Project: Investigate a Dataset - FBI Gun Data

Table of Contents

- Introduction
- Data Wrangling
- Exploratory Data Analysis
  - Research Question 1 (What census data is most associated with high gun per capita in 2010 and 2016?)
  - Research Question 2 (What is the overall trend of gun purchases?)
  - Research Question 3 (How many total checks have there been in each state since 1998?)
- Conclusions

Introduction

Dataset Description

This data comes from the FBI's National Instant Criminal Background Check System . From the official site:

> Mandated by the Brady Handgun Violence Prevention Act of 1993 and launched by the FBI on November 30, 1998, NICS is used by Federal Firearms Licensees (FFLs) to instantly determine whether a prospective buyer is eligible to buy firearms. Before ringing up the sale, cashiers call in a check to the FBI or to other designated agencies to ensure that each customer does not have a criminal record or isn't otherwise ineligible to make a purchase. More than 230 million such checks have been made, leading to more than 1.3 million denials.

Few notes to be considered in this investigation:

- **Firearm Background Checks Do Not Equal Sales**
- **States cannot be directly compared**

In [1]:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
dirname = os.getcwd()
base_path = os.path.join(dirname, '../data/ncis-and-census-data')
```

Data Wrangling

The data comes from the FBI's National Instant Criminal Background Check System. The NICS is used by to determine whether a prospective buyer is eligible to buy firearms or explosives. 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. The data has been supplemented with state level data from census.gov.

- The NICS data is found in one sheet of an .xlsx file. It contains the number of firearm checks by month, state, and type.
- The U.S. census data is found in a .csv file. It contains several variables at the state level. Most variables just have one data point per state (2016), but a few have data for more than one year.

have one data point per state (2016), but a few have data for more than one year.

In [3]:

```
df_gun = pd.read_excel(os.path.join(base_path, 'gun_data.xlsx'), sheet_name='Sheet1', en
gine='openpyxl')
df_gun.info()
df_gun.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12485 entries, 0 to 12484
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   month                      12485 non-null  object 
 1   state                      12485 non-null  object 
 2   permit                     12461 non-null  float64
 3   permit_recheck             1100 non-null   float64
 4   handgun                    12465 non-null  float64
 5   long_gun                   12466 non-null  float64
 6   other                      5500 non-null   float64
 7   multiple                   12485 non-null  int64  
 8   admin                      12462 non-null  float64
 9   prepawn_handgun            10542 non-null  float64
 10  prepawn_long_gun           10540 non-null  float64
 11  prepawn_other              5115 non-null   float64
 12  redemption_handgun         10545 non-null  float64
 13  redemption_long_gun        10544 non-null  float64
 14  redemption_other           5115 non-null   float64
 15  returned_handgun           2200 non-null   float64
 16  returned_long_gun          2145 non-null   float64
 17  returned_other             1815 non-null   float64
 18  rentals_handgun            990 non-null    float64
 19  rentals_long_gun           825 non-null    float64
 20  private_sale_handgun       2750 non-null   float64
 21  private_sale_long_gun      2750 non-null   float64
 22  private_sale_other         2750 non-null   float64
 23  return_to_seller_handgun   2475 non-null   float64
 24  return_to_seller_long_gun  2750 non-null   float64
 25  return_to_seller_other     2255 non-null   float64
 26  totals                     12485 non-null  int64  
dtypes: float64(23), int64(2), object(2)
memory usage: 2.6+ MB
```

Out[3]:

	month	state	permit	permit_recheck	handgun	long_gun	other	multiple	admin	prepawn_handgun	...	returned_ot
0	2017-09	Alabama	16717.0	0.0	5734.0	6320.0	221.0	317	0.0	15.0	...	
1	2017-09	Alaska	209.0	2.0	2320.0	2930.0	219.0	160	0.0	5.0	...	
2	2017-09	Arizona	5069.0	382.0	11063.0	7946.0	920.0	631	0.0	13.0	...	
3	2017-09	Arkansas	2935.0	632.0	4347.0	6063.0	165.0	366	51.0	12.0	...	
4	2017-09	California	57839.0	0.0	37165.0	24581.0	2984.0	0	0.0	0.0	...	

5 rows × 27 columns

The dataset is pretty clean. All numbers are floats.

In [4]:

```
df_census = pd.read_csv(os.path.join(base_path, 'U.S. Census Data.csv'))
df_census.info()
```

```
df_census.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 85 entries, 0 to 84
Data columns (total 52 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Fact            80 non-null     object
 1   Fact Note       28 non-null     object
 2   Alabama         65 non-null     object
 3   Alaska          65 non-null     object
 4   Arizona         65 non-null     object
 5   Arkansas        65 non-null     object
 6   California      65 non-null     object
 7   Colorado        65 non-null     object
 8   Connecticut     65 non-null     object
 9   Delaware        65 non-null     object
 10  Florida         65 non-null     object
 11  Georgia         65 non-null     object
 12  Hawaii          65 non-null     object
 13  Idaho           65 non-null     object
 14  Illinois        65 non-null     object
 15  Indiana         65 non-null     object
 16  Iowa            65 non-null     object
 17  Kansas          65 non-null     object
 18  Kentucky        65 non-null     object
 19  Louisiana       65 non-null     object
 20  Maine           65 non-null     object
 21  Maryland        65 non-null     object
 22  Massachusetts   65 non-null     object
 23  Michigan        65 non-null     object
 24  Minnesota       65 non-null     object
 25  Mississippi     65 non-null     object
 26  Missouri        65 non-null     object
 27  Montana         65 non-null     object
 28  Nebraska        65 non-null     object
 29  Nevada          65 non-null     object
 30  New Hampshire   65 non-null     object
 31  New Jersey      65 non-null     object
 32  New Mexico      65 non-null     object
 33  New York        65 non-null     object
 34  North Carolina  65 non-null     object
 35  North Dakota    65 non-null     object
 36  Ohio            65 non-null     object
 37  Oklahoma        65 non-null     object
 38  Oregon          65 non-null     object
 39  Pennsylvania    65 non-null     object
 40  Rhode Island    65 non-null     object
 41  South Carolina  65 non-null     object
 42  South Dakota    65 non-null     object
 43  Tennessee       65 non-null     object
 44  Texas           65 non-null     object
 45  Utah            65 non-null     object
 46  Vermont         65 non-null     object
 47  Virginia        65 non-null     object
 48  Washington      65 non-null     object
 49  West Virginia   65 non-null     object
 50  Wisconsin       65 non-null     object
 51  Wyoming         65 non-null     object
dtypes: object(52)
memory usage: 34.7+ KB
```

Out[4]:

	Fact	Fact Note	Alabama	Alaska	Arizona	Arkansas	California	Colorado	Connecticut	Delaware	...	South Dakota	Tenn
0	Population estimates, July 1, 2016, (V2016)	NaN	4,863,300	741,894	6,931,071	2,988,248	39,250,017	5,540,545	3,576,452	952,065	...	865454	66

	Fact	Fact Note	Alabama	Alaska	Arizona	Arkansas	California	Colorado	Connecticut	Delaware	...	South Dakota	Tenn
1	Population estimates base, April 1, 2010, (V2...	NaN	4,780,131	710,249	6,392,301	2,916,025	37,254,522	5,029,324	3,574,114	897,936	...	814195	63
2	Population, percent change - April 1, 2010 (es...	NaN	1.70%	4.50%	8.40%	2.50%	5.40%	10.20%	0.10%	6.00%	...	0.063	
3	Population, Census, April 1, 2010	NaN	4,779,736	710,231	6,392,017	2,915,918	37,253,956	5,029,196	3,574,097	897,934	...	814180	63
4	Persons under 5 years, percent, July 1, 2016, ...	NaN	6.00%	7.30%	6.30%	6.40%	6.30%	6.10%	5.20%	5.80%	...	0.071	

5 rows × 52 columns

Census dataset is not tidy. Datatypes needs to be revisited, commas and percentage symbols needs to be removed for numeric values.

In [5]:

```
df_census.describe()
```

Out[5]:

	Fact	Fact Note	Alabama	Alaska	Arizona	Arkansas	California	Colorado	Connecticut	Delaware	...	South Dakota	Tenn
count	80	28	65	65	65	65	65	65	65	65	...	65	
unique	80	15	65	64	64	64	63	64	63	64	...	65	
top	Population estimates, July 1, 2016, (V2016)	(c)	4,863,300	7.30%	50.30%	50.90%	6.80%	3.30%	0.10%	51.60%	...	865454	
freq	1	6	1	2	2	2	2	2	2	2	...	1	

4 rows × 52 columns

CSV data is not clean it needs to be restructured by transposing the dataframe.

In [6]:

```
print('Census Data Duplicate Records', sum(df_census.duplicated()))
print('Census Data Empty Records', sum(df_census.isna().any()))
df_census.isna().sum()
```

```
Census Data Duplicate Records 3
Census Data Empty Records 52
```

Out[6]:

```
Fact            5
Fact Note      57
Alabama        20
Alaska         20
Arizona        20
Arkansas       20
California     20
```

```
Colorado          20
Connecticut       20
Delaware          20
Florida           20
Georgia           20
Hawaii            20
Idaho             20
Illinois          20
Indiana           20
Iowa              20
Kansas            20
Kentucky          20
Louisiana         20
Maine             20
Maryland          20
Massachusetts     20
Michigan          20
Minnesota         20
Mississippi       20
Missouri          20
Montana           20
Nebraska          20
Nevada            20
New Hampshire     20
New Jersey        20
New Mexico        20
New York          20
North Carolina    20
North Dakota      20
Ohio              20
Oklahoma          20
Oregon            20
Pennsylvania      20
Rhode Island      20
South Carolina    20
South Dakota      20
Tennessee         20
Texas             20
Utah              20
Vermont           20
Virginia          20
Washington        20
West Virginia     20
Wisconsin         20
Wyoming           20
dtype: int64
```

Three rows are completely duplicated which is quite a mess.

In [7]:

```
df_census[df_census.isna()]
```

Out[7]:

	Fact	Fact Note	Alabama	Alaska	Arizona	Arkansas	California	Colorado	Connecticut	Delaware	...	South Dakota	Tennessee	Te
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
80	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
81	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N

	Fact	Fact Note	Alabama	Alaska	Arizona	Arkansas	California	Colorado	Connecticut	Delaware	...	South Dakota	Tennessee	Te
82	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
83	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
84	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N

85 rows × 52 columns

Empty rows is considered for almost every column.

In [8]:

```
df_gun.describe()
```

Out[8]:

	permit	permit_recheck	handgun	long_gun	other	multiple	admin	prepawn_har
count	12461.000000	1100.000000	12465.000000	12466.000000	5500.000000	12485.000000	12462.000000	10542.0
mean	6413.629404	1165.956364	5940.881107	7810.847585	360.471636	268.603364	58.898090	4.8
std	23752.338269	9224.200609	8618.584060	9309.846140	1349.478273	783.185073	604.814818	10.9
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	0.000000	0.000000	865.000000	2078.250000	17.000000	15.000000	0.000000	0.0
50%	518.000000	0.000000	3059.000000	5122.000000	121.000000	125.000000	0.000000	0.0
75%	4272.000000	0.000000	7280.000000	10380.750000	354.000000	301.000000	0.000000	5.0
max	522188.000000	116681.000000	107224.000000	108058.000000	77929.000000	38907.000000	28083.000000	164.0

8 rows × 25 columns

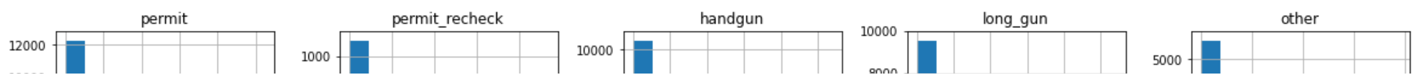
The most checks that were done in a month was over half a million.

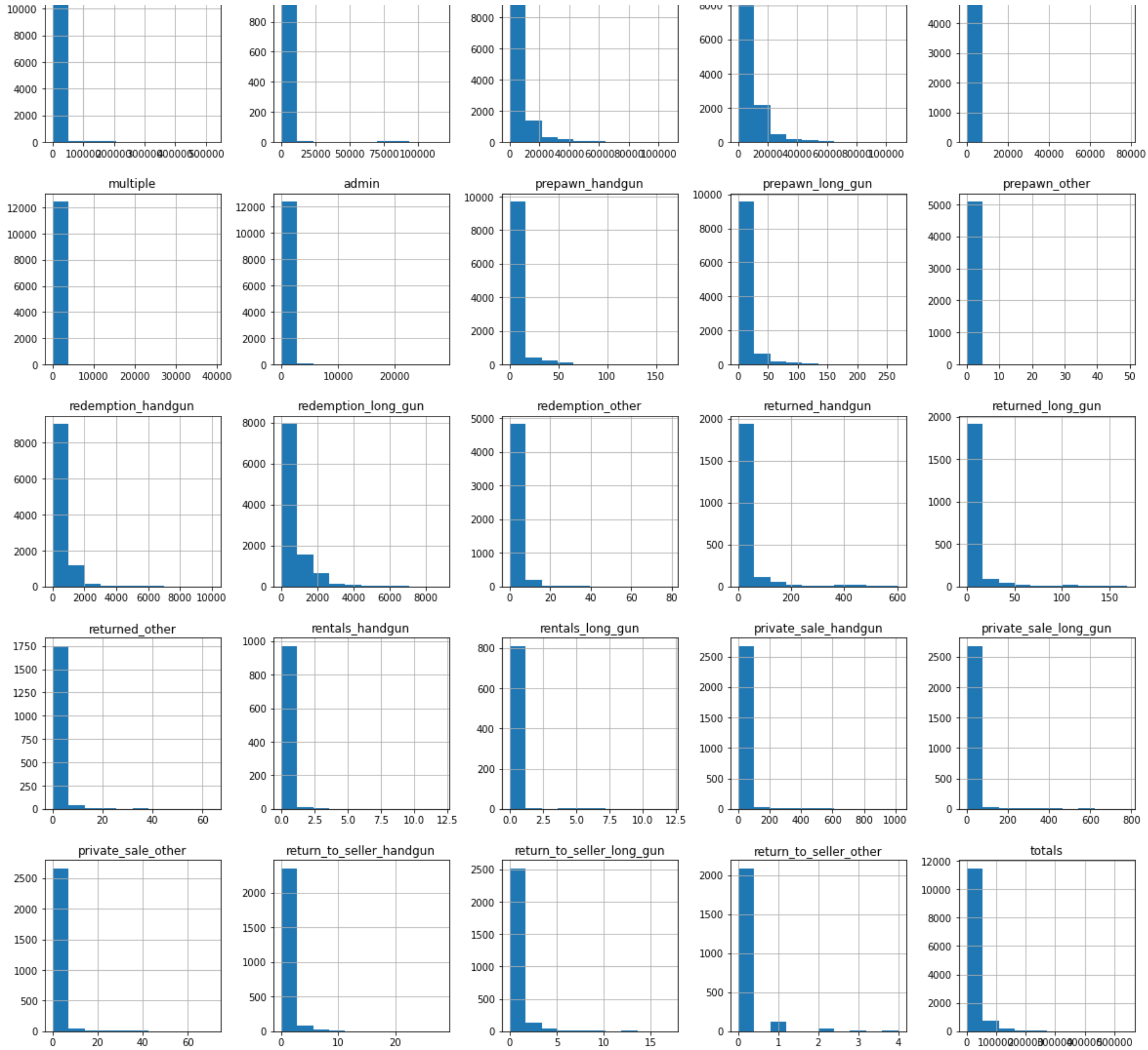
In [9]:

```
df_gun.hist(figsize=(20,20))
```

Out[9]:

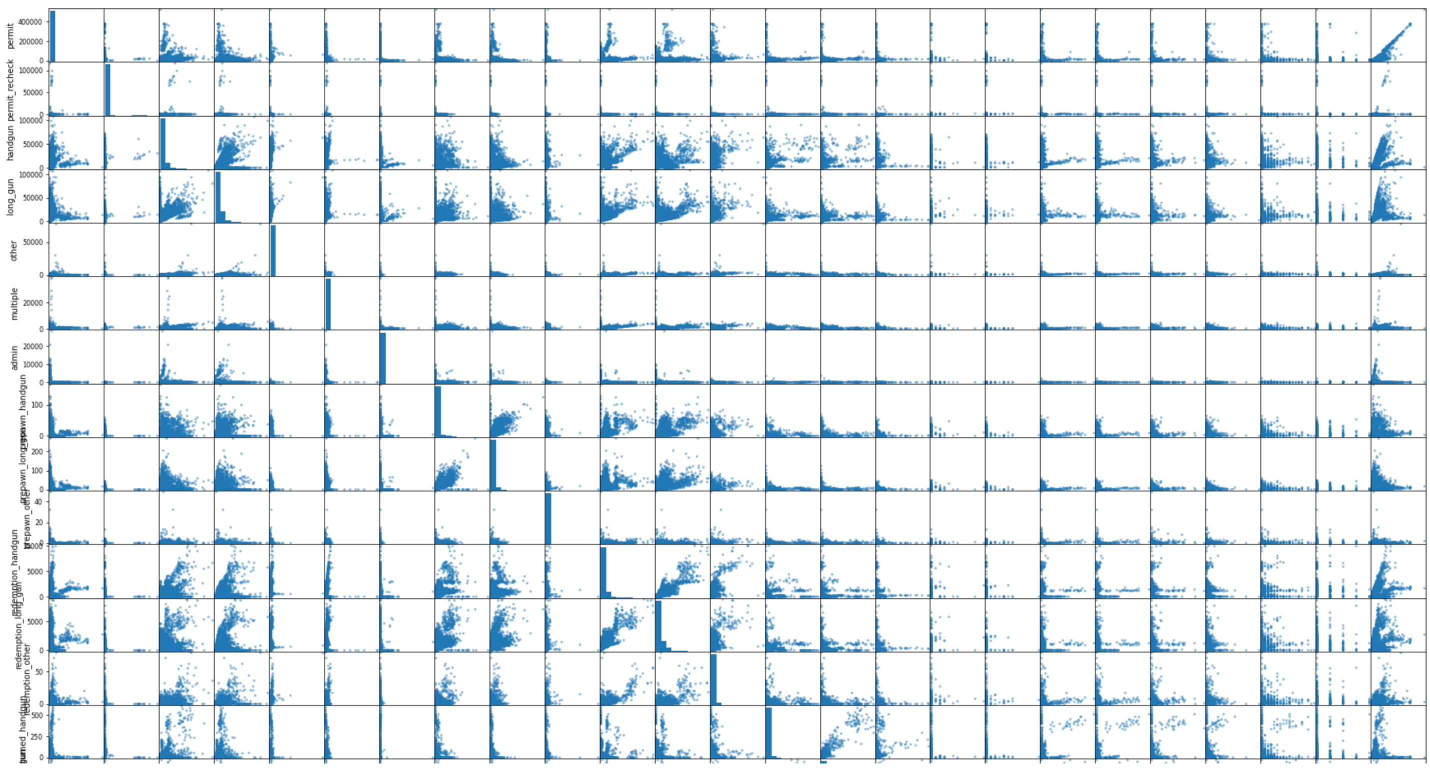
```
array([[<AxesSubplot:title={'center':'permit'}>,
        <AxesSubplot:title={'center':'permit_recheck'}>,
        <AxesSubplot:title={'center':'handgun'}>,
        <AxesSubplot:title={'center':'long_gun'}>,
        <AxesSubplot:title={'center':'other'}>],
       [<AxesSubplot:title={'center':'multiple'}>,
        <AxesSubplot:title={'center':'admin'}>,
        <AxesSubplot:title={'center':'prepawn_handgun'}>,
        <AxesSubplot:title={'center':'prepawn_long_gun'}>,
        <AxesSubplot:title={'center':'prepawn_other'}>],
       [<AxesSubplot:title={'center':'redemption_handgun'}>,
        <AxesSubplot:title={'center':'redemption_long_gun'}>,
        <AxesSubplot:title={'center':'redemption_other'}>,
        <AxesSubplot:title={'center':'returned_handgun'}>,
        <AxesSubplot:title={'center':'returned_long_gun'}>],
       [<AxesSubplot:title={'center':'returned_other'}>,
        <AxesSubplot:title={'center':'rentals_handgun'}>,
        <AxesSubplot:title={'center':'rentals_long_gun'}>,
        <AxesSubplot:title={'center':'private_sale_handgun'}>,
        <AxesSubplot:title={'center':'private_sale_long_gun'}>],
       [<AxesSubplot:title={'center':'private_sale_other'}>,
        <AxesSubplot:title={'center':'return_to_seller_handgun'}>,
        <AxesSubplot:title={'center':'return_to_seller_long_gun'}>,
        <AxesSubplot:title={'center':'return_to_seller_other'}>,
        <AxesSubplot:title={'center':'totals'}>]], dtype=object)
```

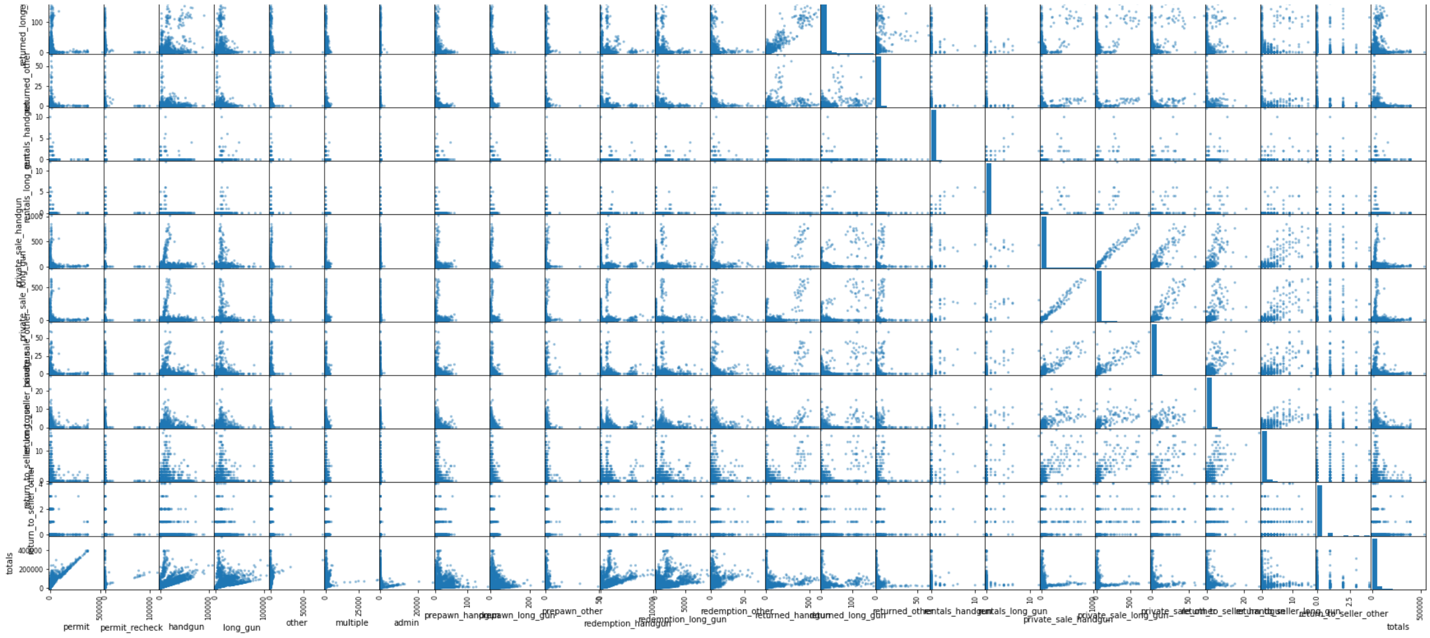




In [10]:

```
pd.plotting.scatter_matrix(df_gun, figsize=(30,30));
```





From the scatter plot we can expect the number of checks to rise throughout the years in this dataset.

In [11]:

```
print('Gun Data Duplicate Records', sum(df_gun.duplicated()))
print('Gun Data Empty Records', sum(df_gun.isna().any()))
print('Gun Data where (Total == 0) Records', (df_gun['totals'] == 0).sum())
df_gun.isna().sum()
```

```
Gun Data Duplicate Records 0
Gun Data Empty Records 23
Gun Data where (Total == 0) Records 265
```

Out[11]:

```
month                            0
state                            0
permit                          24
permit_recheck               11385
handgun                         20
long_gun                        19
other                         6985
multiple                         0
admin                           23
prepawn_handgun               1943
prepawn_long_gun              1945
prepawn_other                 7370
redemption_handgun            1940
redemption_long_gun           1941
redemption_other              7370
returned_handgun             10285
returned_long_gun            10340
returned_other               10670
rentals_handgun              11495
rentals_long_gun             11660
private_sale_handgun          9735
private_sale_long_gun         9735
private_sale_other            9735
return_to_seller_handgun     10010
return_to_seller_long_gun     9735
return_to_seller_other       10230
totals                           0
dtype: int64
```

In [12]:

```
df_gun[df_gun.isna()].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12485 entries, 0 to 12484
```

```
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   month                      0 non-null      object 
 1   state                      0 non-null      object 
 2   permit                     0 non-null      float64
 3   permit_recheck             0 non-null      float64
 4   handgun                    0 non-null      float64
 5   long_gun                   0 non-null      float64
 6   other                      0 non-null      float64
 7   multiple                   0 non-null      float64
 8   admin                      0 non-null      float64
 9   prepawn_handgun            0 non-null      float64
 10  prepawn_long_gun           0 non-null      float64
 11  prepawn_other              0 non-null      float64
 12  redemption_handgun         0 non-null      float64
 13  redemption_long_gun        0 non-null      float64
 14  redemption_other           0 non-null      float64
 15  returned_handgun           0 non-null      float64
 16  returned_long_gun          0 non-null      float64
 17  returned_other             0 non-null      float64
 18  rentals_handgun            0 non-null      float64
 19  rentals_long_gun           0 non-null      float64
 20  private_sale_handgun       0 non-null      float64
 21  private_sale_long_gun      0 non-null      float64
 22  private_sale_other         0 non-null      float64
 23  return_to_seller_handgun   0 non-null      float64
 24  return_to_seller_long_gun  0 non-null      float64
 25  return_to_seller_other     0 non-null      float64
 26  totals                     0 non-null      float64
dtypes: float64(25), object(2)
memory usage: 2.6+ MB
```

Data Cleaning

Data Cleaning activities for each DataFrame:-

In `df_census` dataframe:

- **Drop `Fact Note` column**
- **Fix numeric and percentage values**

In `df_gun` dataframe:

- **Remove missing rows**
- **Reorder columns**
- **Remove rows with `total == 0`**
- **Fix month column**

In [13]:

```
df_census_clean = df_census.copy()
df_gun_clean = df_gun.copy()
```

Define

- **Drop `Fact Note` column**

Code

In [14]:

```
df_census_clean.drop(['Fact Note'], axis=1, inplace=True)
df_census_clean.set_index('Fact', inplace=True)
df_census_clean = df_census_clean.T.reset_index()
```

```
df_census_clean.rename(columns={'index': 'state'}, inplace=True)
```

Test

In [15]:

```
df_census_clean.head()
```

Out[15]:

Fact	state	Population estimates, July 1, 2016, (V2016)	Population estimates base, April 1, 2010, (V2016)	Population, percent change - April 1, 2010 (estimates base) to July 1, 2016, (V2016)	Population, Census, April 1, 2010	Persons under 5 years, percent, July 1, 2016, (V2016)	Persons under 5 years, percent, April 1, 2010	Persons under 18 years, percent, July 1, 2016, (V2016)	Persons under 18 years, percent, April 1, 2010	Persons 65 years and over, percent, July 1, 2016, (V2016)	...	NaN	Valu Flag
0	Alabama	4,863,300	4,780,131	1.70%	4,779,736	6.00%	6.40%	22.60%	23.70%	16.10%	...	NaN	Na
1	Alaska	741,894	710,249	4.50%	710,231	7.30%	7.60%	25.20%	26.40%	10.40%	...	NaN	Na
2	Arizona	6,931,071	6,392,301	8.40%	6,392,017	6.30%	7.10%	23.50%	25.50%	16.90%	...	NaN	Na
3	Arkansas	2,988,248	2,916,025	2.50%	2,915,918	6.40%	6.80%	23.60%	24.40%	16.30%	...	NaN	Na
4	California	39,250,017	37,254,522	5.40%	37,253,956	6.30%	6.80%	23.20%	25.00%	13.60%	...	NaN	Na

5 rows × 86 columns

Define

- Fix numeric and percentage values

Code

In [16]:

```
df_census_clean = df_census_clean[df_census_clean.columns[:-20]]
numeric_columns = df_census_clean.columns[1:len(df_census_clean.columns)]
for col in numeric_columns:
    df_census_clean[col] = df_census_clean[col].replace(',', '', regex=True).replace(r'[
^\d.-]', r'', regex=True)
    df_census_clean[col] = pd.to_numeric(df_census_clean[col], downcast='float')
```

Test

In [17]:

```
df_census_clean.head()
```

Out[17]:

Fact	state	Population estimates, July 1, 2016, (V2016)	Population estimates base, April 1, 2010, (V2016)	Population, percent change - April 1, 2010 (estimates base) to July 1, 2016, (V2016)	Population, Census, April 1, 2010	Persons under 5 years, percent, July 1, 2016, (V2016)	Persons under 5 years, percent, April 1, 2010	Persons under 18 years, percent, July 1, 2016, (V2016)	Persons under 18 years, percent, April 1, 2010	Persons 65 years and over, percent, July 1, 2016, (V2016)	...	All
0	Alabama	4863300.0	4780131.0	1.7	4779736.0	6.0	6.4	22.600000	23.700001	16.100000	...	37

Fact	state	Population estimates, July 1, 2016, (V2016)	Population estimates base, April 1, 2010, (V2016)	Population, percent change - April 1, 2010 (estimates base) to July 1, 2016, (V2016)	Population, Census, April 1, 2010	Persons under 5 years, percent, July 1, 2016, (V2016)	Persons under 5 years, percent, April 1, 2010	Persons under 18 years, percent, July 1, 2016, (V2016)	Persons under 18 years, percent, April 1, 2010	Persons 65 years and over, percent, July 1, 2016, (V2016)	...	
1	Alaska	741894.0	710249.0	4.5	710231.0	7.3	7.6	25.200001	26.400000	10.400000	...	68
2	Arizona	6931071.0	6392301.0	8.4	6392017.0	6.3	7.1	23.500000	25.500000	16.900000	...	49
3	Arkansas	2988248.0	2916025.0	[illegible]	[illegible]	6.4	6.8	23.600000	24.400000	16.299999	...	[illegible]
4	California	39250016.0	37254520.0	5.4	37253956.0	6.3	6.8	23.200001	25.000000	13.600000	...	3548

5 rows × 66 columns

In [18]:

```
df_census_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 66 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   state
50 non-null     object
 1   Population estimates, July 1, 2016,  (V2016)
50 non-null     float32
 2   Population estimates base, April 1, 2010,  (V2016)
50 non-null     float32
 3   Population, percent change - April 1, 2010 (estimates base) to July 1, 2016,  (V2016
)   50 non-null     float32
 4   Population, Census, April 1, 2010
50 non-null     float32
 5   Persons under 5 years, percent, July 1, 2016,  (V2016)
50 non-null     float32
 6   Persons under 5 years, percent, April 1, 2010
50 non-null     float32
 7   Persons under 18 years, percent, July 1, 2016,  (V2016)
50 non-null     float32
 8   Persons under 18 years, percent, April 1, 2010
50 non-null     float32
 9   Persons 65 years and over, percent,  July 1, 2016,  (V2016)
50 non-null     float32
 10  Persons 65 years and over, percent, April 1, 2010
50 non-null     float32
 11  Female persons, percent,  July 1, 2016,  (V2016)
50 non-null     float32
 12  Female persons, percent, April 1, 2010
50 non-null     float32
 13  White alone, percent, July 1, 2016,  (V2016)
50 non-null     float32
 14  Black or African American alone, percent, July 1, 2016,  (V2016)
50 non-null     float32
 15  American Indian and Alaska Native alone, percent, July 1, 2016,  (V2016)
50 non-null     float32
 16  Asian alone, percent, July 1, 2016,  (V2016)
50 non-null     float32
 17  Native Hawaiian and Other Pacific Islander alone, percent, July 1, 2016,  (V2016)
46 non-null     float32
 18  Two or More Races, percent, July 1, 2016,  (V2016)
50 non-null     float32
 19  Hispanic or Latino, percent, July 1, 2016,  (V2016)
50 non-null     float32
 20  White alone, not Hispanic or Latino, percent, July 1, 2016,  (V2016)
50 non-null     float32
 21  Veterans, 2011-2015
50 non-null     float32
 22  Foreign born persons, percent, 2011-2015
50 non-null     float32
 23  Housing units,  July 1, 2016,  (V2016)
50 non-null     float32
 24  Housing units, April 1, 2010
50 non-null     float32
```

```
 25  Owner-occupied housing unit rate, 2011-2015
50 non-null     float32
 26  Median value of owner-occupied housing units, 2011-2015
50 non-null     float32
 27  Median selected monthly owner costs -with a mortgage, 2011-2015
50 non-null     float32
 28  Median selected monthly owner costs -without a mortgage, 2011-2015
50 non-null     float32
 29  Median gross rent, 2011-2015
50 non-null     float32
 30  Building permits, 2016
50 non-null     float32
 31  Households, 2011-2015
50 non-null     float32
 32  Persons per household, 2011-2015
50 non-null     float32
 33  Living in same house 1 year ago, percent of persons age 1 year+, 2011-2015
50 non-null     float32
 34  Language other than English spoken at home, percent of persons age 5 years+, 2011-20
15  50 non-null     float32
 35  High school graduate or higher, percent of persons age 25 years+, 2011-2015
50 non-null     float32
 36  Bachelor's degree or higher, percent of persons age 25 years+, 2011-2015
50 non-null     float32
 37  With a disability, under age 65 years, percent, 2011-2015
50 non-null     float32
 38  Persons  without health insurance, under age 65 years, percent
50 non-null     float32
 39  In civilian labor force, total, percent of population age 16 years+, 2011-2015
50 non-null     float32
 40  In civilian labor force, female, percent of population age 16 years+, 2011-2015
50 non-null     float32
 41  Total accommodation and food services sales, 2012 ($1,000)
50 non-null     float32
 42  Total health care and social assistance receipts/revenue, 2012 ($1,000)
50 non-null     float32
 43  Total manufacturers shipments, 2012 ($1,000)
48 non-null     float32
 44  Total merchant wholesaler sales, 2012 ($1,000)
50 non-null     float32
 45  Total retail sales, 2012 ($1,000)
50 non-null     float32
 46  Total retail sales per capita, 2012
50 non-null     float32
 47  Mean travel time to work (minutes), workers age 16 years+, 2011-2015
50 non-null     float32
 48  Median household income (in 2015 dollars), 2011-2015
50 non-null     float32
 49  Per capita income in past 12 months (in 2015 dollars), 2011-2015
50 non-null     float32
 50  Persons in poverty, percent
50 non-null     float32
 51  Total employer establishments, 2015
50 non-null     float32
 52  Total employment, 2015
50 non-null     float32
 53  Total annual payroll, 2015 ($1,000)
50 non-null     float32
 54  Total employment, percent change, 2014-2015
49 non-null     float32
 55  Total nonemployer establishments, 2015
50 non-null     float32
 56  All firms, 2012
50 non-null     float32
 57  Men-owned firms, 2012
50 non-null     float32
 58  Women-owned firms, 2012
50 non-null     float32
 59  Minority-owned firms, 2012
50 non-null     float32
 60  Nonminority-owned firms, 2012
50 non-null     float32
```

```
 61  Veteran-owned firms, 2012
50 non-null     float32
 62  Nonveteran-owned firms, 2012
50 non-null     float32
 63  Population per square mile, 2010
50 non-null     float32
 64  Land area in square miles, 2010
50 non-null     float32
 65  FIPS Code
50 non-null     float32
dtypes: float32(65), object(1)
memory usage: 13.2+ KB
```

In [19]:

```python
print('Census Data Duplicate Records', sum(df_census_clean.duplicated()))
print('Census Data Empty Records', sum(df_census_clean.isna().any()))
df_census_clean.isna().sum()
```

```
Census Data Duplicate Records 0
Census Data Empty Records 3
```

Out[19]:

```
Fact
state
0
Population estimates, July 1, 2016,  (V2016)
0
Population estimates base, April 1, 2010,  (V2016)
0
Population, percent change - April 1, 2010 (estimates base) to July 1, 2016,  (V2016)
0
Population, Census, April 1, 2010
0

..
Veteran-owned firms, 2012
0
Nonveteran-owned firms, 2012
0
Population per square mile, 2010
0
Land area in square miles, 2010
0
FIPS Code
0
Length: 66, dtype: int64
```

Define

- **Remove missing rows**

Code

In [20]:

```python
df_gun_clean.dropna(inplace=True)
```

Test

In [21]:

```python
print('Gun Data Duplicate Records', sum(df_gun_clean.duplicated()))
print('Gun Data Empty Records', sum(df_gun_clean.isna().any()))
df_gun_clean.isna().sum()
```

```
Gun Data Duplicate Records 0
Gun Data Empty Records 0
```

Gun Data Empty Records 0

Out[21]:

```
month                          0
state                          0
permit                         0
permit_recheck                 0
handgun                        0
long_gun                       0
other                          0
multiple                       0
admin                          0
prepawn_handgun                0
prepawn_long_gun               0
prepawn_other                  0
redemption_handgun             0
redemption_long_gun            0
redemption_other               0
returned_handgun               0
returned_long_gun              0
returned_other                 0
rentals_handgun                0
rentals_long_gun               0
private_sale_handgun           0
private_sale_long_gun          0
private_sale_other             0
return_to_seller_handgun       0
return_to_seller_long_gun      0
return_to_seller_other         0
totals                         0
dtype: int64
```

In [22]:

```
df_gun_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 770 entries, 0 to 769
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   month                      770 non-null    object 
 1   state                      770 non-null    object 
 2   permit                     770 non-null    float64
 3   permit_recheck             770 non-null    float64
 4   handgun                    770 non-null    float64
 5   long_gun                   770 non-null    float64
 6   other                      770 non-null    float64
 7   multiple                   770 non-null    int64  
 8   admin                      770 non-null    float64
 9   prepawn_handgun            770 non-null    float64
 10  prepawn_long_gun           770 non-null    float64
 11  prepawn_other              770 non-null    float64
 12  redemption_handgun         770 non-null    float64
 13  redemption_long_gun        770 non-null    float64
 14  redemption_other           770 non-null    float64
 15  returned_handgun           770 non-null    float64
 16  returned_long_gun          770 non-null    float64
 17  returned_other             770 non-null    float64
 18  rentals_handgun            770 non-null    float64
 19  rentals_long_gun           770 non-null    float64
 20  private_sale_handgun       770 non-null    float64
 21  private_sale_long_gun      770 non-null    float64
 22  private_sale_other         770 non-null    float64
 23  return_to_seller_handgun   770 non-null    float64
 24  return_to_seller_long_gun  770 non-null    float64
 25  return_to_seller_other     770 non-null    float64
 26  totals                     770 non-null    int64  
dtypes: float64(23), int64(2), object(2)
memory usage: 168.4+ KB
```

In [23]:

```
df_gun_clean.head()
```

Out[23]:

	month	state	permit	permit_recheck	handgun	long_gun	other	multiple	admin	prepawn_handgun	...	returned_ot
0	2017-09	Alabama	16717.0	0.0	5734.0	6320.0	221.0	317	0.0	15.0	...	
1	2017-09	Alaska	209.0	2.0	2320.0	2930.0	219.0	160	0.0	5.0	...	
2	2017-09	Arizona	5069.0	382.0	11063.0	7946.0	920.0	631	0.0	13.0	...	
3	2017-09	Arkansas	2935.0	632.0	4347.0	6063.0	165.0	366	51.0	12.0	...	
4	2017-09	California	57839.0	0.0	37165.0	24581.0	2984.0	0	0.0	0.0	...	

5 rows × 27 columns

Define

- Fix month column

Code

In [24]:

```
def split_and_get(m, sep, idx):
    return m.split(sep)[idx]
```

In [25]:

```
df_gun_clean.drop(df_gun_clean[df_gun_clean['totals'] == 0].index, inplace=True)
df_gun_clean['year'] = df_gun_clean['month'].apply(lambda m: split_and_get(m, "-", 0)).a
stype(int)
df_gun_clean['month'] = df_gun_clean['month'].apply(lambda m: split_and_get(m, "-", 1)).
astype(int)
```

Test

In [26]:

```
df_gun_clean.head()
```

Out[26]:

	month	state	permit	permit_recheck	handgun	long_gun	other	multiple	admin	prepawn_handgun	...	rentals_har
0	9	Alabama	16717.0	0.0	5734.0	6320.0	221.0	317	0.0	15.0	...	
1	9	Alaska	209.0	2.0	2320.0	2930.0	219.0	160	0.0	5.0	...	
2	9	Arizona	5069.0	382.0	11063.0	7946.0	920.0	631	0.0	13.0	...	
3	9	Arkansas	2935.0	632.0	4347.0	6063.0	165.0	366	51.0	12.0	...	
4	9	California	57839.0	0.0	37165.0	24581.0	2984.0	0	0.0	0.0	...	

5 rows × 28 columns

Define

- **Reorder columns**

Code

In [27]:

```
gun_cols = ['year', 'month', 'state', 'permit', 'permit_recheck', 'handgun', 'long_gun',
        'other', 'multiple', 'admin', 'prepawn_handgun', 'prepawn_long_gun',
        'prepawn_other', 'redemption_handgun', 'redemption_long_gun',
        'redemption_other', 'returned_handgun', 'returned_long_gun',
        'returned_other', 'rentals_handgun', 'rentals_long_gun',
        'private_sale_handgun', 'private_sale_long_gun', 'private_sale_other',
        'return_to_seller_handgun', 'return_to_seller_long_gun',
        'return_to_seller_other', 'totals']
df_gun_clean = df_gun_clean[gun_cols]
```

Test

In [28]:

```
df_gun_clean.head()
```

Out[28]:

	year	month	state	permit	permit_recheck	handgun	long_gun	other	multiple	admin	...	returned_other	rentals_l
0	2017	9	Alabama	16717.0	0.0	5734.0	6320.0	221.0	317	0.0	...	0.0	
1	2017	9	Alaska	209.0	2.0	2320.0	2930.0	219.0	160	0.0	...	0.0	
2	2017	9	Arizona	5069.0	382.0	11063.0	7946.0	920.0	631	0.0	...	0.0	
3	2017	9	Arkansas	2935.0	632.0	4347.0	6063.0	165.0	366	51.0	...	0.0	
4	2017	9	California	57839.0	0.0	37165.0	24581.0	2984.0	0	0.0	...	0.0	

5 rows × 28 columns

Define

- **Remove rows with** `total == 0`

Code

In [29]:

```
df_gun_clean.drop(df_gun_clean[df_gun_clean['totals'] == 0].index, inplace=True)
```

Test

In [30]:

```
df_gun_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 768 entries, 0 to 769
Data columns (total 28 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   year                       768 non-null    int64
 1   month                      768 non-null    int64
 2   state                      768 non-null    object
 3   permit                     768 non-null    float64
 4   permit_recheck             768 non-null    float64
 5   handgun                    768 non-null    float64
```

```
 6   long_gun                   768 non-null    float64
 7   other                      768 non-null    float64
 8   multiple                   768 non-null    int64
 9   admin                      768 non-null    float64
 10  prepawn_handgun            768 non-null    float64
 11  prepawn_long_gun           768 non-null    float64
 12  prepawn_other              768 non-null    float64
 13  redemption_handgun         768 non-null    float64
 14  redemption_long_gun        768 non-null    float64
 15  redemption_other           768 non-null    float64
 16  returned_handgun           768 non-null    float64
 17  returned_long_gun          768 non-null    float64
 18  returned_other             768 non-null    float64
 19  rentals_handgun            768 non-null    float64
 20  rentals_long_gun           768 non-null    float64
 21  private_sale_handgun       768 non-null    float64
 22  private_sale_long_gun      768 non-null    float64
 23  private_sale_other         768 non-null    float64
 24  return_to_seller_handgun   768 non-null    float64
 25  return_to_seller_long_gun  768 non-null    float64
 26  return_to_seller_other     768 non-null    float64
 27  totals                     768 non-null    int64
dtypes: float64(23), int64(4), object(1)
memory usage: 174.0+ KB
```

Exploratory Data Analysis

Research Question 1 (What census data is most associated with high gun per capita in 2010 and 2016?)

Relation between population estimates and gun sales in both years.

Aggregating gun data by state for year of 2016, then joining it with population data of the same year.

In [31]:

```
df_population_16 = df_census_clean[['state', 'Population estimates, July 1, 2016,  (V201
6)']]
df_gun_total_at_16 = df_gun_clean[df_gun_clean['year'] == 2016][['year', 'state', 'total
s']]
df_gun_total_at_16 = df_gun_total_at_16.groupby('state')['totals'].sum()
df_gun_total_at_16 = pd.DataFrame(df_gun_total_at_16)
df_gun_census_16 = df_population_16.merge(df_gun_total_at_16, on='state', how='inner')
df_gun_census_16.head()
```

Out[31]:

	state	Population estimates, July 1, 2016, (V2016)	totals
0	Alabama	4863300.0	239533
1	Alaska	741894.0	40744
2	Arizona	6931071.0	185555
3	Arkansas	2988248.0	120054
4	California	39250016.0	1039015

Aggregating gun data by state for year of 2010, then joining it with population data of the same year.

In [32]:

```
df_population_10 = df_census_clean[['state', 'Population estimates base, April 1, 2010,
(V2016)']]
df_gun_total_at_10 = df_gun_clean[df_gun_clean['year'] == 2010][['year', 'state', 'total
s']]
```

```
df_gun_total_at_10 = df_gun_total_at_10.groupby('state')['totals'].sum()
df_gun_total_at_10 = pd.DataFrame(df_gun_total_at_10)
df_gun_census_10 = df_population_10.merge(df_gun_total_at_16, on='state', how='inner')
df_gun_census_10.head()
```

Out[32]:

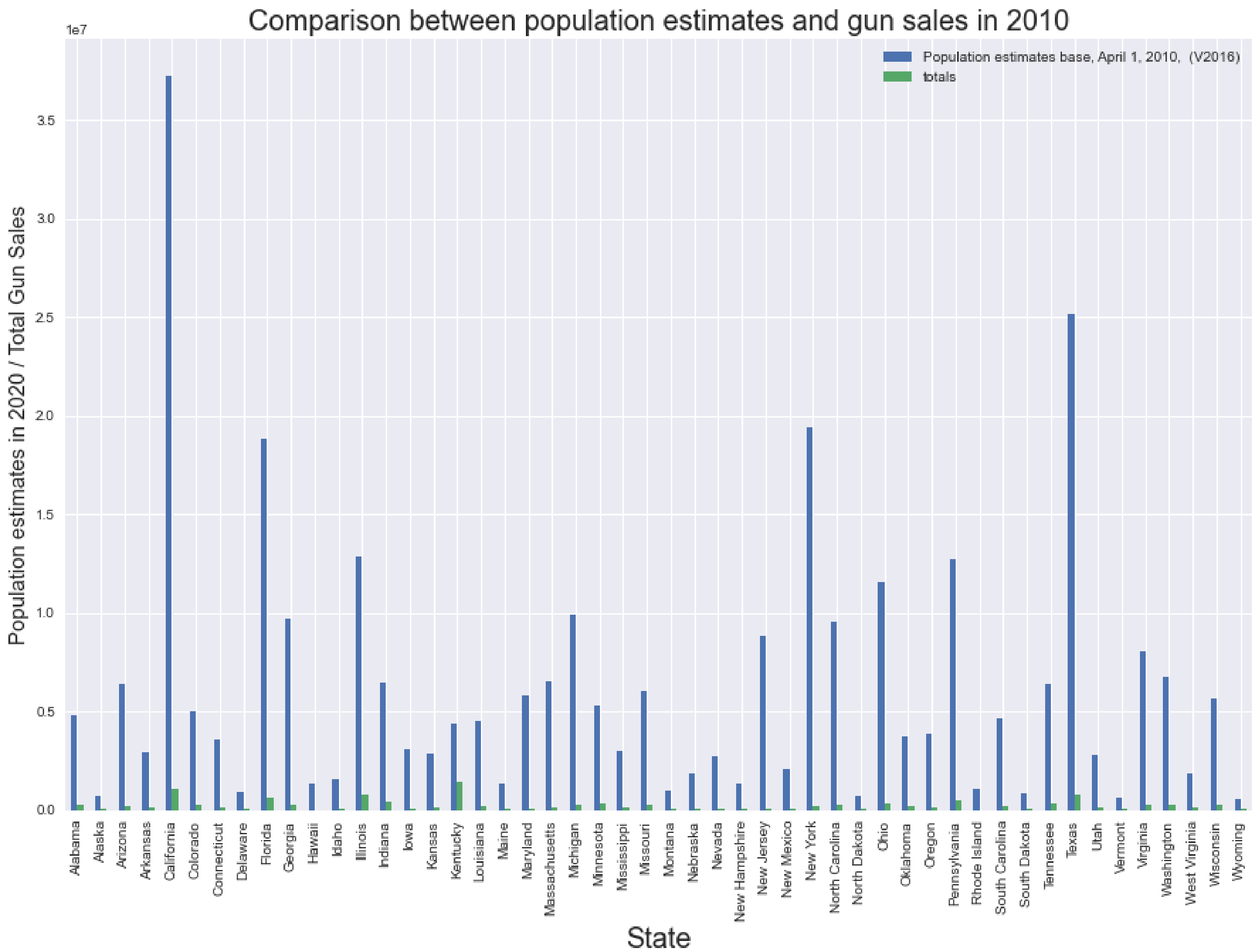
	state	Population estimates base, April 1, 2010, (V2016)	totals
0	Alabama	4780131.0	239533
1	Alaska	710249.0	40744
2	Arizona	6392301.0	185555
3	Arkansas	2916025.0	120054
4	California	37254520.0	1039015

In [33]:

```
plt.style.use('seaborn')
df_gun_census_10.plot(x='state', y=['Population estimates base, April 1, 2010,  (V2016)'
, 'totals'], kind='bar', figsize=(15, 10))
plt.xlabel("State", fontsize=20)
plt.ylabel("Population estimates in 2020 / Total Gun Sales", fontsize=15)
plt.title("Comparison between population estimates and gun sales in 2010", fontsize=20)
```

Out[33]:

```
Text(0.5, 1.0, 'Comparison between population estimates and gun sales in 2010')
```



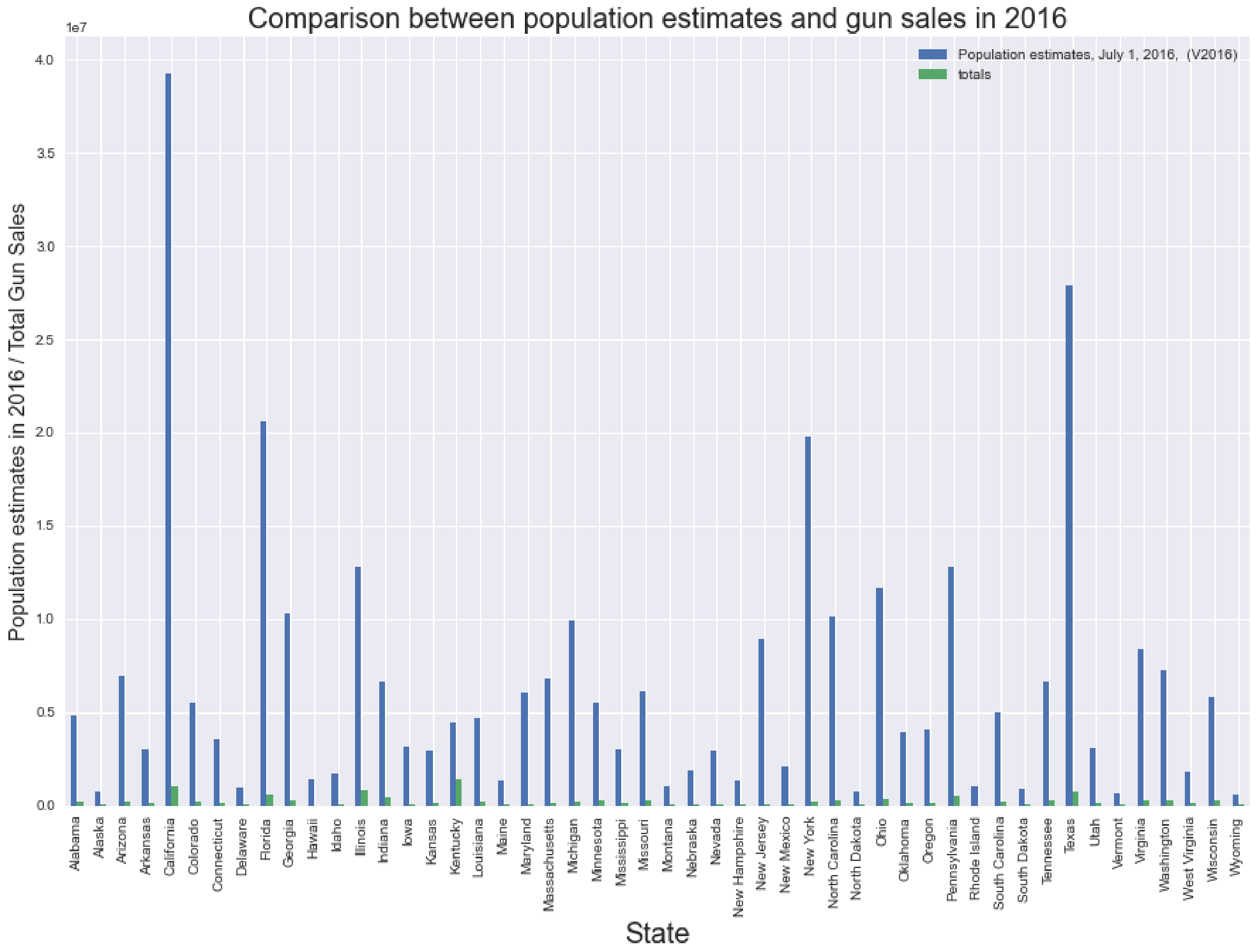
Assuming Gun sales refers to an approximation of Background Checks, Kentucky has the highest ratio of population to background checks in 2010.

In [34]:

```
plt.style.use('seaborn')
df_gun_census_16.plot(x='state', y=['Population estimates, July 1, 2016,  (V2016)', 'tot
als'], kind='bar', figsize=(15, 10))
plt.xlabel("State", fontsize=20)
plt.ylabel("Population estimates in 2016 / Total Gun Sales", fontsize=15)
plt.title("Comparison between population estimates and gun sales in 2016", fontsize=20)
```

Out[34]:

```
Text(0.5, 1.0, 'Comparison between population estimates and gun sales in 2016')
```



Assuming Gun sales refers to an approximation of Background Checks, Kentucky has the highest ratio of population to background checks in 2016.

Research Question 2 (What is the overall trend of gun purchases?)

Using the totals from the NICS data, we can see what the overall trend of gun sales are from 1998 to 2018

In [35]:

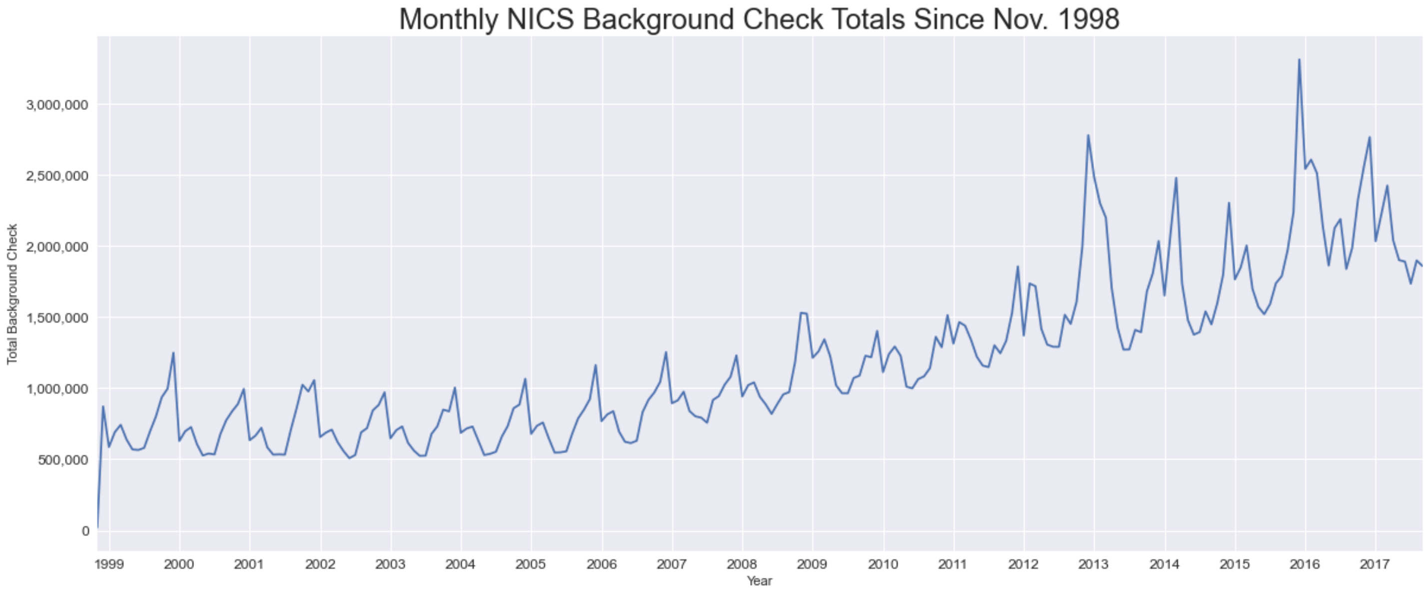
```
totals = df_gun.groupby("month")["totals"].sum()

tick_placement = pd.np.arange(2, len(totals), 12)
plt.style.use('seaborn')
ax = totals.plot(figsize=(20,8))

ax.set_title("Monthly NICS Background Check Totals Since Nov. 1998", fontsize=24)
ax.set_yticklabels([ "{0:,.0f}".format(y) for y in ax.get_yticks() ], fontsize=12)
plt.setp(ax.get_xticklabels(), rotation=0, fontsize=12)
ax.set_xticks(tick_placement)
ax.set_xticklabels([ totals.index[i].split("-")[0] for i in tick_placement ])
```

```
ax.set_xlim(0, len(totals) - 1)
ax.set_xlabel("Year")
ax.set_ylabel("Total Background Check")
```

```
/var/folders/hy/f6clqs_925597rrs_wgvh0q40000gp/T/ipykernel_71043/2429522092.py:3: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead.
  tick_placement = pd.np.arange(2, len(totals), 12)
/var/folders/hy/f6clqs_925597rrs_wgvh0q40000gp/T/ipykernel_71043/2429522092.py:8: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_yticklabels([ "{0:,.0f}".format(y) for y in ax.get_yticks() ], fontsize=12)
```

Out[35]:

```
Text(0, 0.5, 'Total Background Check')
```



This visualization shows an exponential increase in background checks since 1998. Each spike shows that gun sales greatly increase in December of each year. The greatest spike being in 2015 due to Black Friday sales.

Research Question 3 (How many total checks have there been in each state since 1998?)

For this visualization, I grouped all of the states together and summed up the total checks per state since 1998

In [36]:

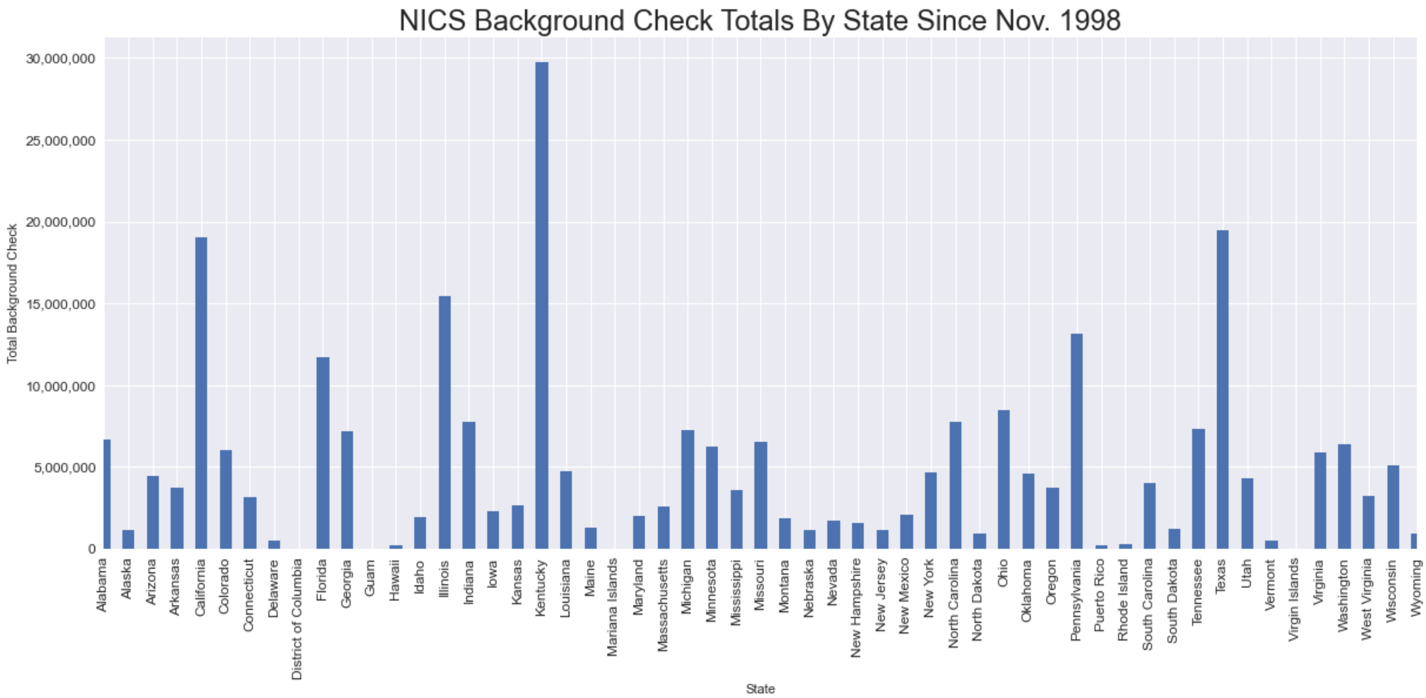
```
# get the total checks by each state and each month
checks_by_state = df_gun.groupby(['state', 'month'])['totals'].sum().reset_index()

# group the states and sum the totals
state_totals = checks_by_state.groupby('state')['totals'].sum()

# plot graph
state_total_tick_placement = pd.np.arange(len(state_totals))
plt.style.use('seaborn')
state_ax = state_totals.plot(kind='bar',figsize=(20,8))

state_ax.set_title("NICS Background Check Totals By State Since Nov. 1998", fontsize=24)
state_ax.set_yticklabels([ "{0:,.0f}".format(y) for y in state_ax.get_yticks() ], fontsize=12);
plt.setp(state_ax.get_xticklabels(), fontsize=12)
state_ax.set_xticks(state_total_tick_placement)
state_ax.set_xticklabels(state_totals.index)
state_ax.set_xlim(0, len(state_totals) - 1)
state_ax.set_xlabel("State")
state_ax.set_ylabel("Total Background Check")
```

```
/var/folders/hy/f6clqs_925597rrs_wgvh0q40000gp/T/ipykernel_71043/2242814138.py:8: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead.
```

```
ision: import numpy directly instead.
  state_total_tick_placement = pd.np.arange(len(state_totals))
/var/folders/hy/f6clqs_925597rrs_wgvh0q40000gp/T/ipykernel_71043/2242814138.py:13: UserWa
rning: FixedFormatter should only be used together with FixedLocator
  state_ax.set_yticklabels([ "{0:,.0f}".format(y) for y in state_ax.get_yticks() ], fonts
ize=12);
```

Out[36]:

```
Text(0, 0.5, 'Total Background Check')
```



As you can see in this graph, Kentucky has the most activity in background checks for guns since 1998. The state is known to have the least restrictive gun control laws compared to other states.

Conclusions

Limitation: **In the Gun Data: Only 2016 and 2017 records are available. Census data are restricted on population in 2010 and 2016.**

This analysis allowed me to see the bigger picture when it comes to guns in America. Even though I didn't calculate actual gun sales, the data still allowed me to see trends between each state and all over the U.S.

- **NICS Background check activity has steadily risen since 1998**

  > **The spikes in December likely due to Black Friday sales.**

- **Kentucky has the highest amount of background checks since 1998**

  > **Kentucky has some of the least restrictive gun control compared to other states**

References

- [NICS Fire Alarm Checks](#)

In [36]: