	0.042188	2
104th Street, Queens, NY	0.343599	-0.009553	0.012026	0.100842	-0.052530	2
10th Avenue, Kings, NY	0.215863	0.020450	0.005181	0.061946	0.045345	2
110th Street, Queens, NY	0.249414	-0.009147	0.008281	0.088973	0.027141	2
113th Street, Queens, NY	0.149895	0.009809	-0.000570	0.078707	0.004080	2
115th Street, Queens, NY	0.098821	-0.012935	0.010322	0.040047	0.046014	2
116th Avenue, Queens, NY	0.307560	0.050899	0.008503	0.060094	0.026771	2
118th Avenue, Queens, NY	0.172262	0.039406	0.005934	0.031358	0.022354	2
119th Avenue, Queens, NY	0.130095	0.072943	-0.014372	0.039745	-0.012283	2
11th Street, Queens, NY	0.273600	-0.061220	-0.298006	-0.053123	-0.014442	2
120th Avenue, Queens, NY	0.222067	-0.016874	-0.311235	-0.043562	0.025484	2
120th Street, Queens, NY	0.132980	-0.000827	0.004279	0.063053	0.016604	2
125th Street, Queens, NY	0.106288	-0.006714	0.012159	0.037057	0.049230	2
126th Street, Queens, NY	0.111360	-0.006844	0.009437	0.044798	0.058009	2
127th Street, Queens, NY	0.194626	-0.013164	0.016316	0.053522	0.074926	2
129th Street, Queens, NY	0.091662	0.010974	-0.007076	0.069706	-0.048920	2

In [90]:
```
# get all streets with label
label = 3
cluster = streets_transformed[streets_transformed['labels'] == label]
cluster.head(20)
```

Out[90]:

street_city	c_1	c_2	c_3	c_4	c_5	labels
101st Avenue, Queens, NY; Kings, NY	0.764141	0.034507	0.035077	0.119753	0.078745	3
102nd Street, Queens, NY	0.500105	-0.032335	0.028026	0.104668	0.028145	3
103rd Avenue, Queens, NY	0.620032	0.050023	0.006279	0.161315	0.082366	3
107th Avenue, Queens, NY	0.741983	0.072170	-0.006037	0.185652	0.128805	3
108th Street, Queens, NY	1.048073	-0.013278	0.024073	0.229510	0.005503	3
109th Avenue, Queens, NY	0.605643	0.075610	-0.005460	0.152991	0.097241	3
111th Avenue, Queens, NY	0.495848	0.076588	-0.005653	0.117896	0.034217	3
111th Street, Queens, NY	0.517596	0.035930	0.010052	0.108786	0.004425	3
112th Street, Queens, NY	0.377664	0.017149	0.005958	0.108614	0.059717	3
114th Street, Queens, NY	0.351033	0.022536	0.029139	0.014658	0.048354	3
115th Avenue, Queens, NY	0.357207	0.093036	-0.007887	0.064808	-0.027722	3
11th Avenue, Kings, NY	0.414364	0.003184	0.003968	0.131237	0.006213	3
13th Avenue, Kings, NY	0.720038	-0.043260	0.035434	0.156607	-0.079395	3
147th Street, Queens, NY	0.553307	-0.008873	-0.003458	0.217318	0.125007	3
149th Street, Queens, NY	0.429961	-0.004927	-0.013458	0.225307	0.130227	3
14th Avenue, Kings, NY	0.688990	-0.015283	0.022946	0.151387	-0.019386	3
14th Avenue, Queens, NY	0.372240	-0.053439	0.008110	0.172000	0.135843	3
150th Street, Queens, NY	1.056655	0.032689	0.009824	0.293781	0.219395	3
160th Street, Queens, NY	0.493885	0.017130	-0.002409	0.183025	0.099583	3
162nd Street, Queens, NY	0.374994	-0.040483	0.004987	0.173019	0.092189	3

In [91]:
```
# get all streets with label
label = 4
cluster = streets_transformed[streets_transformed['labels'] == label]
cluster.head(20)
```

Out[91]:

street_city	c_1	c_2	c_3	c_4	c_5	labels
100th Avenue, Queens, NY	-0.022777	0.035073	-0.009544	0.015340	-0.011279	4
102nd Avenue, Queens, NY	-0.038127	0.009522	0.004315	0.002515	0.006400	4
105th Avenue, Queens, NY	0.074769	0.011226	0.006450	0.022493	0.042930	4
105th Street, Queens, NY	0.082791	0.015629	0.003623	0.029666	-0.033543	4
106th Avenue, Queens, NY	-0.023414	0.014212	0.000953	0.015848	0.019457	4
106th Street, Queens, NY	0.058492	0.003625	0.006585	0.026897	0.027492	4
107th Street, Queens, NY	0.074881	0.010327	0.007149	0.027829	0.026929	4
108th Avenue, Queens, NY	0.054486	0.011167	-0.000261	0.039151	0.049477	4
109th Street, Queens, NY	0.033672	0.008483	0.004915	0.017076	0.010573	4
10th Street, Kings, NY	-0.018750	-0.015476	0.010299	0.004291	0.011846	4
10th Street, Queens, NY	0.058059	0.002571	0.002747	0.034506	-0.023278	4
110th Avenue, Queens, NY	-0.039107	0.012483	0.006315	-0.006075	0.000998	4
112th Avenue, Queens, NY	0.010272	0.023492	0.006560	-0.001216	0.005095	4
113th Avenue, Queens, NY	0.033842	0.023487	0.002403	0.023167	0.013423	4
114th Road, Queens, NY	-0.036776	0.009005	0.005797	-0.004573	0.003961	4
115th Road, Queens, NY	-0.034985	0.022805	-0.002682	0.007901	-0.004115	4
118th Street, Queens, NY	-0.018377	0.000519	0.006017	0.013752	0.016243	4
11th Avenue, Queens, NY	0.027955	-0.005864	-0.000641	0.051215	0.061492	4
11th Street, Kings, NY	-0.017274	-0.006563	0.009041	0.003755	0.012549	4
121st Avenue, Queens, NY	-0.031048	0.008494	0.006439	-0.002891	0.003333	4

In [119]:
```
# XXX: Save streets_transformed as CSV file
# streets_transformed.to_csv('s3://sagemaker-us-east-1-006275120779/cluster
```

K = 4

In [147]:
```
# get all streets with label
label = 0
cluster = streets_transformed[streets_transformed['labels'] == label]
cluster.head(20)
```

Out[147]:

street_city	c_1	c_2	c_3	c_4	c_5	labels
101st Avenue, Queens, NY; Kings, NY	0.764141	0.034507	0.035077	0.119753	0.078745	0
102nd Street, Queens, NY	0.500105	-0.032335	0.028026	0.104668	0.028145	0
103rd Avenue, Queens, NY	0.620032	0.050023	0.006279	0.161315	0.082366	0
104th Street, Queens, NY	0.343599	-0.009553	0.012026	0.100842	-0.052530	0
107th Avenue, Queens, NY	0.741983	0.072170	-0.006037	0.185652	0.128805	0
108th Street, Queens, NY	1.048073	-0.013278	0.024073	0.229510	0.005503	0
109th Avenue, Queens, NY	0.605643	0.075610	-0.005460	0.152991	0.097241	0
111th Avenue, Queens, NY	0.495848	0.076588	-0.005653	0.117896	0.034217	0
111th Street, Queens, NY	0.517596	0.035930	0.010052	0.108786	0.004425	0
112th Street, Queens, NY	0.377664	0.017149	0.005958	0.108614	0.059717	0
114th Street, Queens, NY	0.351033	0.022536	0.029139	0.014658	0.048354	0
115th Avenue, Queens, NY	0.357207	0.093036	-0.007887	0.064808	-0.027722	0
11th Avenue, Kings, NY	0.414364	0.003184	0.003968	0.131237	0.006213	0
13th Avenue, Kings, NY	0.720038	-0.043260	0.035434	0.156607	-0.079395	0
147th Street, Queens, NY	0.553307	-0.008873	-0.003458	0.217318	0.125007	0
149th Street, Queens, NY	0.429961	-0.004927	-0.013458	0.225307	0.130227	0
14th Avenue, Kings, NY	0.688990	-0.015283	0.022946	0.151387	-0.019386	0
14th Avenue, Queens, NY	0.372240	-0.053439	0.008110	0.172000	0.135843	0
150th Street, Queens, NY	1.056655	0.032689	0.009824	0.293781	0.219395	0
160th Street, Queens, NY	0.493885	0.017130	-0.002409	0.183025	0.099583	0

In [146]:
```
# get all streets with label
label = 1
cluster = streets_transformed[streets_transformed['labels'] == label]
cluster.head(20)
```

Out[146]:

street_city	c_1	c_2	c_3	c_4	c_5	labels
100th Avenue, Queens, NY	-0.022777	0.035073	-0.009544	0.015340	-0.011279	1
100th Drive, Queens, NY	-0.107436	0.000347	0.003553	-0.008035	-0.003171	1
100th Road, Queens, NY	-0.113299	0.002167	0.002954	-0.008789	-0.003356	1
100th Street, Kings, NY	-0.085527	0.002889	0.005174	-0.008929	0.001665	1
100th Street, Queens, NY"	-0.108015	0.001801	0.003135	-0.007808	-0.003363	1
101st Avenue, Queens, NY	-0.085332	0.009004	0.001116	0.000842	0.001956	1
101st Road, Queens, NY	-0.112439	0.001839	0.003215	-0.009062	-0.003436	1
102nd Avenue, Queens, NY	-0.038127	0.009522	0.004315	0.002515	0.006400	1
102nd Road, Queens, NY	-0.086397	0.003194	0.002695	-0.002384	0.001831	1
103rd Drive, Queens, NY	-0.113254	0.002054	0.003019	-0.008877	-0.003378	1
103rd Road, Queens, NY	-0.099085	0.004015	0.001925	-0.002354	0.000643	1
104th Road, Queens, NY	-0.105193	-0.002420	0.004479	-0.008243	-0.002431	1
106th Avenue, Queens, NY	-0.023414	0.014212	0.000953	0.015848	0.019457	1
106th Road, Queens, NY	-0.107760	0.001938	0.003228	-0.007811	-0.001065	1
107th Road, Queens, NY	-0.110355	0.002588	0.003210	-0.009065	-0.002994	1
108th Drive, Queens, NY	-0.107523	0.001955	0.003277	-0.007673	-0.001850	1
108th Road, Queens, NY	-0.106820	0.002200	0.002757	-0.006270	-0.002141	1
109th Drive, Queens, NY	-0.108211	0.002544	0.002737	-0.006799	-0.002752	1
109th Road, Queens, NY	-0.075171	0.008675	0.002156	0.000709	0.004463	1
10th Avenue, Queens, NY	-0.067349	-0.004768	0.003487	0.009364	0.014756	1

In [148]:
```
# get all streets with label
label = 2
cluster = streets_transformed[streets_transformed['labels'] == label]
cluster.head(20)
```

Out[148]:

street_city	c_1	c_2	c_3	c_4	c_5	labels
10th Avenue, New York, NY	3.698854	-0.552950	0.150901	-0.410364	0.779525	2
11th Avenue, New York, NY	2.572565	-0.427033	-0.033742	-0.123142	0.590916	2
1st Avenue, New York, NY	5.012228	-0.831180	-0.067207	-0.367988	-0.491100	2
2nd Avenue, New York, NY	7.583614	-0.750211	0.970788	-0.866848	-0.323405	2
37th Avenue, Queens, NY	1.791203	-0.177043	-0.232613	0.270588	0.026712	2
3rd Avenue, New York, NY	6.210366	-0.578836	0.718754	-0.437398	-0.441828	2
5th Avenue, Kings, NY	2.061547	-0.363421	0.129129	0.344241	-0.112422	2
5th Avenue, New York, NY	4.738774	-0.878272	0.313918	-0.377596	-0.283302	2
6th Avenue, New York, NY	4.528898	-0.697028	0.640128	-0.560177	-0.190708	2
7th Avenue, New York, NY	3.565042	-0.742301	0.251810	-0.586870	0.063815	2
8th Avenue, New York, NY	3.895898	-0.838191	-0.026639	-0.689205	0.176713	2
9th Avenue, New York, NY	3.622095	-0.465043	0.495452	-0.420841	0.457436	2
Amsterdam Avenue, New York, NY	3.423664	-0.447013	0.015209	-0.008571	0.041159	2
Atlantic Avenue, Kings, NY	4.444749	0.342087	-0.103294	-0.087429	0.087625	2
Bedford Avenue, Kings, NY	3.436114	-0.097656	0.144606	0.375397	-0.055749	2
Belt Parkway, Kings, NY	2.368025	0.807087	0.000476	-0.093474	0.058969	2
Belt Parkway, Queens, NY; Kings, NY	3.391768	1.358405	-0.036260	-0.247719	-0.294292	2
Boston Road, Bronx, NY	2.046237	0.149924	0.062450	0.218815	-0.033566	2
Broadway, Kings, NY	1.942008	-0.439840	-0.860743	0.008842	-0.048871	2
Broadway, New York, NY	6.479069	-1.025939	0.377224	-0.055325	-0.039014	2

In [149]:
```
# get all streets with label
label = 3
cluster = streets_transformed[streets_transformed['labels'] == label]
cluster.head(20)
```

Out[149]:

street_city	c_1	c_2	c_3	c_4	c_5	labels
100th Street, Queens, NY	0.136673	0.025241	0.008200	0.025324	0.013837	3
101st Street, Queens, NY	0.121993	-0.005713	0.007017	0.045564	-0.029129	3
103-24 Roosevelt Avenue, Queens, NY	0.120114	-0.019387	0.006643	0.030362	-0.111885	3
103rd Street, Queens, NY	0.192047	-0.022458	0.024841	0.017981	0.018449	3
104th Avenue, Queens, NY	0.205218	0.063101	-0.009013	0.063680	0.042188	3
105th Avenue, Queens, NY	0.074769	0.011226	0.006450	0.022493	0.042930	3
105th Street, Queens, NY	0.082791	0.015629	0.003623	0.029666	-0.033543	3
106th Street, Queens, NY	0.058492	0.003625	0.006585	0.026897	0.027492	3
107th Street, Queens, NY	0.074881	0.010327	0.007149	0.027829	0.026929	3
108th Avenue, Queens, NY	0.054486	0.011167	-0.000261	0.039151	0.049477	3
109th Street, Queens, NY	0.033672	0.008483	0.004915	0.017076	0.010573	3
10th Avenue, Kings, NY	0.215863	0.020450	0.005181	0.061946	0.045345	3
10th Street, Queens, NY	0.058059	0.002571	0.002747	0.034506	-0.023278	3
110th Street, Queens, NY	0.249414	-0.009147	0.008281	0.088973	0.027141	3
113th Avenue, Queens, NY	0.033842	0.023487	0.002403	0.023167	0.013423	3
113th Street, Queens, NY	0.149895	0.009809	-0.000570	0.078707	0.004080	3
115th Street, Queens, NY	0.098821	-0.012935	0.010322	0.040047	0.046014	3
116th Avenue, Queens, NY	0.307560	0.050899	0.008503	0.060094	0.026771	3
118th Avenue, Queens, NY	0.172262	0.039406	0.005934	0.031358	0.022354	3
119th Avenue, Queens, NY	0.130095	0.072943	-0.014372	0.039745	-0.012283	3

In [159]:
```
clustered_streets = streets_transformed.copy()
```

In [160]:
```
clustered_streets.insert(0, 'street_city', clustered_streets.index)
clustered_streets.head()
```

Out[160]:

street_city	street_city	c_1	c_2	c_3	c_4	c_5	labels
100th Avenue, Queens, NY	100th Avenue, Queens, NY	-0.022777	0.035073	-0.009544	0.015340	-0.011279	1
100th Drive, Queens, NY	100th Drive, Queens, NY	-0.107436	0.000347	0.003553	-0.008035	-0.003171	1
100th Road, Queens, NY	100th Road, Queens, NY	-0.113299	0.002167	0.002954	-0.008789	-0.003356	1
100th Street, Kings, NY	100th Street, Kings, NY	-0.085527	0.002889	0.005174	-0.008929	0.001665	1
100th Street, Queens, NY	100th Street, Queens, NY	0.136673	0.025241	0.008200	0.025324	0.013837	3

In [161]:
```
clustered_streets.to_csv('s3://sagemaker-us-east-1-006275120779/clustered_s
```

In [ ]: