F.B.I Gun Data

May 8, 2020

0.1 PROJECT_02 : F.B.I GUN DATA

```
<li><a herf='#intro'>Introduction</a></li>
<li><a herf='#wrang'>Wrangle Data</a></li>
<li><a herf='#eda'>Explore data analysis</a></li>
<li><a herf="#conclusions">Conclusions</a></li>
<li><a herf='#commun'>Communicating Results</a></li>
```

INTRODUCTION - **OVERVIEW** > 1. The data comes from the FBI's National Instant Criminal Background Check System.

2. The NICS is used by to determine whether a prospective buyer is eligible to buy firearms or explosives.

3. Gun shops call into this system to ensure that each customer does not have a criminal record or isn't otherwise ineligible to make a purchase.

```
In [1]: # import requied models
        import numpy as np
        import pandas as pd
        from matplotlib import pyplot as plt
        %matplotlib inline
        import seaborn as sns
        sns.set_style('darkgrid')
```

WRANGLE DATA

Gaddring Data

```
In [2]: df_guns = pd.read_excel('gun_data.xlsx')
```

```
In [3]: df_guns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12485 entries, 0 to 12484
Data columns (total 27 columns):
month                        12485 non-null object
state                        12485 non-null object
permit                       12461 non-null float64
```

```
permit_recheck               1100 non-null float64
handgun                      12465 non-null float64
long_gun                     12466 non-null float64
other                        5500 non-null float64
multiple                     12485 non-null int64
admin                        12462 non-null float64
prepawn_handgun              10542 non-null float64
prepawn_long_gun             10540 non-null float64
prepawn_other                5115 non-null float64
redemption_handgun           10545 non-null float64
redemption_long_gun          10544 non-null float64
redemption_other             5115 non-null float64
returned_handgun             2200 non-null float64
returned_long_gun            2145 non-null float64
returned_other               1815 non-null float64
rentals_handgun              990 non-null float64
rentals_long_gun             825 non-null float64
private_sale_handgun         2750 non-null float64
private_sale_long_gun        2750 non-null float64
private_sale_other           2750 non-null float64
return_to_seller_handgun     2475 non-null float64
return_to_seller_long_gun    2750 non-null float64
return_to_seller_other       2255 non-null float64
totals                       12485 non-null int64
dtypes: float64(23), int64(2), object(2)
memory usage: 2.6+ MB
```

0.1.1 Checking data frame columns which has no valuse to the data frame

In [4]:
```
df_guns.iloc[:, 15:25].head(20)
```

Out[4]:
```
    returned_handgun  returned_long_gun  returned_other  rentals_handgun  \
0                0.0                0.0             0.0              0.0
1               28.0               30.0             0.0              0.0
2               82.0                5.0             0.0              0.0
3                0.0                0.0             0.0              0.0
4                0.0                0.0             0.0              0.0
5              202.0               46.0             1.0              0.0
6                0.0                0.0             0.0              0.0
7                0.0                0.0             0.0              0.0
8                0.0                0.0             0.0              0.0
9              264.0               28.0             0.0              0.0
10               0.0                0.0             0.0              0.0
11               0.0                0.0             0.0              0.0
12               2.0                0.0             0.0              0.0
13              27.0                5.0             1.0              0.0
14               0.0                0.0             0.0              0.0
```

```
15          22.0            0.0          0.0           0.0
16          25.0            1.0          0.0           0.0
17          10.0            7.0          0.0           0.0
18           1.0            1.0          0.0           0.0
19           0.0            0.0          0.0           0.0

    rentals_long_gun  private_sale_handgun  private_sale_long_gun  \
0                0.0                   9.0                   16.0
1                0.0                  17.0                   24.0
2                0.0                  38.0                   12.0
3                0.0                  13.0                   23.0
4                0.0                   0.0                    0.0
5                0.0                   0.0                    0.0
6                0.0                   0.0                    0.0
7                0.0                  55.0                   34.0
8                0.0                   0.0                    0.0
9                0.0                  11.0                    9.0
10               0.0                  17.0                    7.0
11               0.0                   0.0                    0.0
12               0.0                   0.0                    0.0
13               0.0                   7.0                   14.0
14               0.0                   0.0                    0.0
15               0.0                  75.0                   57.0
16               0.0                   1.0                    7.0
17               0.0                  16.0                   12.0
18               0.0                  21.0                   19.0
19               0.0                  28.0                   43.0

    private_sale_other  return_to_seller_handgun  return_to_seller_long_gun
0                  3.0                       0.0                        0.0
1                  1.0                       0.0                        0.0
2                  2.0                       0.0                        0.0
3                  0.0                       0.0                        2.0
4                  0.0                       0.0                        0.0
5                  0.0                       0.0                        0.0
6                  0.0                       0.0                        0.0
7                  3.0                       1.0                        2.0
8                  0.0                       0.0                        0.0
9                  0.0                       0.0                        1.0
10                 0.0                       0.0                        0.0
11                 0.0                       0.0                        0.0
12                 0.0                       0.0                        0.0
13                 1.0                       3.0                        0.0
14                 0.0                       0.0                        0.0
15                 6.0                       4.0                        4.0
16                 0.0                       0.0                        0.0
17                 0.0                       0.0                        1.0
18                 1.0                       2.0                        1.0
```

```
19                   3.0                        2.0                         2.0
```

0.1.2 Cleaning coulmns labels in the data frame

```
In [5]: column_name = df_guns.columns[15:26]

        df_guns = df_guns.drop(columns=column_name)
        df_guns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12485 entries, 0 to 12484
Data columns (total 16 columns):
month                     12485 non-null object
state                     12485 non-null object
permit                    12461 non-null float64
permit_recheck            1100 non-null float64
handgun                   12465 non-null float64
long_gun                  12466 non-null float64
other                     5500 non-null float64
multiple                  12485 non-null int64
admin                     12462 non-null float64
prepawn_handgun           10542 non-null float64
prepawn_long_gun          10540 non-null float64
prepawn_other             5115 non-null float64
redemption_handgun        10545 non-null float64
redemption_long_gun       10544 non-null float64
redemption_other          5115 non-null float64
totals                    12485 non-null int64
dtypes: float64(12), int64(2), object(2)
memory usage: 1.5+ MB
```

0.1.3 Checking for duplicate rows in data frame

```
In [6]: print( 'df_guns duplicate rows : {}'.format( sum( df_guns.duplicated() ) ) )
```

```
df_guns duplicate rows : 0
```

0.1.4 Viewing shape of the data frame with null values

```
In [7]: df_guns[df_guns.isnull()==False].shape
```

```
Out[7]: (12485, 16)
```

0.1.5 Converting columns object type to the right data type

```
In [8]: # converting month columns into pandase date time
        df_guns['month'] = pd.to_datetime(df_guns['month'])
```

```
In [9]: # converting all columns into float
        df_guns['multiple'] = pd.to_numeric(df_guns['multiple']).astype(float)
        df_guns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12485 entries, 0 to 12484
Data columns (total 16 columns):
month                       12485 non-null datetime64[ns]
state                       12485 non-null object
permit                      12461 non-null float64
permit_recheck              1100 non-null float64
handgun                     12465 non-null float64
long_gun                    12466 non-null float64
other                       5500 non-null float64
multiple                    12485 non-null float64
admin                       12462 non-null float64
prepawn_handgun             10542 non-null float64
prepawn_long_gun            10540 non-null float64
prepawn_other               5115 non-null float64
redemption_handgun          10545 non-null float64
redemption_long_gun         10544 non-null float64
redemption_other            5115 non-null float64
totals                      12485 non-null int64
dtypes: datetime64[ns](1), float64(13), int64(1), object(1)
memory usage: 1.5+ MB
```

0.1.6 Replacing all null valuse with there mean

```
In [10]: mean = df_guns.mean()
         df_guns = df_guns.fillna(mean)
         df_guns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12485 entries, 0 to 12484
Data columns (total 16 columns):
month                       12485 non-null datetime64[ns]
state                       12485 non-null object
permit                      12485 non-null float64
permit_recheck              12485 non-null float64
handgun                     12485 non-null float64
long_gun                    12485 non-null float64
other                       12485 non-null float64
multiple                    12485 non-null float64
admin                       12485 non-null float64
prepawn_handgun             12485 non-null float64
prepawn_long_gun            12485 non-null float64
prepawn_other               12485 non-null float64
```

```
redemption_handgun     12485 non-null float64
redemption_long_gun    12485 non-null float64
redemption_other       12485 non-null float64
totals                 12485 non-null int64
dtypes: datetime64[ns](1), float64(13), int64(1), object(1)
memory usage: 1.5+ MB
```

0.2 Explor DATA Analysis

0.2.1 Checking which column in the data frame will dicribe more about the data frame

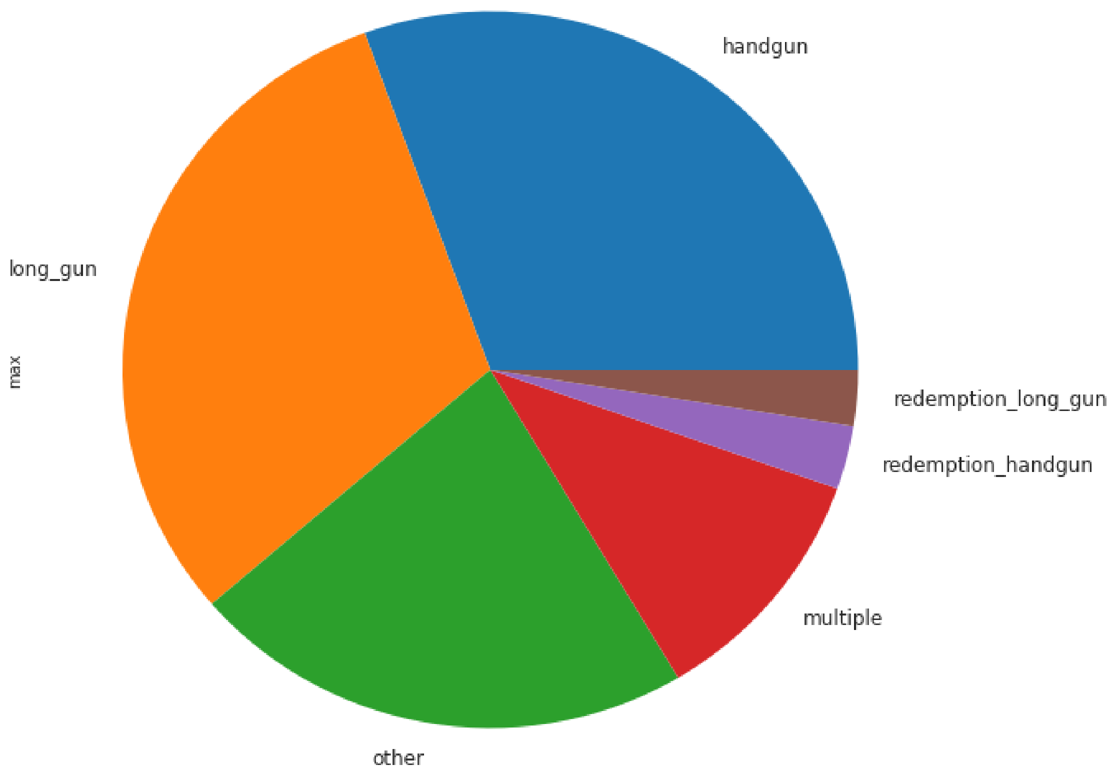
In [11]:
```
discribe = df_guns.describe()
discribe
```

Out[11]:
```
              permit  permit_recheck        handgun       long_gun  \
count   12485.000000    12485.000000   12485.000000   12485.000000
mean     6413.629404     1165.956364    5940.881107    7810.847585
std     23729.495816     2736.848174    8611.677589    9302.758891
min         0.000000        0.000000       0.000000       0.000000
25%         0.000000     1165.956364     868.000000    2079.000000
50%       522.000000     1165.956364    3067.000000    5130.000000
75%      4338.000000     1165.956364    7277.000000   10374.000000
max    522188.000000   116681.000000  107224.000000  108058.000000

              other      multiple         admin  prepawn_handgun  \
count  12485.000000  12485.000000  12485.000000     12485.000000
mean     360.471636    268.603364     58.898090         4.828021
std      895.634628    783.185073    604.257419        10.023040
min        0.000000      0.000000      0.000000         0.000000
25%      163.000000     15.000000      0.000000         0.000000
50%      360.471636    125.000000      0.000000         1.000000
75%      360.471636    301.000000      0.000000         4.828021
max    77929.000000  38907.000000  28083.000000       164.000000

       prepawn_long_gun  prepawn_other  redemption_handgun  \
count      12485.000000   12485.000000        12485.000000
mean           7.834156       0.165591          407.970413
std           15.130888       0.676584          720.023310
min            0.000000       0.000000            0.000000
25%            0.000000       0.000000            0.000000
50%            3.000000       0.165591          147.000000
75%            7.834156       0.165591          421.000000
max          269.000000      49.000000        10046.000000

       redemption_long_gun  redemption_other        totals
count         12485.000000      12485.000000  12485.000000
mean            599.332417          1.815249  21595.725911
std             875.000351          2.927929  32591.418387
```

min	0.000000	0.000000	0.000000
25%	0.000000	1.000000	4638.000000
50%	340.000000	1.815249	12399.000000
75%	670.000000	1.815249	25453.000000
max	8831.000000	79.000000	541978.000000

Pie Chart Of The Most Ordered Gun's

```
In [12]: discribe.iloc[7, np.r_[2:6, 10:12]].plot(kind='pie', figsize=(10,10), fontsize=12);
```



0.2.2 Observation

1. prepawn_long_gun
2. prepawn_handgun
3. prepawn_other
4. redemption_other

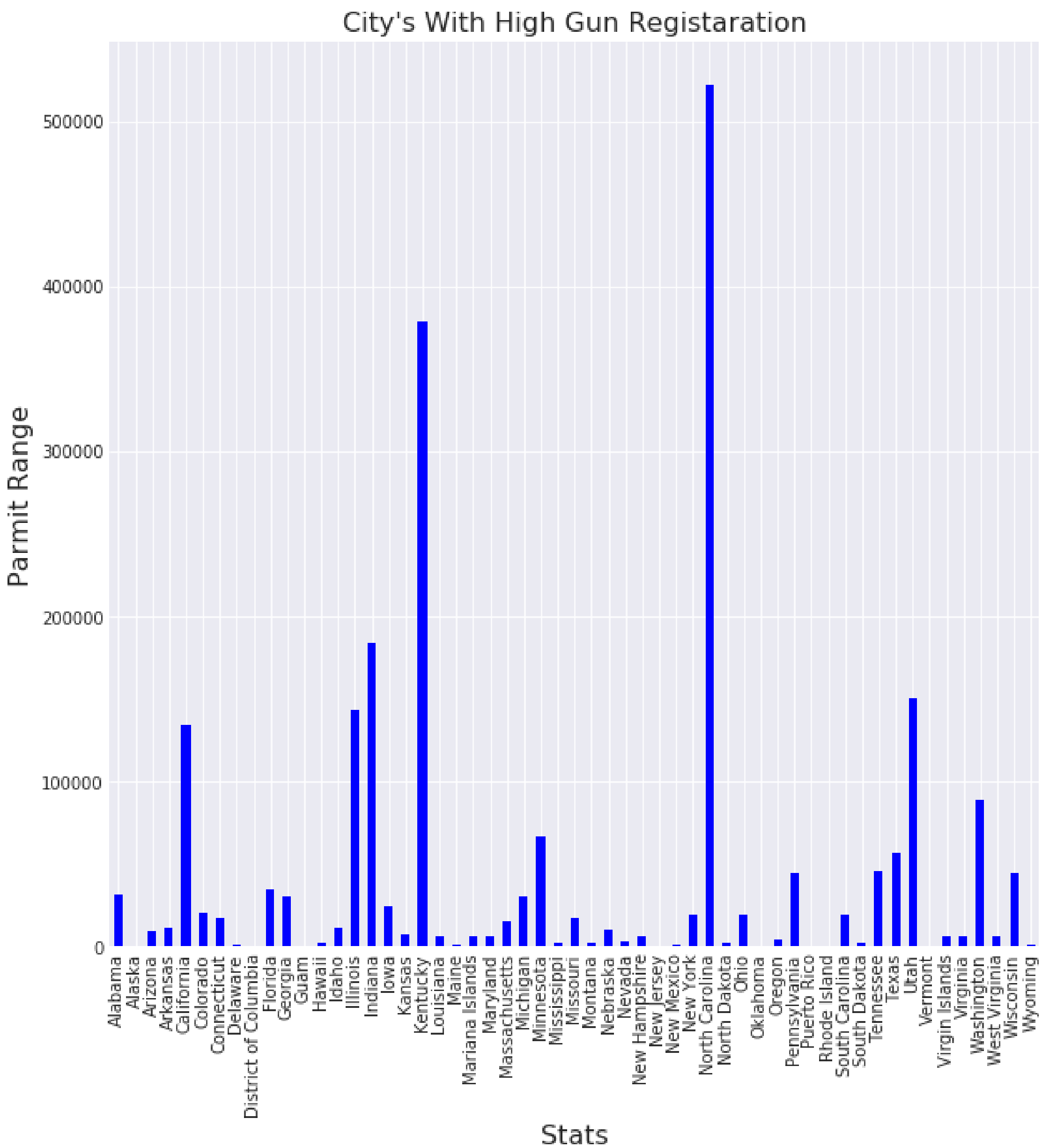
Has maximum order lower than 150 which can not be viwed on the pie chart

0.2.3 Targetting The Mean & Max of all state for easy exploration

```
In [13]: df_mean = df_guns.groupby('state').mean()
         df_max = df_guns.groupby('state').max()
```

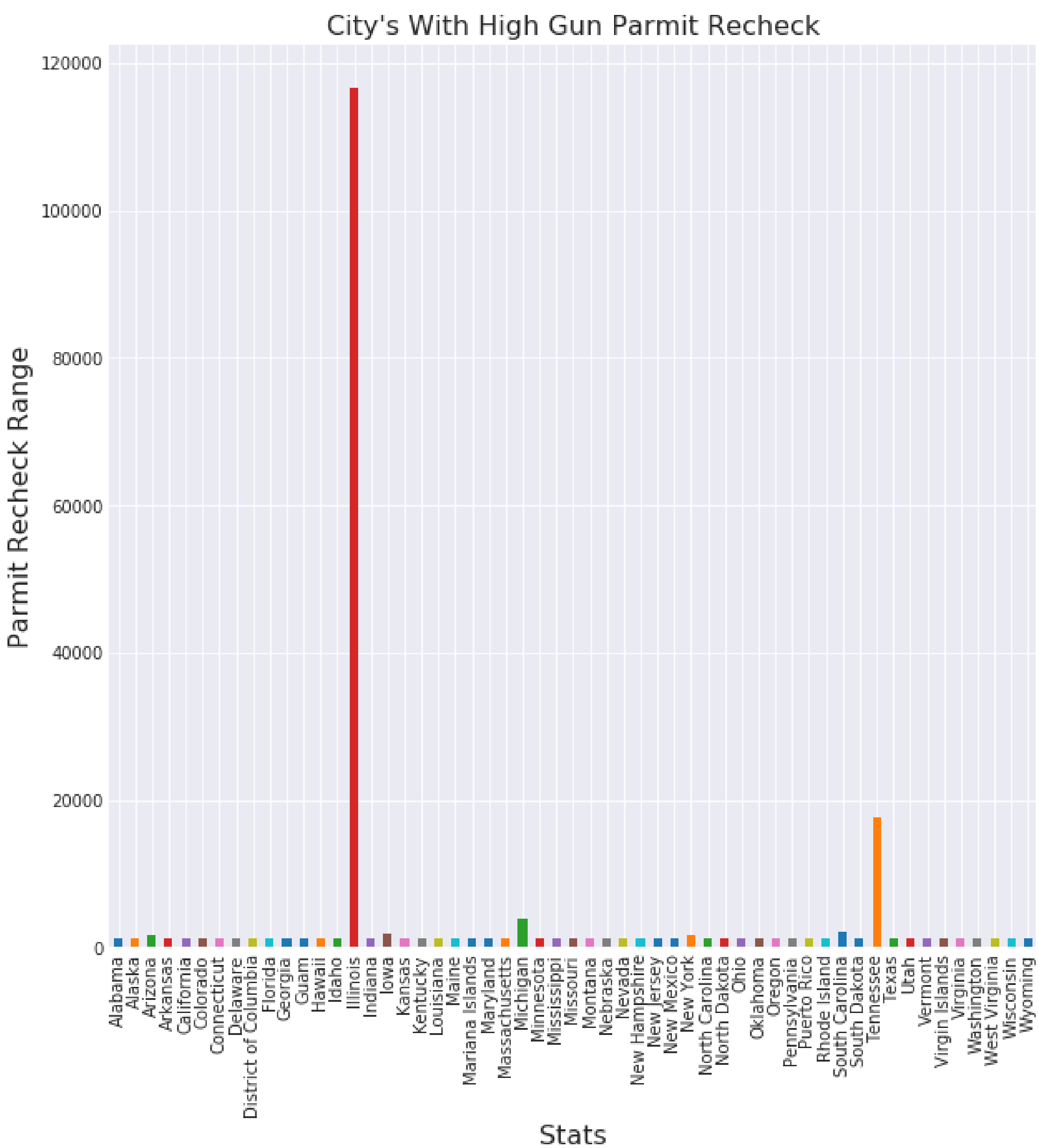
0.3 City's With High Gun Registararion

```
In [14]: # plotting data for city with high gun parmit over the past years
         df_max['permit'].plot(kind='bar',figsize=(10, 10), color='b')
         plt.title("City's With High Gun Registararion", fontsize=16)
         plt.xlabel('Stats', fontsize=16)
         plt.ylabel('Parmit Range', fontsize=16);
```



0.4 Which City's have Highest Gun Permit Recheck?

```
In [15]: # plotting for city's which are more safe by gun permit recheck over the year's
         df_max['permit_recheck'].plot(kind='bar',figsize=(10, 10))
         plt.title("City's With High Gun Parmit Recheck", fontsize=16)
         plt.xlabel('Stats', fontsize=16)
         plt.ylabel('Parmit Recheck Range', fontsize=16);
```



0.5 What census data is most associated with high gun per capita?

```
In [16]: df_guns.iloc[:, np.r_[0, 1, 4:8, 10:15]].groupby('state', as_index=False).max()
```

```
Out[16]:                state       month       handgun       long_gun  \
         0            Alabama 2017-09-01  47605.000000   42433.000000
         1             Alaska 2017-09-01   5265.000000    6304.000000
         2            Arizona 2017-09-01  25562.000000   19634.000000
         3           Arkansas 2017-09-01  13780.000000   19908.000000
         4         California 2017-09-01  74399.000000   93224.000000
         5           Colorado 2017-09-01  34653.000000   27112.000000
         6        Connecticut 2017-09-01  17828.000000   12310.000000
         7           Delaware 2017-09-01   3615.000000    3274.000000
         8   District of Columbia 2017-09-01     83.000000     193.000000
         9            Florida 2017-09-01  86940.000000   59904.000000
         10           Georgia 2017-09-01  34974.000000   32795.000000
         11              Guam 2017-09-01    145.000000      77.000000
         12            Hawaii 2017-09-01      2.000000      28.000000
         13             Idaho 2017-09-01   7023.000000   10015.000000
         14          Illinois 2017-09-01  60745.000000   30602.000000
         15           Indiana 2017-09-01  44150.000000   31940.000000
         16              Iowa 2017-09-01    436.000000    8121.000000
         17            Kansas 2017-09-01  13924.000000   17592.000000
         18          Kentucky 2017-09-01  25569.000000   26550.000000
         19         Louisiana 2017-09-01  27778.000000   33919.000000
         20             Maine 2017-09-01   7493.000000    7545.000000
         21   Mariana Islands 2017-09-01   5940.881107    7810.847585
         22          Maryland 2017-09-01  38055.000000   11046.000000
         23     Massachusetts 2017-09-01   9398.000000    5746.000000
         24          Michigan 2017-09-01  25578.000000   28946.000000
         25         Minnesota 2017-09-01  17763.000000   21302.000000
         26       Mississippi 2017-09-01  20000.000000   25190.000000
         27          Missouri 2017-09-01  43300.000000   36872.000000
         28           Montana 2017-09-01   6287.000000    8919.000000
         29          Nebraska 2017-09-01    282.000000    6715.000000
         30            Nevada 2017-09-01   9510.000000    9429.000000
         31     New Hampshire 2017-09-01   9537.000000    7366.000000
         32        New Jersey 2017-09-01   7726.000000    6531.000000
         33        New Mexico 2017-09-01   9027.000000   10310.000000
         34          New York 2017-09-01  15179.000000   34876.000000
         35    North Carolina 2017-09-01   2536.000000   41758.000000
         36      North Dakota 2017-09-01   3893.000000    6324.000000
         37              Ohio 2017-09-01  60280.000000   47197.000000
         38          Oklahoma 2017-09-01  28388.000000   29041.000000
         39            Oregon 2017-09-01  23952.000000   22599.000000
         40      Pennsylvania 2017-09-01  90055.000000  105826.000000
         41       Puerto Rico 2017-09-01   1458.000000     285.000000
         42      Rhode Island 2017-09-01   1831.000000    1865.000000
         43    South Carolina 2017-09-01  21019.000000   18436.000000
```

```
44          South Dakota 2017-09-01    5240.000000    8910.000000
45             Tennessee 2017-09-01   51923.000000   43721.000000
46                 Texas 2017-09-01  107224.000000  108058.000000
47                  Utah 2017-09-01    9885.000000   12126.000000
48               Vermont 2017-09-01    2428.000000    2622.000000
49        Virgin Islands 2017-09-01    5940.881107    7810.847585
50              Virginia 2017-09-01   41097.000000   39104.000000
51            Washington 2017-09-01   29473.000000   28335.000000
52         West Virginia 2017-09-01   15080.000000   18169.000000
53             Wisconsin 2017-09-01   25154.000000   22451.000000
54               Wyoming 2017-09-01    4157.000000    4722.000000

           other  multiple  prepawn_long_gun  prepawn_other  \
0    1698.000000    1752.0        132.000000       3.000000
1     394.000000     373.0         24.000000       1.000000
2    1345.000000    1102.0         30.000000       2.000000
3     365.000000     738.0        125.000000       3.000000
4   77929.000000   38907.0          7.834156       0.165591
5    1903.000000    8758.0          7.834156       0.165591
6    1276.000000     202.0          7.834156      49.000000
7     360.471636     137.0          7.834156       0.165591
8     360.471636       2.0          7.834156       0.165591
9    5096.000000    3436.0         54.000000       2.000000
10    863.000000    1227.0         61.000000       3.000000
11    360.471636      14.0          7.834156       0.165591
12    360.471636       2.0          7.834156       2.000000
13    360.471636    1414.0         21.000000       2.000000
14    360.471636    1963.0          7.834156       0.165591
15   2370.000000    1443.0         19.000000       2.000000
16    360.471636      33.0          7.834156       1.000000
17    605.000000     927.0         19.000000       1.000000
18    607.000000    1246.0        205.000000       2.000000
19   1453.000000    1199.0         99.000000       4.000000
20    360.471636     435.0         72.000000       1.000000
21    360.471636       7.0          7.834156       0.165591
22    360.471636      53.0         17.000000      13.000000
23   1498.000000     439.0          7.834156       1.000000
24    743.000000     363.0         33.000000       1.000000
25   1215.000000     706.0         21.000000       1.000000
26    540.000000     752.0        115.000000       2.000000
27   2272.000000    1679.0         97.000000       2.000000
28    360.471636     412.0         44.000000      10.000000
29    360.471636      18.0          8.000000       1.000000
30    538.000000     749.0          7.834156       1.000000
31    360.471636      36.0          7.834156       1.000000
32    444.000000     103.0          7.834156       0.165591
33    553.000000     553.0         33.000000       1.000000
34   1300.000000     389.0          7.834156       2.000000
```

35	1282.000000	489.0	109.000000	2.000000
36	360.471636	202.0	9.000000	1.000000
37	2942.000000	2488.0	133.000000	32.000000
38	1550.000000	1784.0	134.000000	4.000000
39	360.471636	781.0	7.834156	0.165591
40	360.471636	2311.0	7.834156	0.165591
41	360.471636	42.0	7.834156	1.000000
42	360.471636	365.0	7.834156	0.165591
43	1000.000000	2679.0	22.000000	3.000000
44	374.000000	341.0	9.000000	1.000000
45	1807.000000	2216.0	7.834156	0.165591
46	4585.000000	5293.0	269.000000	4.000000
47	379.000000	362.0	7.834156	2.000000
48	360.471636	130.0	7.834156	0.165591
49	360.471636	3.0	7.834156	0.165591
50	1829.000000	720.0	7.834156	0.165591
51	2650.000000	986.0	49.000000	4.000000
52	476.000000	863.0	58.000000	2.000000
53	1173.000000	165.0	12.000000	4.000000
54	360.471636	263.0	51.000000	1.000000

	redemption_handgun	redemption_long_gun	redemption_other
0	3380.000000	3308.000000	11.000000
1	407.970413	599.332417	5.000000
2	2179.000000	1204.000000	11.000000
3	1643.000000	3908.000000	6.000000
4	785.000000	831.000000	79.000000
5	407.970413	599.332417	1.815249
6	407.970413	599.332417	1.815249
7	407.970413	599.332417	7.000000
8	407.970413	599.332417	1.815249
9	4401.000000	1823.000000	12.000000
10	3047.000000	3630.000000	18.000000
11	407.970413	599.332417	1.815249
12	407.970413	599.332417	2.000000
13	523.000000	1111.000000	5.000000
14	407.970413	599.332417	1.815249
15	407.970413	1129.000000	23.000000
16	407.970413	599.332417	2.000000
17	892.000000	827.000000	12.000000
18	2857.000000	4384.000000	20.000000
19	1597.000000	2566.000000	7.000000
20	407.970413	599.332417	4.000000
21	407.970413	599.332417	1.815249
22	407.970413	599.332417	14.000000
23	407.970413	599.332417	14.000000
24	407.970413	599.332417	8.000000
25	407.970413	1031.000000	71.000000

```
26        2063.000000      3167.000000     10.000000
27        2002.000000      2805.000000     18.000000
28         677.000000      1887.000000      4.000000
29         407.970413       599.332417      2.000000
30         486.000000       599.332417      2.000000
31         407.970413       599.332417      5.000000
32         407.970413       599.332417      1.815249
33         717.000000      1525.000000     15.000000
34         407.970413       599.332417      8.000000
35        3126.000000      4667.000000     19.000000
36         407.970413       599.332417     12.000000
37        2191.000000      2156.000000     20.000000
38        2503.000000      3331.000000     20.000000
39         407.970413       599.332417      1.815249
40         407.970413       599.332417      1.815249
41         407.970413       599.332417      1.815249
42         407.970413       599.332417      1.815249
43        1717.000000      1786.000000     18.000000
44         407.970413       599.332417     10.000000
45         407.970413       599.332417      1.815249
46       10046.000000      8831.000000     62.000000
47         407.970413       599.332417      3.000000
48         407.970413       599.332417      3.000000
49         407.970413       599.332417      1.815249
50         407.970413       599.332417      1.815249
51        1866.000000      1931.000000     17.000000
52        1734.000000      3766.000000      8.000000
53         407.970413       645.000000     12.000000
54         407.970413       599.332417      3.000000
```

0.5.1 Observation

1. The Cencuse Data Mostly Associated With High Gun Per Capital is 2017-09-01

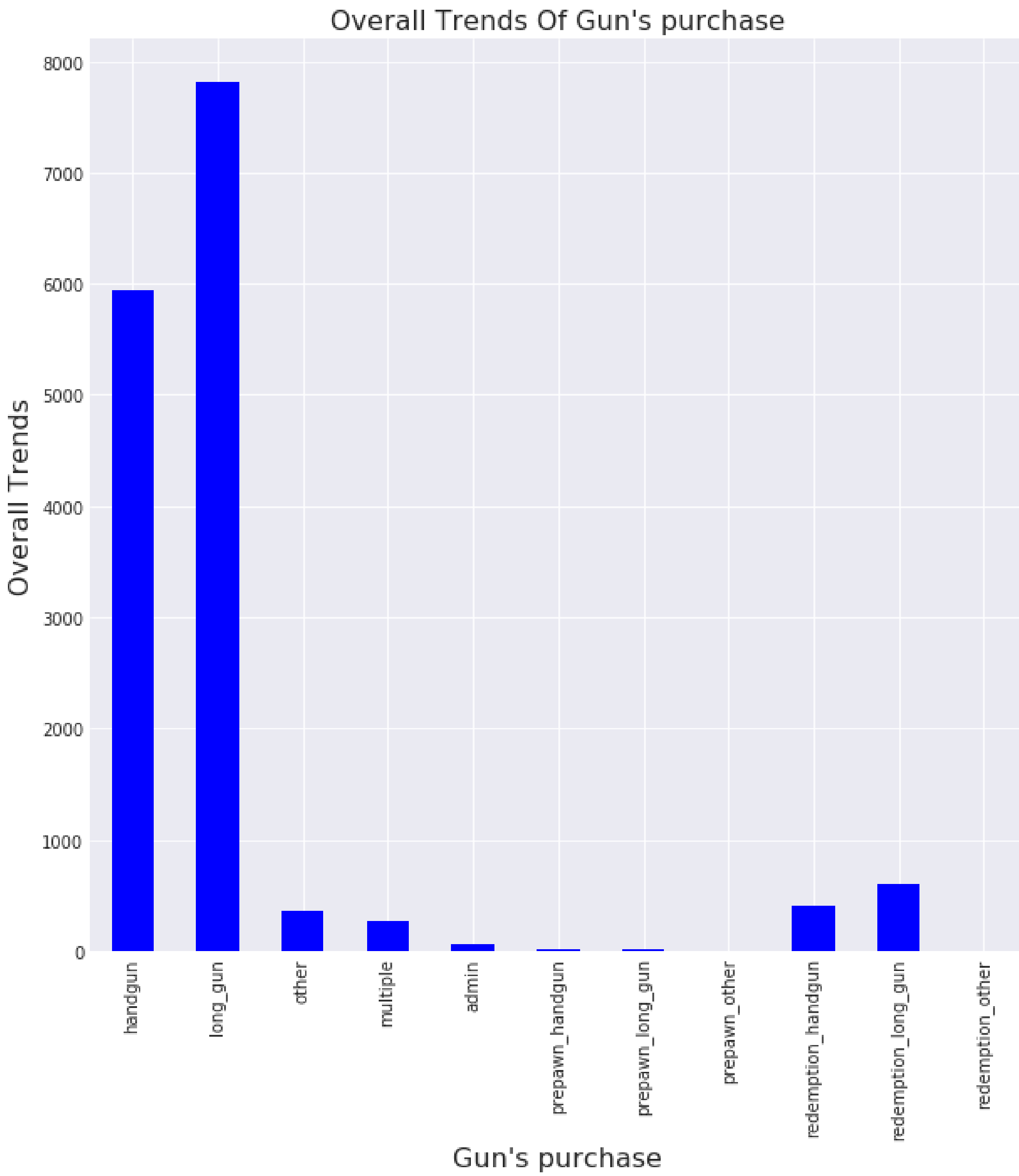
0.6 Overall Trend's Of Gun's purchase

In [17]:
```
#targetting overall trends of guns puchsed "mean" comperisime
overall = df_guns.loc[:, 'handgun':'redemption_other'].mean()
overall
```

Out[17]:
```
handgun                 5940.881107
long_gun                7810.847585
other                    360.471636
multiple                 268.603364
admin                     58.898090
prepawn_handgun            4.828021
prepawn_long_gun           7.834156
prepawn_other              0.165591
```

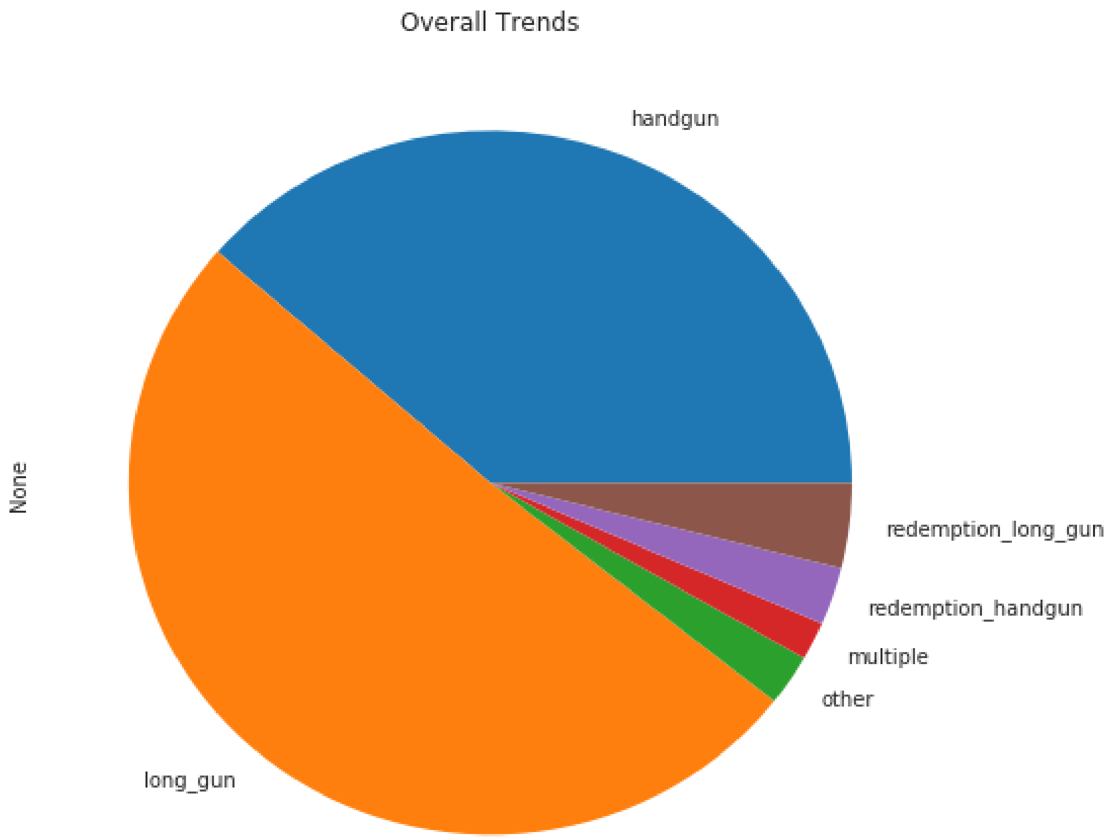
```
redemption_handgun      407.970413
redemption_long_gun     599.332417
redemption_other          1.815249
dtype: float64
```

In [18]:
```
# plotting bar chart overall trends of guns purchase by
overall.plot(kind='bar', figsize=(10,10), color='b')
plt.xlabel('Gun\'s purchase', fontsize=16)
plt.ylabel('Overall Trends', fontsize=16)
plt.title('Overall Trends Of Gun\'s purchase', fontsize=16);
```



0.6.1 plotting pie chart of Overall Trends Of Gun's purchase

```
In [19]: # iloc[7, np.r_[2:6, 10:12]]
         overall[overall >= 100].plot(kind='pie', figsize=(8,8))
         plt.title('Overall Trends');
```



0.6.2 Observation

1. prepawn_handgun
2. prepawn_long_gun
3. prepawn_other
4. redemption_other

- All have a mean below 100 which can not be seen on the pie chart

0.7 Overall Trens Of Gun's Purchase Per City's

```
In [20]: # targeting each city with there max guns purchase
         # what type of gun did each city purchase the most
```

```
# targeting guns types and city's
guns = df_guns.loc[:, 'handgun':'redemption_other'].columns

citys = df_guns.groupby('state').max().iloc[:, np.r_[3:7, 8:14]]
```

In [21]:
```
# targeting each city's and its highest gun's purchase
gun_max_name = []
gun_max_number = []
city_names = []

def citys_max_guns_use(data_set):

    for x in data_set.index:
        # target each citys
        citys_target = data_set.query('state == "{}"'.format(x))

        # target maximum for each city for easy acces
        citys_max = citys_target.max()

        # target the gun used in each city maximum
        citys_gun = citys_max[citys_max == citys_max.max()].index[0]

        # target the maximum number of guns puchase in each city
        maximum = citys_max.max()

        # appending all data
        gun_max_name.append(citys_gun)
        gun_max_number.append(maximum)
        city_names.append(x)

citys_max_guns_use(citys)
```

0.7.1 Observations

- df_citys_max data frame contains maximum guns purchase for each city

0.7.2 Limitation Of New Data Frame

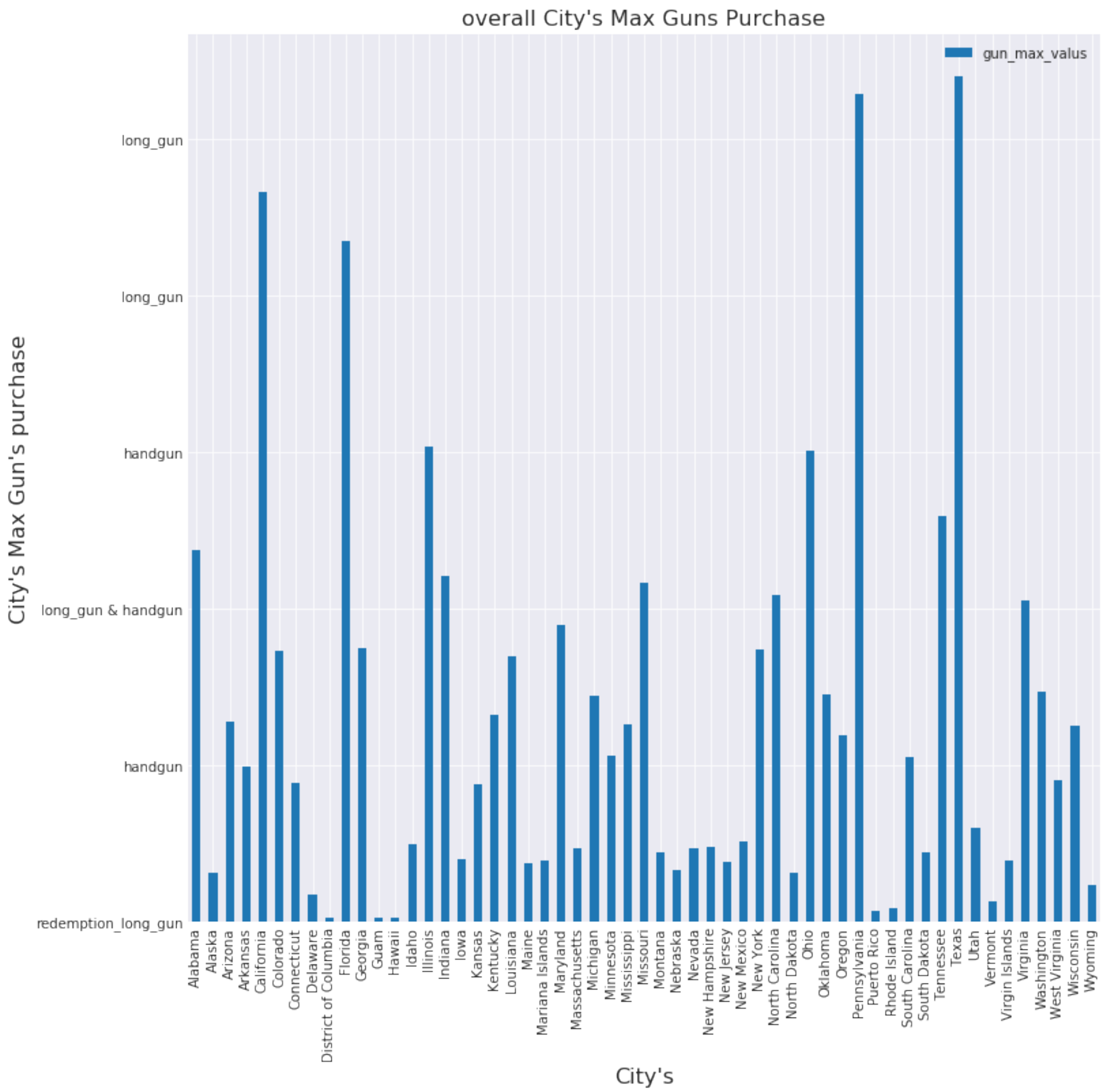
- df_citys_max data frame is limited by cencuse data month
- df_citys_max data frame is limited by guns's permit
- df_citys_max data frame is limited by guns's permit recheck

In [22]:
```
# creaing new data frame
df_citys_max = pd.DataFrame({'stats': city_names, 'gun_purchase': gun_max_name, 'gun_ma
df_citys_max.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55 entries, 0 to 54
```

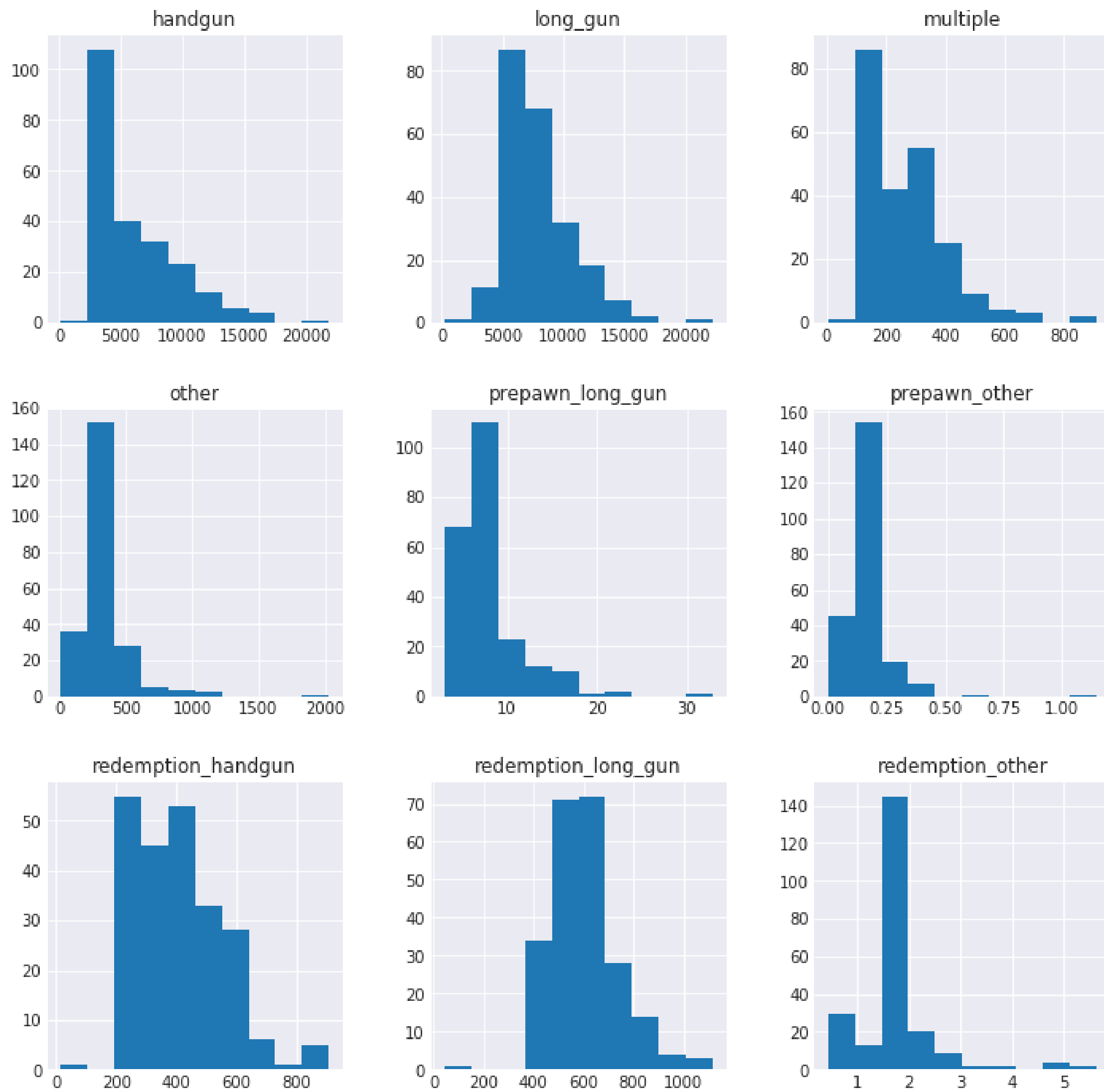
```
Data columns (total 3 columns):
stats              55 non-null object
gun_purchase       55 non-null object
gun_max_valus      55 non-null float64
dtypes: float64(1), object(2)
memory usage: 1.4+ KB
```

```
In [23]: # plotting graph
         x_axis = df_citys_max.groupby('stats').mean()
         x_axis.plot(kind='bar', figsize=(12,12));
         plt.yticks([0,20000, 40000, 60000, 80000, 100000], ['redemption_long_gun', 'handgun', '
         plt.xlabel('City\'s', fontsize=16)
         plt.ylabel('City\'s Max Gun\'s purchase', fontsize=16)
         plt.title('overall City\'s Max Guns Purchase',fontsize=16);
```



0.7.3 Moving Average Of Guns Per cencuse Histograme Graph

```
In [24]: guns_moving_avg = df_guns.iloc[:, np.r_[0, 1, 4:8, 10:15]].groupby('month', as_index=Fa
         guns_moving_avg.hist(figsize=(12, 12));
```



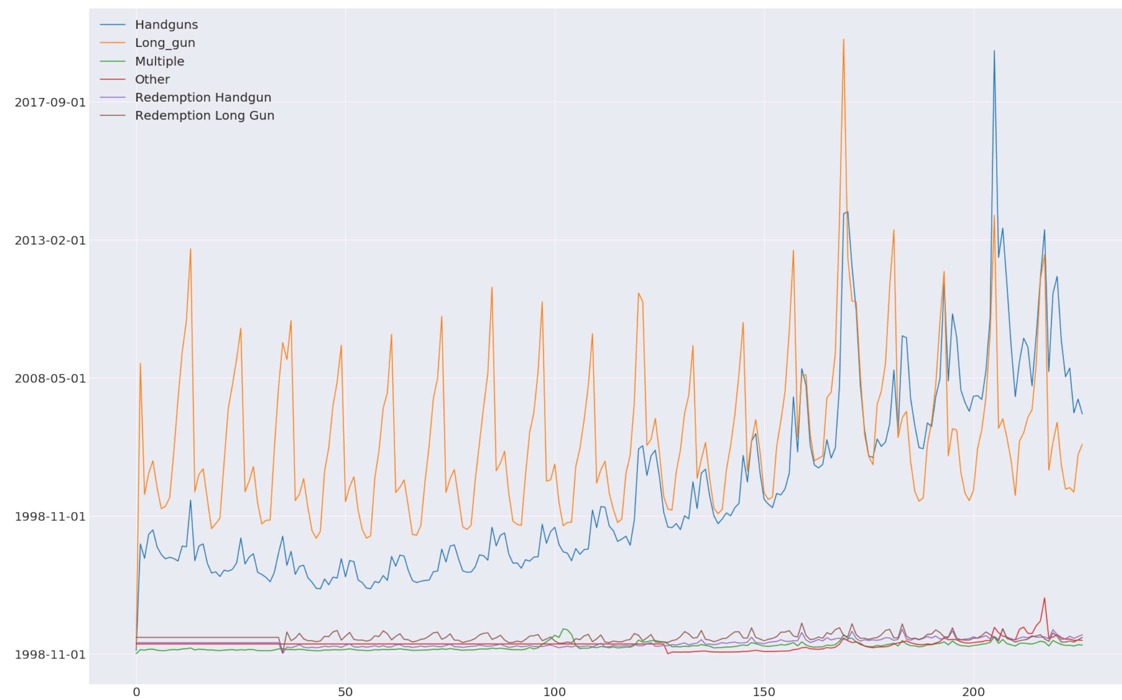
0.7.4 Moving Average Of Guns Per cencuse Straight Line Graph

Setting x axis and y axis

```
In [25]: x = np.arange(len(guns_moving_avg))
         handgun = guns_moving_avg['handgun']
         long_gun = guns_moving_avg['long_gun']
```

```
multiple = guns_moving_avg['multiple']
other = guns_moving_avg['other']
redemption_handgun = guns_moving_avg['redemption_handgun']
redemption_long_gun = guns_moving_avg['redemption_long_gun']
```

In [26]:
```
# plotting straight line graph
plt.rcParams["figure.figsize"] = (30,20)
plt.rcParams.update({'font.size': 20})
plt.plot(x, handgun, label='Handguns')
plt.plot(x, long_gun, label='Long_gun')
plt.plot(x, multiple, label='Multiple')
plt.plot(x, other, label='Other')
plt.plot(x, redemption_handgun, label='Redemption Handgun')
plt.plot(x, redemption_long_gun, label='Redemption Long Gun')
plt.yticks([0, 5000, 10000, 15000, 20000], ['1998-11-01','1998-11-01', '2008-05-01', '2
plt.legend();
```



In [27]:
```
guns_moving_avg.mean()
```

Out[27]:
```
handgun              5940.881107
long_gun             7810.847585
other                 360.471636
multiple              268.603364
prepawn_long_gun        7.834156
prepawn_other           0.165591
```

```
redemption_handgun      407.970413
redemption_long_gun     599.332417
redemption_other          1.815249
dtype: float64
```

0.7.5 observation

1. prepawn_long_gun
2. prepawn_handgun
3. prepawn_other
4. redemption_other

All have a mean valuse lower than 50 which can not be viwed on the line graph

1 Conclusions

WRANGLING DATA

```
1. Read file with pandas excel not csv
2. Converted all numerical column to float
3. Converted month data type from object to date time
4. Replaced all null value with the mean
5. Removed column labels that has no valuse to the data frame
```

Limitations For Explor DATA Analysis

```
1. prepawn_long_gun, prepawn_handgun, prepawn_other, redemption_other cloumns values are to low

2. prepawn_long_gun, prepawn_handgun, prepawn_other, redemption_other cloumns values are to low
```

2 comunication

BAR GRAPH

```
- To show a clear comperisime between maximum Gun's used
- To show a clear comperisime between Stats maximum Gun registration
- to show a clear comperisime between Stats maximum Gun registration recheck
- To show a clear comperisime between Guns used the most between each state
```

PIE CHART

```
- To show a clear percentage of  overall gun's purchas by mean valuse
- To show a clear percentage of most ordered gun's by max valuse
```

HISTOGRAME GRAPH

```
- Display moving average per cencuse for each guns purchased
```

Straight line graph

```
- Display moving average for each guns purchased per cencuse
```