South_Africa_Crime_Statistics_Python_Notebook (1)

December 18, 2024

CRIME STATISTICS IN SOUTH AFRICA

Problem Statement

The World Bank under its Governance Global Practice supports its client countries to help them build capable, efficient, open, inclusive, and accountable institutions. Our company, Mckinsey Consultants, has been given the task to work with the South African Police Services(SAPS) to ensure that its crime statistics are in line with the international best practice. This will be achieved through a Memorandum of Understanding with Statistics South Africa (Stats SA), aimed at further enhancing the quality and integrity of the South African Crime Statistics.

Business Understanding

Crimes in South Africa has been on the rise especially in recent years. South African Police Services however are required to try and curb the crimes according to the international crime practice. The objectives of this analysis are: * Identify the year that had the highest rate of crime * Examine the province that has the highest rate of crime * Determine the time was crime prevalent and in which police station, province overall * Investigate the police station which recorded the highest number of crimes * Understand the correlation between the number of reported crimes and the population size. * Identify which crimes have experienced a steady decline over the years.

Data Understanding

[3]:
```
# Importing Pandas library
import pandas as pd

# Importing Numpy library
import numpy as np

#Import plotting library
import matplotlib.pyplot as plt
import seaborn as sns
```

[4]:
```
#loading the csv file on python by first assigning it to a variable
prov_pop = 'ProvincePopulation.csv'

prov_pop1 = pd.read_csv(prov_pop)

#viewing the table
prov_pop1.head(10)
```

[4]:

	Province	Population	Area	Density
0	Gauteng	12272263	18178	675.1
1	Kwazulu/Natal	10267300	94361	108.8
2	Mpumalanga	4039939	76495	52.8
3	Western Cape	5822734	129462	45.0
4	Limpopo	5404868	125755	43.0
5	Eastern Cape	6562053	168966	38.8
6	North West	3509953	104882	33.5
7	Free State	2745590	129825	21.1
8	Northern Cape	1145861	372889	3.1

[5]:
```
#loading the crime data csv file on python by first assigning it to a variable
sa_crime_rate = 'South Africa crime data.csv'

df = pd.read_csv(sa_crime_rate)

#viewing the table
df.head(20)
```

[5]:

	Province	Station	Category	2005-2006	2006-2007	2007-2008	2008-2009	2009-2010	2010-2011
0	Western Cape	Cape Town Central	All theft not mentioned elsewhere	6692	6341	5966	5187	4985	5127
1	Gauteng	Jhb Central	All theft not mentioned elsewhere	6093	4602	3761	3610	3267	3037
2	Western Cape	Mitchells Plain	All theft not mentioned elsewhere	5341	6093	6316	6803	6035	5761
3	Free State	Park Road	All theft not mentioned elsewhere	5108	4282	3834	3316	3101	3013
4	Gauteng	Pretoria Central	All theft not mentioned elsewhere	5099	4536	3309	2694	2616	2606
5	North West	Rustenburg	All theft not mentioned elsewhere						
6	Kwazulu/Natal	Durban Central	All theft not mentioned elsewhere						
7	Gauteng	Brooklyn	All theft not mentioned elsewhere						
8	Gauteng	Booysens	All theft not mentioned elsewhere						
9	Gauteng	Hillbrow	All theft not mentioned elsewhere						
10	Mpumalanga	Nelspruit	All theft not mentioned elsewhere						
11	Western Cape	Cape Town Central	Theft out of or from motor vehicle						
12	Western Cape	Mitchells Plain	Drug-related crime						
13	Gauteng	Sandton	All theft not mentioned elsewhere						
14	Western Cape	Bellville	All theft not mentioned elsewhere						
15	Gauteng	Sunnyside	All theft not mentioned elsewhere						
16	Eastern Cape	East London	All theft not mentioned elsewhere						
17	Kwazulu/Natal	Durban Central	Robbery with aggravating circumstances						
18	Western Cape	Mitchells Plain	Common assault						
19	Eastern Cape	Humewood	All theft not mentioned elsewhere						

```
5        4239       4173       3398       3388       2737       2117
6        4162       4529       3499       3353       3183       2933
7        3931       3583       2878       2568       2415       2162
8        3681       3277       2849       2603       2580       3107
9        3489       2914       3093       2706       2250       2298
10       3481       2385       1812       1558       1437       1562
11       3468       2924       2329       1856       2905       3051
12       3064       3683       4792       5699       6571       6260
13       3030       3037       2556       2383       2431       2601
14       3010       2828       2721       2911       2691       2180
15       2967       2659       1866       1881       2044       2168
16       2892       2432       2058       1892       1582       1567
17       2721       3214       2134       1966       1371        899
18       2657       2339       2131       2735       2749       2444
19       2585       2276       1889       1712       1288       1569

    2011-2012  2012-2013  2013-2014  2014-2015  2015-2016
0        5285       5937       5600       5335       5176
1        2886       2638       2809       3050       2434
2        6108       5514       4975       4043       3635
3        2679       3116       2927       2297       2103
4        2635       3226       3246       2892       3030
5        2139       1914       1897       1868       1862
6        3219       3418       3390       2872       2865
7        2050       1883       2442       2200       2107
8        2568       1339       1290       1039        943
9        2051       1835       1610       1607       1618
10       1450       1390       1307       1309       1152
11       3474       3294       3612       3441       3509
12       5850       6310       6044       4768       4609
13       2446       2310       2275       2838       2866
14       2435       2231       2249       2136       1839
15       1862       1861       2350       1985       1820
16       1498       1251       1618       1394       1268
17        924        885        951        982       1024
18       2810       2757       2185       1847       2079
19       1294       1236       1408       1132       1044
```

[6]:
```
# General information about the crime data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30861 entries, 0 to 30860
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Province   30861 non-null  object
```

```
 1   Station    30861 non-null  object
 2   Category   30861 non-null  object
 3   2005-2006  30861 non-null  int64
 4   2006-2007  30861 non-null  int64
 5   2007-2008  30861 non-null  int64
 6   2008-2009  30861 non-null  int64
 7   2009-2010  30861 non-null  int64
 8   2010-2011  30861 non-null  int64
 9   2011-2012  30861 non-null  int64
 10  2012-2013  30861 non-null  int64
 11  2013-2014  30861 non-null  int64
 12  2014-2015  30861 non-null  int64
 13  2015-2016  30861 non-null  int64
dtypes: int64(11), object(3)
memory usage: 3.3+ MB
```

[8]:
```
pop= 'South Africa GDP and POP1.csv'

df1 = pd.read_csv(pop)

# Viewing data from the GDP and POP data
df1
```

[8]:

	Year	GDP Nominal (Current USD)	GDP Real (Inflation adj.)	GDP change	GDP per capita	\
0	NaN				NaN	
1	2017.0	$348,871,647,960	$426,813,227,524	1.32%	$7,487	
2	2016.0	$295,746,599,722	$421,266,226,143	0.57%	$7,495	
3	2015.0	$317,536,830,641	$418,898,007,438	1.28%	$7,563	
4	2014.0	$350,636,208,164	$413,605,718,439	1.85%	$7,583	
5	2013.0	$366,643,223,164	$406,104,993,311	2.49%	$7,564	
6	2012.0	$396,327,875,201	$396,257,207,214	2.21%	$7,500	
7	2011.0	$416,418,874,939	$387,676,549,661	3.28%	$7,455	
8	2010.0	$375,349,442,837	$375,349,442,837	3.04%	$7,329	
9	2009.0	$295,936,485,833	$364,276,420,244	-1.54%	$7,217	
10	2008.0	$286,769,839,733	$369,966,840,761	3.19%	$7,432	
11	2007.0	$299,415,505,152	$358,526,105,166	5.36%	$7,299	
12	2006.0	$271,638,484,826	$340,285,199,108	5.60%	$7,018	
13	2005.0	$257,671,413,751	$322,228,183,700	5.28%	$6,730	
14	2004.0	$228,937,347,866	$306,076,361,734	4.55%	$6,472	
15	2003.0	$175,256,916,996	$292,743,217,487	2.95%	$6,266	
16	2002.0	$115,748,110,113	$284,357,295,801	3.70%	$6,161	
17	2001.0	$121,600,818,310	$274,210,460,320	2.70%	$6,017	
18	2000.0	$136,361,854,808	$267,001,436,053	4.20%	$5,938	
19	1999.0	$136,631,966,609	$256,239,373,462	2.40%	$5,779	
20	1998.0	$137,774,361,015	$250,233,772,323	0.50%	$5,728	
21	1997.0	$152,586,154,514	$248,988,825,940	2.60%	$5,792	
22	1996.0	$147,607,982,695	$242,679,162,577	4.30%	$5,745	

```
23  1995.0  $155,460,285,076   $232,674,175,450    3.10%        $5,615
24  1994.0  $139,752,450,152   $225,678,162,745    3.20%        $5,564
```

```
      Pop.  Population
0   change         NaN
1    1.43%  57,009,756
2    1.48%  56,207,646
3    1.54%  55,386,367
4    1.60%  54,544,186
5    1.62%  53,687,121
6    1.59%  52,832,658
7    1.54%  52,003,755
8    1.47%  51,216,964
9    1.40%  50,477,011
10   1.34%  49,779,471
11   1.30%  49,119,759
12   1.27%  48,489,459
13   1.25%  47,880,601
14   1.23%  47,291,610
15   1.23%  46,719,196
16   1.27%  46,150,913
17   1.34%  45,571,274
18   1.42%  44,967,708
19   1.50%  44,338,543
20   1.62%  43,682,260
21   1.77%  42,987,461
22   1.94%  42,241,011
23   2.15%  41,435,758
24   2.35%  40,564,059
```

[9]:
```
# General information about the GDP and POP data
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Year            24 non-null     float64
 1   GDP Nominal     25 non-null     object
 2   GDP Real        25 non-null     object
 3   GDP             25 non-null     object
 4   GDP per capita  24 non-null     object
 5   Pop.            25 non-null     object
 6   Population      24 non-null     object
dtypes: float64(1), object(6)
memory usage: 1.5+ KB
```

[ ]:
```
# Picking data of interest from 2005 to 2015
df1 = df1.iloc[3:14].reset_index(drop=True)
df1
```

[ ]:

	Year	GDP Nominal	…	Pop.	Population
0	2015.0	$317,536,830,641	…	1.54%	55,386,367
1	2014.0	$350,636,208,164	…	1.60%	54,544,186
2	2013.0	$366,643,223,164	…	1.62%	53,687,121
3	2012.0	$396,327,875,201	…	1.59%	52,832,658
4	2011.0	$416,418,874,939	…	1.54%	52,003,755
5	2010.0	$375,349,442,837	…	1.47%	51,216,964
6	2009.0	$295,936,485,833	…	1.40%	50,477,011
7	2008.0	$286,769,839,733	…	1.34%	49,779,471
8	2007.0	$299,415,505,152	…	1.30%	49,119,759
9	2006.0	$271,638,484,826	…	1.27%	48,489,459
10	2005.0	$257,671,413,751	…	1.25%	47,880,601

[11 rows x 7 columns]

[10]:
```
# Adjusting the column names by replacing the whitespace with _ character for␣
 ↪convennience
df1.columns = df1.columns.str.upper().str.strip().str.replace(' ','_').str.
 ↪replace('.','')
df1.columns
```

[10]:
```
Index(['YEAR', 'GDP_NOMINAL', 'GDP_REAL', 'GDP', 'GDP_PER_CAPITA', 'POP',
       'POPULATION'],
      dtype='object')
```

[11]:
```
# Dropping the unrequired columns
df1=df1.drop(['GDP_REAL','GDP','GDP_PER_CAPITA', 'POP'], axis=1)
df1
```

[11]:

	YEAR	GDP_NOMINAL	POPULATION
0	NaN	(Current USD)	NaN
1	2017.0	$348,871,647,960	57,009,756
2	2016.0	$295,746,599,722	56,207,646
3	2015.0	$317,536,830,641	55,386,367
4	2014.0	$350,636,208,164	54,544,186
5	2013.0	$366,643,223,164	53,687,121
6	2012.0	$396,327,875,201	52,832,658
7	2011.0	$416,418,874,939	52,003,755
8	2010.0	$375,349,442,837	51,216,964
9	2009.0	$295,936,485,833	50,477,011
10	2008.0	$286,769,839,733	49,779,471
11	2007.0	$299,415,505,152	49,119,759
12	2006.0	$271,638,484,826	48,489,459

```
13  2005.0  $257,671,413,751   47,880,601
14  2004.0  $228,937,347,866   47,291,610
15  2003.0  $175,256,916,996   46,719,196
16  2002.0  $115,748,110,113   46,150,913
17  2001.0  $121,600,818,310   45,571,274
18  2000.0  $136,361,854,808   44,967,708
19  1999.0  $136,631,966,609   44,338,543
20  1998.0  $137,774,361,015   43,682,260
21  1997.0  $152,586,154,514   42,987,461
22  1996.0  $147,607,982,695   42,241,011
23  1995.0  $155,460,285,076   41,435,758
24  1994.0  $139,752,450,152   40,564,059
```

[18]:
```
# Viewing general information about the df1 GDP and POP data
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 24 to 0
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   YEAR         11 non-null     object
 1   GDP_NOMINAL  25 non-null     object
 2   POPULATION   24 non-null     object
dtypes: object(3)
memory usage: 732.0+ bytes
```

[19]:
```
# Cleaning the GDP and POP data by removing '$' and ','
df1['GDP_NOMINAL'] =df1.GDP_NOMINAL.str.replace('$','').str.replace(',', '').
 ↪astype(int)
df1['POPULATION'] = df1.POPULATION.str.replace(',', '').astype(int)

# Viewing general information about the cleaned data
df1.info()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[19], line 2
      1 # Cleaning the GDP and POP data by removing '$' and ','
----> 2 df1['GDP_NOMINAL'] =df1.GDP_NOMINAL.str.replace('$','').str.replace(',
 ↪'').astype(int)
      3 df1['POPULATION'] = df1.POPULATION.str.replace(',', '').astype(int)
      5 # Viewing general information about the cleaned data

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/generic.py:6324, in NDFrame.astype(self, dtype, copy, errors)
   6317     results = [
```

```
   6318         self.iloc[:, i].astype(dtype, copy=copy)
   6319         for i in range(len(self.columns))
   6320     ]
   6322 else:
   6323     # else, only a single dtype is given
-> 6324     new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
   6325     return self._constructor(new_data).__finalize__(self,␣
 ↪method="astype")
   6327 # GH 33113: handle empty frame or series

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/internals/managers.py:451, in BaseBlockManager.astype(self, dtype␣
 ↪copy, errors)
    448 elif using_copy_on_write():
    449     copy = False
--> 451 return self.apply(
    452     "astype",
    453     dtype=dtype,
    454     copy=copy,
    455     errors=errors,
    456     using_cow=using_copy_on_write(),
    457 )

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/internals/managers.py:352, in BaseBlockManager.apply(self, f,␣
 ↪align_keys, **kwargs)
    350         applied = b.apply(f, **kwargs)
    351     else:
--> 352         applied = getattr(b, f)(**kwargs)
    353     result_blocks = extend_blocks(applied, result_blocks)
    355 out = type(self).from_blocks(result_blocks, self.axes)

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/internals/blocks.py:511, in Block.astype(self, dtype, copy,␣
 ↪errors, using_cow)
    491 """
    492 Coerce to the new dtype.
    493
   (…)
    507 Block
    508 """
    509 values = self.values
--> 511 new_values = astype_array_safe(values, dtype, copy=copy, errors=errors)
    513 new_values = maybe_coerce_values(new_values)
    515 refs = None

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/dtypes/astype.py:242, in astype_array_safe(values, dtype, copy,␣
 ↪errors)
```

```
    239     dtype = dtype.numpy_dtype
    241 try:
--> 242     new_values = astype_array(values, dtype, copy=copy)
    243 except (ValueError, TypeError):
    244     # e.g. _astype_nansafe can fail on object-dtype of strings
    245     #  trying to convert to float
    246     if errors == "ignore":

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/dtypes/astype.py:187, in astype_array(values, dtype, copy)
    184     values = values.astype(dtype, copy=copy)
    186 else:
--> 187     values = _astype_nansafe(values, dtype, copy=copy)
    189 # in pandas we don't store numpy str dtypes, so convert to object
    190 if isinstance(dtype, np.dtype) and issubclass(values.dtype.type, str):

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪pandas/core/dtypes/astype.py:138, in _astype_nansafe(arr, dtype, copy, skipna
    134     raise ValueError(msg)
    136 if copy or is_object_dtype(arr.dtype) or is_object_dtype(dtype):
    137     # Explicit copy, or required since NumPy can't view from / to objec .
--> 138     return arr.astype(dtype, copy=True)
    140 return arr.astype(dtype, copy=copy)

ValueError: invalid literal for int() with base 10: '(Current USD)'
```

Data Preparation

[20]:
```
# Dropping irrelevant columns
prov_pop2 = prov_pop1.drop(['Area','Density'], axis =1)
prov_pop2
```

[20]:

	Province	Population
0	Gauteng	12272263
1	Kwazulu/Natal	10267300
2	Mpumalanga	4039939
3	Western Cape	5822734
4	Limpopo	5404868
5	Eastern Cape	6562053
6	North West	3509953
7	Free State	2745590
8	Northern Cape	1145861

[ ]:

[21]:
```
# Adjusting the YEAR column data
```

```
z=['2015-2016', '2014-2015', '2013-2014', '2012-2013', '2011-2012',␣
  ↪'2010-2011', '2009-2010', '2008-2009', '2007-2008', '2006-2007', '2005-2006']
df1.YEAR = pd.DataFrame(z)

# Viewing the adjusted data
df1
```

[21]:

	YEAR	GDP_NOMINAL	POPULATION
24	NaN	$139,752,450,152	40,564,059
23	NaN	$155,460,285,076	41,435,758
22	NaN	$147,607,982,695	42,241,011
21	NaN	$152,586,154,514	42,987,461
20	NaN	$137,774,361,015	43,682,260
19	NaN	$136,631,966,609	44,338,543
18	NaN	$136,361,854,808	44,967,708
17	NaN	$121,600,818,310	45,571,274
16	NaN	$115,748,110,113	46,150,913
15	NaN	$175,256,916,996	46,719,196
14	NaN	$228,937,347,866	47,291,610
13	NaN	$257,671,413,751	47,880,601
12	NaN	$271,638,484,826	48,489,459
11	NaN	$299,415,505,152	49,119,759
10	2005-2006	$286,769,839,733	49,779,471
9	2006-2007	$295,936,485,833	50,477,011
8	2007-2008	$375,349,442,837	51,216,964
7	2008-2009	$416,418,874,939	52,003,755
6	2009-2010	$396,327,875,201	52,832,658
5	2010-2011	$366,643,223,164	53,687,121
4	2011-2012	$350,636,208,164	54,544,186
3	2012-2013	$317,536,830,641	55,386,367
2	2013-2014	$295,746,599,722	56,207,646
1	2014-2015	$348,871,647,960	57,009,756
0	2015-2016	(Current USD)	NaN

[22]:
```
# Reordering the column in reverse order
df1=  df1[::-1]
df1
```

[22]:

	YEAR	GDP_NOMINAL	POPULATION
0	2015-2016	(Current USD)	NaN
1	2014-2015	$348,871,647,960	57,009,756
2	2013-2014	$295,746,599,722	56,207,646
3	2012-2013	$317,536,830,641	55,386,367
4	2011-2012	$350,636,208,164	54,544,186
5	2010-2011	$366,643,223,164	53,687,121
6	2009-2010	$396,327,875,201	52,832,658
7	2008-2009	$416,418,874,939	52,003,755

```
8   2007-2008  $375,349,442,837   51,216,964
9   2006-2007  $295,936,485,833   50,477,011
10  2005-2006  $286,769,839,733   49,779,471
11         NaN  $299,415,505,152   49,119,759
12         NaN  $271,638,484,826   48,489,459
13         NaN  $257,671,413,751   47,880,601
14         NaN  $228,937,347,866   47,291,610
15         NaN  $175,256,916,996   46,719,196
16         NaN  $115,748,110,113   46,150,913
17         NaN  $121,600,818,310   45,571,274
18         NaN  $136,361,854,808   44,967,708
19         NaN  $136,631,966,609   44,338,543
20         NaN  $137,774,361,015   43,682,260
21         NaN  $152,586,154,514   42,987,461
22         NaN  $147,607,982,695   42,241,011
23         NaN  $155,460,285,076   41,435,758
24         NaN  $139,752,450,152   40,564,059
```

Data Analysis and Visualization

[ ]:
```
station = df.groupby(['Station'])
station =station.aggregate(np.sum)
station['Total'] = station.sum(axis=1)
station
```

[ ]:
```
                        2005-2006  2006-2007  …  2015-2016  Total
Station                                       …
'King William''S Town'       4210       4067  …       4077  47317
'Low''S Creek'                274        252  …        213   2657
'Pilgrim''S Rest'             173        173  …        135   1502
'Rankin''S Pass'              100         95  …         99   1191
'Simon''S Town'               676        662  …        907   8549
…                             …          …    …        …     …
Zebediela                     637        582  …       1064   7260
Zeerust                      1330       1385  …       1117  13307
Zele                          787        802  …        682   8657
Zonkizizwe                    859        842  …        913  10391
Zwelitsha                    1584       1647  …       1377  16518

[1143 rows x 12 columns]
```

[ ]:
```
#The station that reported the most crimes
station[station['Total']==max(station['Total'])]
```

[ ]:
```
                 2005-2006  2006-2007  2007-2008  …  2014-2015  2015-2016
Total
Station                                           …
```

```
Mitchells Plain      23599      26131      27453  …      20366      19499
278498

[1 rows x 12 columns]
```

[ ]:
```
# Finding 10 stations with the HIGHEST reported crime case in the province
sorted1 =station.sort_values(by='Total', ascending= False)
sorted1.head(10)
```

[ ]:
```
                  2005-2006  2006-2007  …  2015-2016   Total
Station                                  …
Mitchells Plain       23599      26131  …      19499  278498
Cape Town Central     19773      18817  …      17785  193730
Jhb Central           22179      20768  …      14607  192238
Durban Central        19174      20677  …      14147  182836
Hillbrow              17172      15757  …      10933  149593
Park Road             14751      14257  …      11489  149420
Rustenburg            15174      14927  …      10823  145122
Pretoria Central      16249      15184  …      11444  141998
Honeydew               9669      10510  …      12889  134910
Booysens              17473      16686  …       6147  127844

[10 rows x 12 columns]
```

[ ]:
```
# # Finding 10 stations with the lowest reported crime case in the province
sorted2 =station.sort_values(by='Total', ascending=False)
sorted2.tail(10)
```

[ ]:
```
               2005-2006  2006-2007  2007-2008  …  2014-2015  2015-2016  Total
Station                                          …
Kolomane              26         17         29  …         30         17    275
Steunmekaar           37         34         17  …         19         12    243
Henderson             11         11         11  …         35         25    227
Wanda                 23         26         11  …         14         25    194
Rossouw               23          6         12  …         21         22    183
Vorstershoop           9         10         32  …         17         11    178
Boetsap               14         18         14  …         18         16    159
Mbizeni               15         11         11  …          5          6    120
Mokopong               3         10         12  …          8          8    100
Elands Height          6          6         14  …          5          9     91

[10 rows x 12 columns]
```

[ ]:
```
#The station that reported the least crimes
station[station['Total']==min(station['Total'])]
```

[ ]:

Station	2005-2006	2006-2007	2007-2008	…	2014-2015	2015-2016	Total
Elands Height	6	6	14	…	5	9	91

[1 rows x 12 columns]

[ ]:
```
category = df.groupby(['Category'])
category =category.aggregate(np.sum)
category
category['Total'] = category.sum(axis=1)
category
#category.agg({'2005-2006' : 'sum', '2006-2007' : 'sum', '2007-2008' : 'sum',␣
 ↪'2008-2009' : 'sum', '2009-2010' : 'sum', '2010-2011' : 'sum', '2012-2013' :␣
 ↪'sum', '2014-2015' : 'sum', '2015-2016' : 'sum'})
```

[ ]:

Category	2005-2006	…	Total
All theft not mentioned elsewhere	424690	…	4120351
Arson	7247	…	67688
Assault with the intent to inflict grievous bod…	225659	…	2179207
Attempted murder	20369	…	193335
Bank robbery	59	…	628
Burglary at non-residential premises	54217	…	751717
Burglary at residential premises	261403	…	2763950
Carjacking	12783	…	137621
Commercial crime	51911	…	807206
Common assault	225436	…	2043267
Common robbery	74221	…	647739
Driving under the influence of alcohol or drugs	33076	…	660174
Drug-related crime	94801	…	1879871
Illegal possession of firearms and ammunition	13239	…	157902
Malicious damage to property	141776	…	1397845
Murder	18455	…	191973
Robbery at non-residential premises	4384	…	153617
Robbery at residential premises	10173	…	186629
Robbery of cash in transit	383	…	2809
Robbery with aggravating circumstances	119242	…	1284991
Sexual Offences	67064	…	678348
Sexual offences as result of police action	0	…	23791
Shoplifting	64433	…	798079
Stock-theft	26526	…	290649
Theft of motor vehicle and motorcycle	85595	…	745232
Theft out of or from motor vehicle	138586	…	1420789
Truck hijacking	829	…	12032

[27 rows x 12 columns]

[ ]:
```
#The most reported category of crime
category[category['Total']==max(category['Total'])]
```

[ ]:
```
                                2005-2006  2006-2007  …  2015-2016    Total
Category                                              …
All theft not mentioned elsewhere     424690     407714  …     340372  4120351

[1 rows x 12 columns]
```

[ ]:
```
# Finding 10 most reported crimes
sorted3 =category.sort_values(by='Total', ascending=False)
sorted3.head(10)
```

[ ]:
```
                                                  2005-2006  …    Total
Category                                                     …
All theft not mentioned elsewhere                    424690  …  4120351
Burglary at residential premises                     261403  …  2763950
Assault with the intent to inflict grievous bod…     225659  …  2179207
Common assault                                       225436  …  2043267
Drug-related crime                                    94801  …  1879871
Theft out of or from motor vehicle                   138586  …  1420789
Malicious damage to property                         141776  …  1397845
Robbery with aggravating circumstances               119242  …  1284991
Commercial crime                                      51911  …   807206
Shoplifting                                           64433  …   798079

[10 rows x 12 columns]
```

[ ]:
```
#The least reported category of crime
category[category['Total']==min(category['Total'])]
```

[ ]:
```
              2005-2006  2006-2007  2007-2008  …  2014-2015  2015-2016  Total
Category                                       …
Bank robbery         59        130        146  …         17          6    628

[1 rows x 12 columns]
```

[ ]:
```
# Finding 10 least reported crimes
sorted4 =category.sort_values(by='Total', ascending=False)
sorted4.tail(10)
```

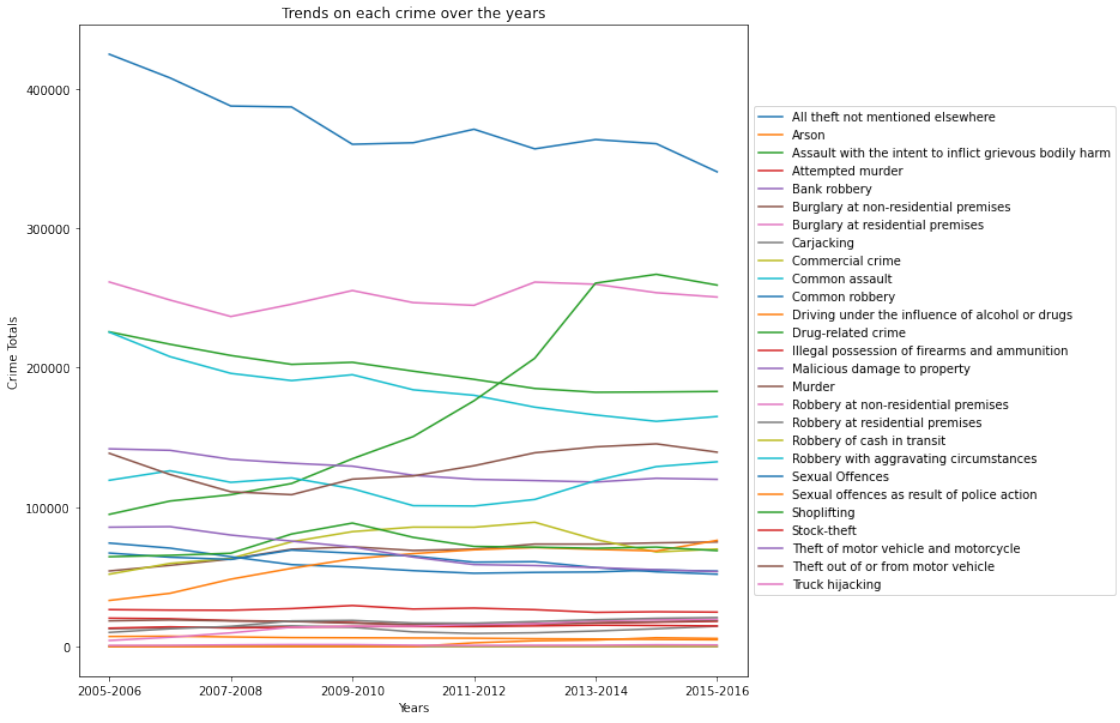
[ ]:
```
                                          2005-2006  …   Total
Category                                             …
Murder                                        18455  …  191973
Robbery at residential premises               10173  …  186629
Illegal possession of firearms and ammunition 13239  …  157902
Robbery at non-residential premises            4384  …  153617
```

```
Carjacking                                        12783  ...  137621
Arson                                              7247  ...   67688
Sexual offences as result of police action            0  ...   23791
Truck hijacking                                     829  ...   12032
Robbery of cash in transit                          383  ...    2809
Bank robbery                                         59  ...     628

[10 rows x 12 columns]
```

[ ]:
```
categorygraph = df.groupby(['Category'])
categorygraph = categorygraph.aggregate(np.sum)
categorygraph = categorygraph.T
```

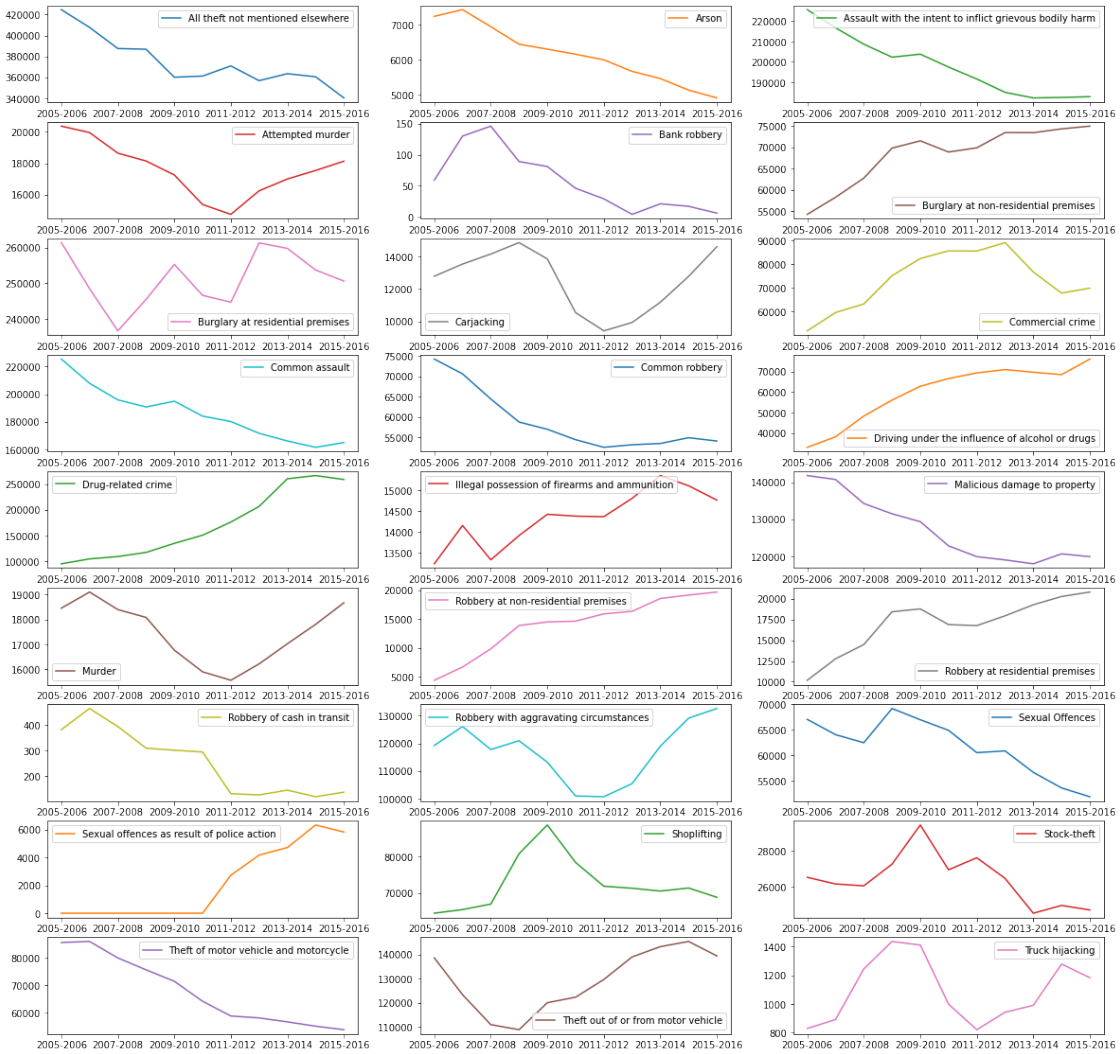
[ ]:
```
# Visualising crime by category
categorygraph.plot(kind='line', figsize=(10,10)).legend(loc='center left',␣
 ↪bbox_to_anchor=(1, 0.5))
plt.title('Trends on each crime over the years')
plt.xlabel('Years')
plt.ylabel('Crime Totals')
```

[ ]:
```
Text(0, 0.5, 'Crime Totals')
```



[ ]:
```
# Splitting the plots to get a better view of the crime categories trend
```

```
# categorygraph.plot(kind='line', subplots=True, figsize=(10,10)).
  ↪legend(loc='center left', bbox_to_anchor=(1, 0.5))
categorygraph.plot(subplots=True, layout=(9, 3), figsize=(20, 20),␣
  ↪sharex=False);
# categorygraph.plot(subplots=True, layout=(27, 1), figsize=(20, 20),␣
  ↪sharex=False);
```



[ ]:
```
provincetotal = df.groupby('Province')
provincetotal =provincetotal.aggregate(np.sum)
provincetotal['Total'] = provincetotal.sum(axis=1)
provincetotal
```

[ ]:

	2005-2006	2006-2007	2007-2008	…	2014-2015	2015-2016
Total						

```
Province                                                  …
Eastern Cape       238977     228884     220813  …     202582     196089
2370079
Free State         137987     128227     127955  …     118879     117688
1397044
Gauteng            654817     639635     615618  …     637332     622218
6855654
Kwazulu/Natal      345784     343798     328368  …     348394     342772
3808898
Limpopo            106983     104857      97166  …     124986     129323
1201185
Mpumalanga         134829     131444     125954  …     117203     119526
1370933
North West         118840     112471     112340  …     114270     114335
1255258
Northern Cape       56515      52689      48954  …      49897      50665
546262
Western Cape       381825     396712     395281  …     492963     490383
4792127

[9 rows x 12 columns]
```

[ ]:
```
provinces = df.groupby('Province')
provinces =provinces.aggregate(np.sum)
total1 = provinces['2005-2006'].sum()
total2 = provinces['2006-2007'].sum()
total3 = provinces['2007-2008'].sum()
total4 = provinces['2008-2009'].sum()
total5 = provinces['2009-2010'].sum()
total6 = provinces['2010-2011'].sum()
total7 = provinces['2011-2012'].sum()
total8 = provinces['2012-2013'].sum()
total9 = provinces['2013-2014'].sum()
total10 = provinces['2014-2015'].sum()
total11 = provinces['2015-2016'].sum()
totals = [pd.Series([ total1,
                      total2,
                      total3,
                      total4,
                      total5,
                      total6,
                      total7,
                      total8,
                      total9,
                      total10,
                      total11], index=provinces.columns)]
```

```
provinces =provinces.append(totals )
provinces
```

[ ]:
```
                2005-2006  2006-2007  2007-2008  …  2013-2014  2014-2015
2015-2016
Eastern Cape       238977     228884     220813  …     210248     202582
196089
Free State         137987     128227     127955  …     126290     118879
117688
Gauteng            654817     639635     615618  …     636195     637332
622218
Kwazulu/Natal      345784     343798     328368  …     355729     348394
342772
Limpopo            106983     104857      97166  …     117638     124986
129323
Mpumalanga         134829     131444     125954  …     115996     117203
119526
North West         118840     112471     112340  …     113935     114270
114335
Northern Cape       56515      52689      48954  …      48947      49897
50665
Western Cape       381825     396712     395281  …     479022     492963
490383
0                 2176557    2138717    2072449  …    2204000    2206506
2182999

[10 rows x 11 columns]
```

[ ]:
```
# year that had the highest rate of crime
# renaming 0 with Total
# df2.rename({'0':'Total'}, inplace=True)
index = provinces.index
index_list = index.tolist()
index_list
index_list[9] = 'Total'
provinces.index = index_list
provinces
#provinces['max_value'] = provinces.max(axis=1)
#provinces['total'] = provinces.sum(axis=1)
```

[ ]:
```
                2005-2006  2006-2007  2007-2008  …  2013-2014  2014-2015
2015-2016
Eastern Cape       238977     228884     220813  …     210248     202582
196089
Free State         137987     128227     127955  …     126290     118879
117688
Gauteng            654817     639635     615618  …     636195     637332
```

```
622218
Kwazulu/Natal      345784     343798     328368  …     355729     348394
342772
Limpopo            106983     104857      97166  …     117638     124986
129323
Mpumalanga         134829     131444     125954  …     115996     117203
119526
North West         118840     112471     112340  …     113935     114270
114335
Northern Cape       56515      52689      48954  …      48947      49897
50665
Western Cape       381825     396712     395281  …     479022     492963
490383
Total             2176557    2138717    2072449  …    2204000    2206506
2182999

[10 rows x 11 columns]
```

[ ]:
```
# Finding the year with the highest number of crime
x = provinces.T
x[x['Total']==max(x['Total'])]
```

[ ]:
```
           Eastern Cape  Free State  …  Western Cape    Total
2014-2015        202582      118879  …        492963  2206506

[1 rows x 10 columns]
```

[ ]:
```
#Order of the crime rates from the highest sorted by years.
sorted5 =x.sort_values(by='Total', ascending= False)
sorted5
```

[ ]:
```
           Eastern Cape  Free State  …  Western Cape    Total
2014-2015        202582      118879  …        492963  2206506
2013-2014        210248      126290  …        479022  2204000
2015-2016        196089      117688  …        490383  2182999
2005-2006        238977      137987  …        381825  2176557
2012-2013        209124      131785  …        465994  2151032
2009-2010        217230      127512  …        417619  2145388
2006-2007        228884      128227  …        396712  2138717
2008-2009        216658      132335  …        398240  2121884
2011-2012        214462      126389  …        447238  2106560
2010-2011        215012      121997  …        426850  2091348
2007-2008        220813      127955  …        395281  2072449

[11 rows x 10 columns]
```

[ ]:
```
# Finding the year with the lowest number of crime
x = provinces.T
x[x['Total']==min(x['Total'])]
```

[ ]:
```
           Eastern Cape  Free State  …  Western Cape    Total
2007-2008        220813      127955  …        395281  2072449

[1 rows x 10 columns]
```

[ ]:
```
#x.info()
#x.append(df1, ignore_index=True)

x
```

[ ]:
```
           Eastern Cape  Free State  …  Western Cape    Total
2005-2006        238977      137987  …        381825  2176557
2006-2007        228884      128227  …        396712  2138717
2007-2008        220813      127955  …        395281  2072449
2008-2009        216658      132335  …        398240  2121884
2009-2010        217230      127512  …        417619  2145388
2010-2011        215012      121997  …        426850  2091348
2011-2012        214462      126389  …        447238  2106560
2012-2013        209124      131785  …        465994  2151032
2013-2014        210248      126290  …        479022  2204000
2014-2015        202582      118879  …        492963  2206506
2015-2016        196089      117688  …        490383  2182999

[11 rows x 10 columns]
```

[ ]:
```
a = list(df1['GDP_NOMINAL'])
b =list(df1['POPULATION'])
x['GDP']=a
x['Population']=b
x
#x['Population']= df1['POPULATION']
#x
```

[ ]:
```
           Eastern Cape  Free State  Gauteng  …    Total           GDP
Population
2005-2006        238977      137987   654817  …  2176557  257671413751
47880601
2006-2007        228884      128227   639635  …  2138717  271638484826
48489459
2007-2008        220813      127955   615618  …  2072449  299415505152
49119759
2008-2009        216658      132335   638186  …  2121884  286769839733
49779471
```

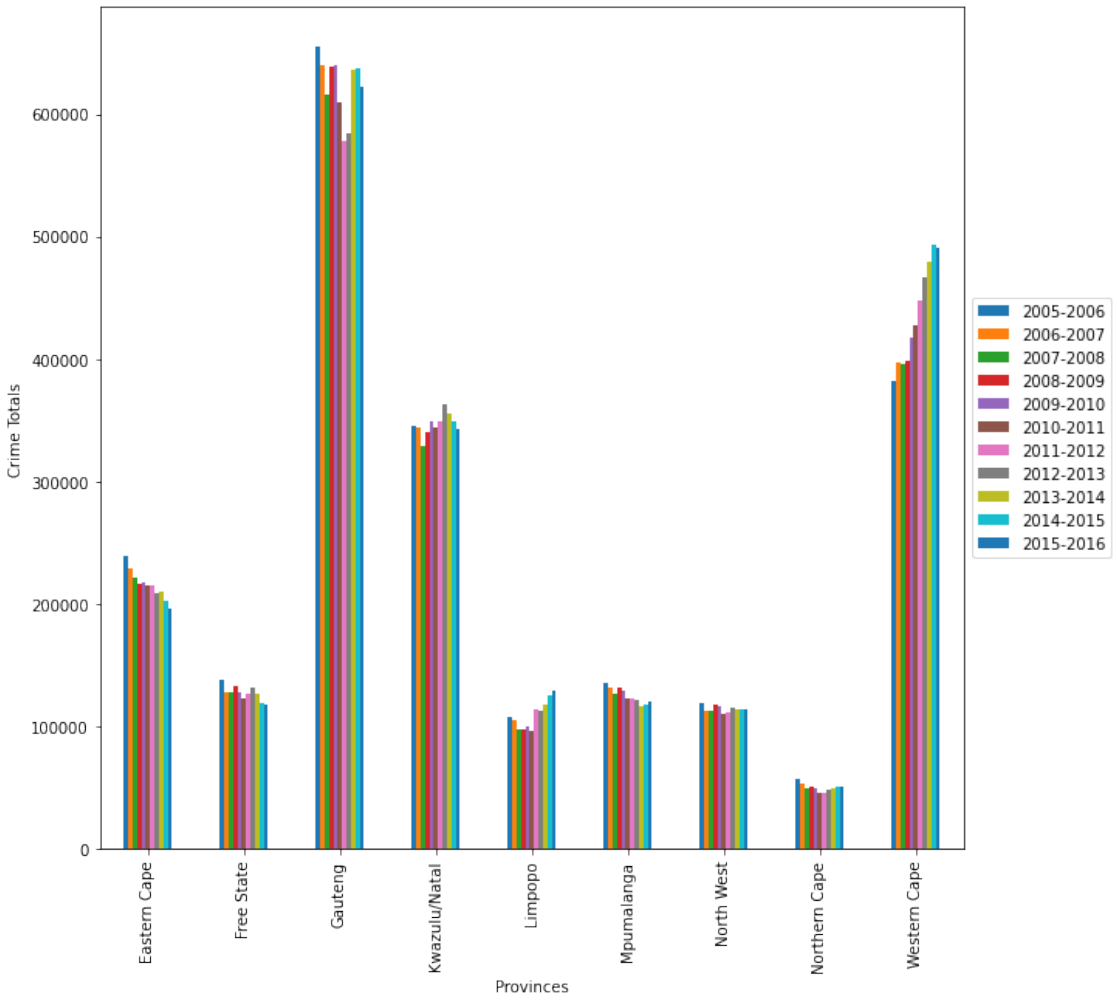
```
2009-2010        217230      127512   640074  …  2145388  295936485833
50477011
2010-2011        215012      121997   609305  …  2091348  375349442837
51216964
2011-2012        214462      126389   577959  …  2106560  416418874939
52003755
2012-2013        209124      131785   584315  …  2151032  396327875201
52832658
2013-2014        210248      126290   636195  …  2204000  366643223164
53687121
2014-2015        202582      118879   637332  …  2206506  350636208164
54544186
2015-2016        196089      117688   622218  …  2182999  317536830641
55386367

[11 rows x 12 columns]
```

[ ]:
```
# Ordering the crime rates
provincegraph = df.groupby('Province')
provincegraph =provincegraph.aggregate(np.sum)
provincegraph
```

[ ]:
```
               2005-2006  2006-2007  2007-2008  …  2013-2014  2014-2015
2015-2016
Province                                         …
Eastern Cape      238977     228884     220813  …     210248     202582
196089
Free State        137987     128227     127955  …     126290     118879
117688
Gauteng           654817     639635     615618  …     636195     637332
622218
Kwazulu/Natal     345784     343798     328368  …     355729     348394
342772
Limpopo           106983     104857      97166  …     117638     124986
129323
Mpumalanga        134829     131444     125954  …     115996     117203
119526
North West        118840     112471     112340  …     113935     114270
114335
Northern Cape      56515      52689      48954  …      48947      49897
50665
Western Cape      381825     396712     395281  …     479022     492963
490383

[9 rows x 11 columns]
```

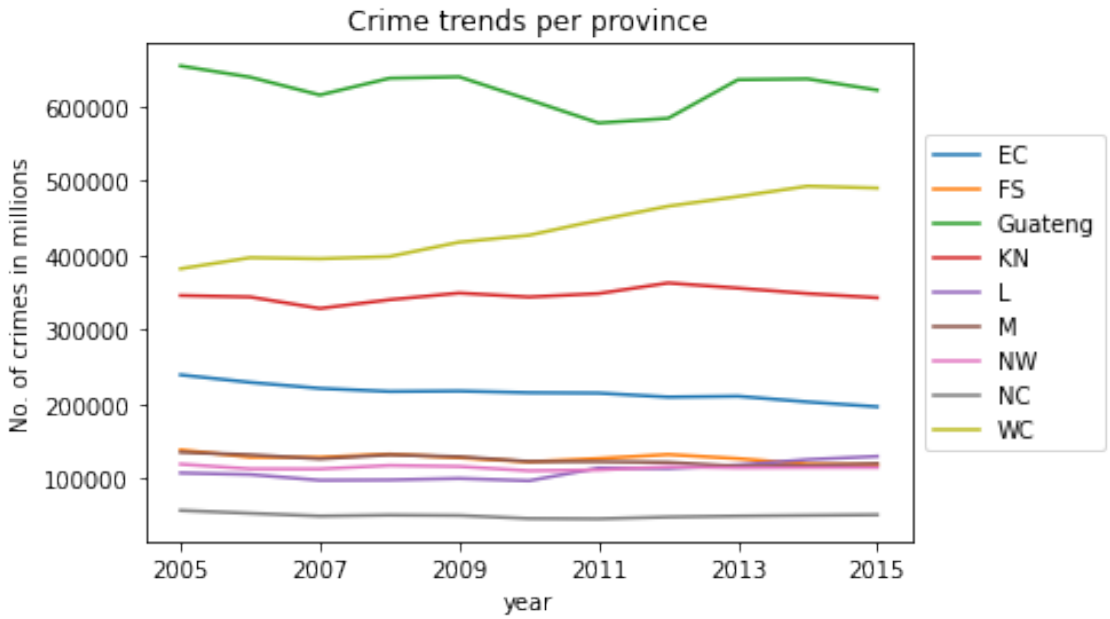
Visualizations graphing the crime committed per province

[ ]:
```
#Crime per province
provincegraph.plot(kind="bar", figsize=(10,10)).legend(loc='center left',␣
 ↪bbox_to_anchor=(1, 0.5))
plt.xlabel('Provinces')
plt.ylabel('Crime Totals')
plt.show()
```



[ ]:
```
# Plotting trends per province

# Column titles
x1 = list(provincegraph.columns)
y1 = x['Eastern Cape']
y2 = x['Free State']
y3 = x['Gauteng']
y4 = x['Kwazulu/Natal']
y5 = x['Limpopo']
```

```
y6 = x['Mpumalanga']
y7 = x['North West']
y8 = x['Northern Cape']
y9 = x['Western Cape']
positions = (0,2, 4, 6, 8, 10, 12, 14)
labels = ("2005", "2007", "2009", "2011", "2013", "2015")
plt.xticks(positions, labels)

plt.plot(x1, y1, label = "EC")
plt.plot(x1, y2, label = 'FS')
plt.plot(x1, y3, label = 'Guateng')
plt.plot(x1, y4, label='KN')
plt.plot(x1, y5, label='L')
plt.plot(x1, y6, label='M')
plt.plot(x1, y7, label='NW')
plt.plot(x1, y8, label='NC')
plt.plot(x1, y9, label='WC')

# show a legend on the plot
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))

# naming the x axis
plt.xlabel('year')
# naming the y axis
plt.ylabel('No. of crimes in millions')
# giving a title to my graph
plt.title('Crime trends per province')
plt.show()
```



Investingating correlation between population and crime rate

Converting data to reasonable measurement

[ ]:
```
x["GDP_in_10bs"] = x["GDP"]/10000000000
x["Population_in_2.5ms"] = x["Population"]/2500000
x["Total_crimes_in_0.1ms"] = x["Total"]/100000
```

[ ]:
```
new_df = x.drop(['Eastern Cape', 'Free State', 'Gauteng', 'Kwazulu/Natal',␣
↪'Limpopo',
       'Mpumalanga', 'North West', 'Northern Cape', 'Western Cape',␣
↪'GDP_in_10bs', 'Population_in_2.5ms',
       'Total_crimes_in_0.1ms'] , axis = 1)
new_df
```

[ ]:

	Total	GDP	Population
2005-2006	2176557	257671413751	47880601
2006-2007	2138717	271638484826	48489459
2007-2008	2072449	299415505152	49119759
2008-2009	2121884	286769839733	49779471
2009-2010	2145388	295936485833	50477011
2010-2011	2091348	375349442837	51216964
2011-2012	2106560	416418874939	52003755
2012-2013	2151032	396327875201	52832658
2013-2014	2204000	366643223164	53687121
2014-2015	2206506	350636208164	54544186
2015-2016	2182999	317536830641	55386367

#Testing the correlation There is a positive correlation between population and crime rate in South Africa increase in population has corresponding increase in crime rate

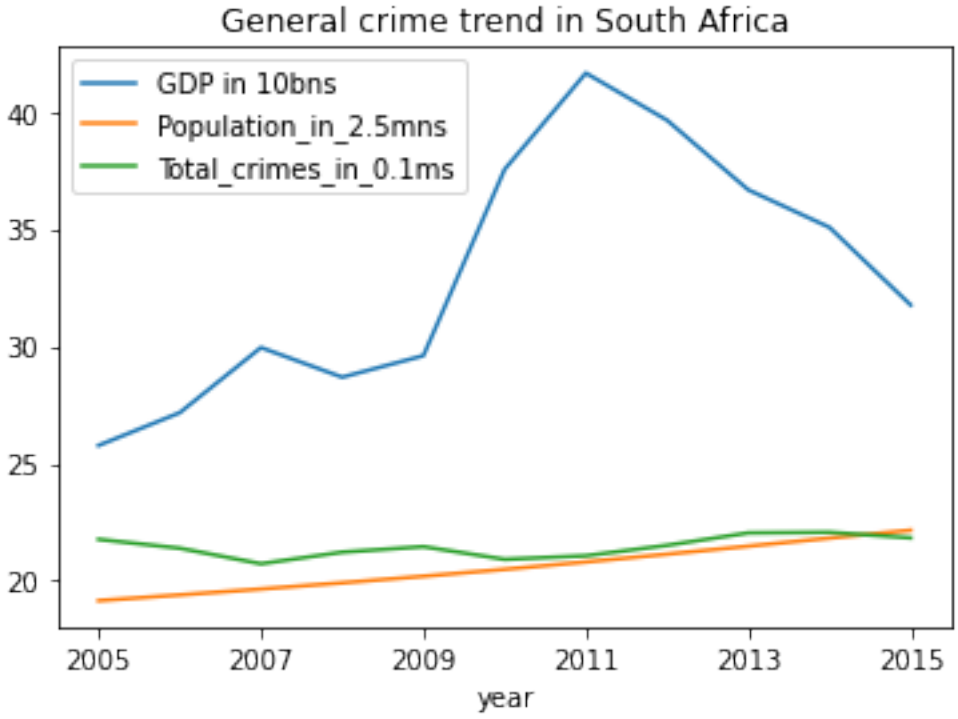
[ ]:
```
new_df.corr()
```

[ ]:

	Total	GDP	Population
Total	1.000000	-0.046573	0.521892
GDP	-0.046573	1.000000	0.622669
Population	0.521892	0.622669	1.000000

[ ]:
```
x1 = list(provincegraph.columns)
y1 = x['GDP_in_10bs']
y2 = x['Population_in_2.5ms']
y3 = x['Total_crimes_in_0.1ms']
positions = (0,2, 4, 6, 8, 10, 12, 14)
labels = ("2005", "2007", "2009", "2011", "2013", "2015")
plt.xticks(positions, labels)
```

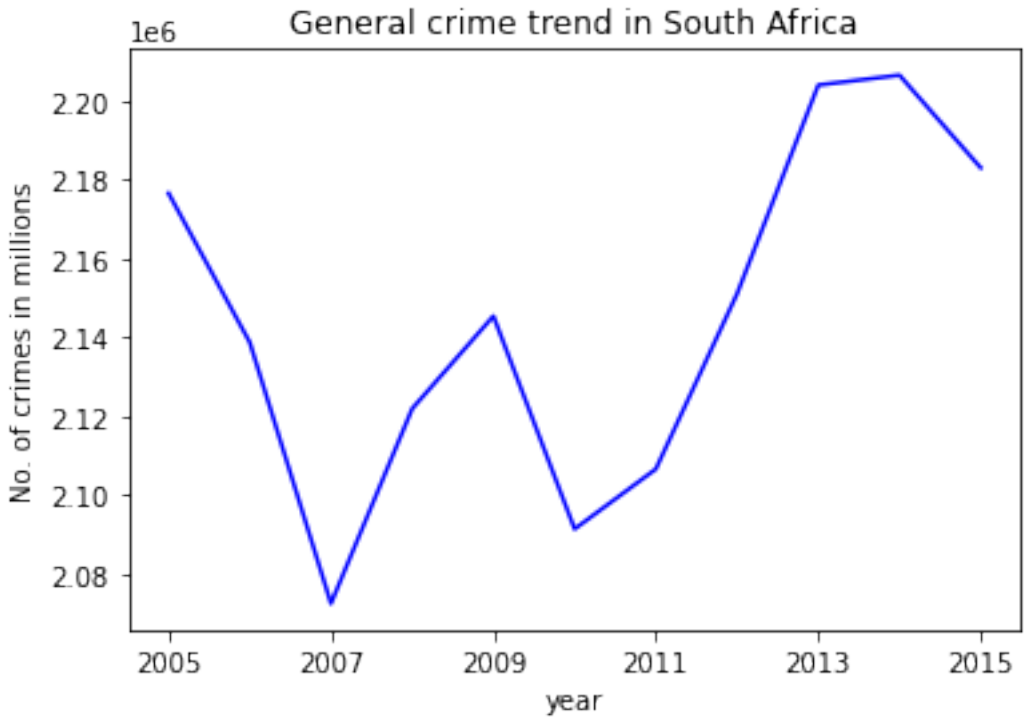
```
plt.plot(x1,y1, label = 'GDP in 10bns')
plt.plot(x1,y2, label = 'Population_in_2.5mns')
plt.plot(x1,y3, label = 'Total_crimes_in_0.1ms')

# naming the x axis
plt.xlabel('year')
# naming the y axis
#plt.ylabel('No. of crimes in millions')
# giving a title to my graph
plt.title('General crime trend in South Africa')
plt.legend()
plt.show()
```



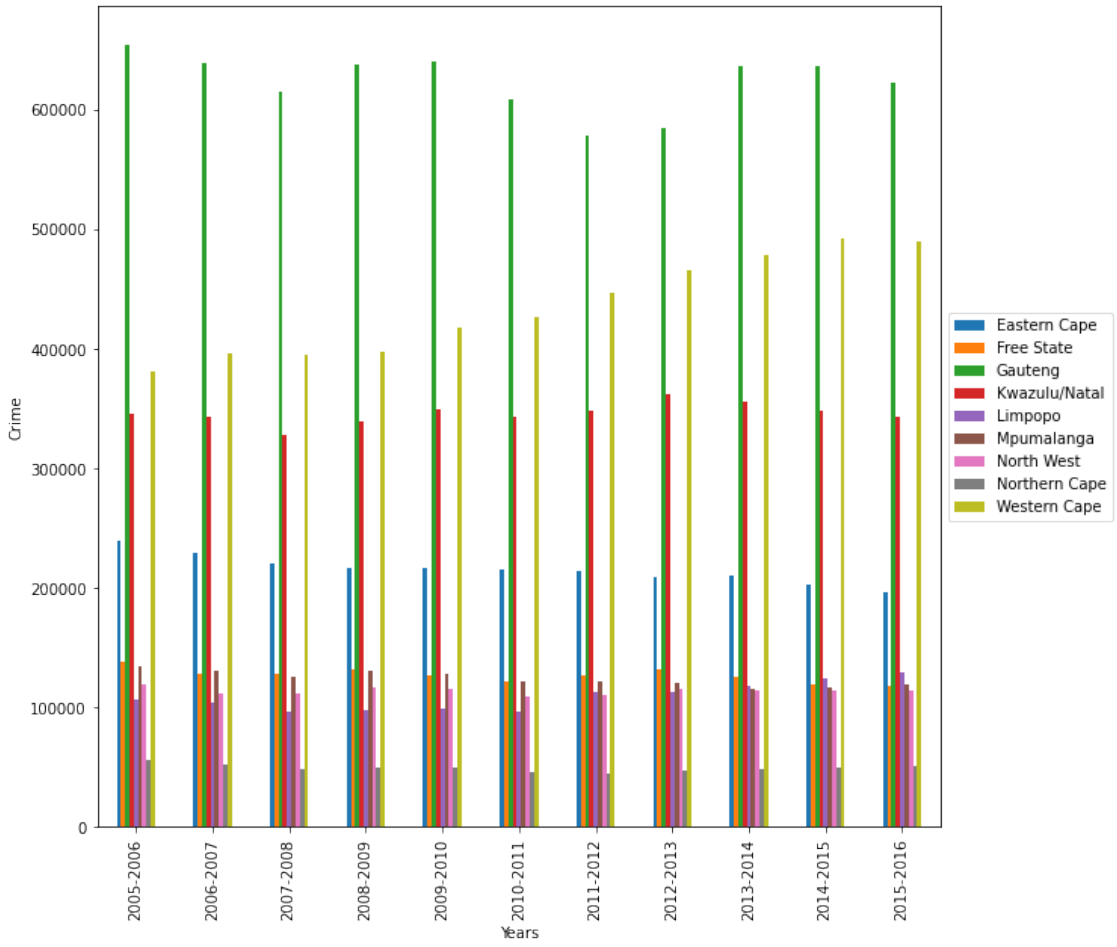
General trend of crime within the country

[ ]:
```
# Column titles
x1 = list(provincegraph.columns)
y1 = x['Total']
positions = (0,2, 4, 6, 8, 10, 12, 14)
labels = ("2005", "2007", "2009", "2011", "2013", "2015")
plt.xticks(positions, labels)

plt.plot(x1, y1, color='blue')
```

```
# naming the x axis
plt.xlabel('year')
# naming the y axis
plt.ylabel('No. of crimes in millions')
# giving a title to my graph
plt.title('General crime trend in South Africa')
plt.show()
```



[ ]:
```
#Crime rates per province in the years
pp = provincegraph.T
pp.plot(kind='bar', figsize=(10,10))
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.xlabel("Years")
plt.ylabel('Crime')
plt.show()
```



Recommendation

- To be able to capture a further and a deeper analysis on the crimes in South Africa, more indicators to correlate with can be analysed to produce a similar evident case but with different indicators i.e. income expenditure per household and the number of individual per household, can produce a diverse case in the analysis.
- To curb crime, more governance action should be implemented in Gauteng Province as it recorded the highest number of reported crimes
- Soweto, Pretoria and Johannesburg are in Gauteng. This accounts for the high number of crimes
- More studies should be conducted on preventive measures of sexual offences as a result of police action
- To supress crime by 40% unemployment amongest the youth should be looked upon since a greater number of individuals involved in crime belong to this bracket of the youth.
- More education forums should be done on drug and substance abuse as crimes commited under influence of drugs are on the rise.

AUTHOR: VINCENT G MUKOMBA