Project: Investigating FBI Gun Data

Table of Contents

Introduction

> **Key Notes**: "The data used in this notebook comes from the FBI's National Instant Criminal Background Check System. The NICS is used 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 on sheet1 of the excel file (.xlsx) gun_data.
>
> - The FBI Gun data inclues different background check types, such as; 'permit', 'permit_recheck', 'handgun', 'long_gun', etc... Data is collected at a month, state level for each variable in the dataset. For this analysis we will be using the total count of all background check types (all variables) at the month, state level.

> 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."

Questions to explore:

> In this analysis we will be exploring purchasing trends in the united states. The main questions we want to answer are;
>
> 1. Does Firearm shopping have a seasonal pattern?
> 2. What is the avg background checks completed by month? And what is the month with the greatest avg?
> 3. Which states had the highest growth in gun registrations?

In [1]:
```
# Use this cell to set up import statements for all of the packages that you
#   plan to use.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style('darkgrid')
```

Data Wrangling

> **Note**: In this section of the report, the following work will be done;
>
> - Load in the data
> - Check for cleanliness
> - Trim and clean the dataset for analysis

General Properties

In [2]:
```
# Loading Data
df_c = pd.read_csv('US_Census_Data.csv')
df_g = pd.read_excel('gun_data.xlsx')
```

In [3]:
```
# Exploring the gun data from the excel file
df_g.head(3)
```

Out[3]:

	month	state	permit	permit_recheck	handgun	long_gun	other	multiple	admin	prepawn
0	2017-09	Alabama	16717.0	0.0	5734.0	6320.0	221.0	317	0.0	
1	2017-09	Alaska	209.0	2.0	2320.0	2930.0	219.0	160	0.0	
2	2017-09	Arizona	5069.0	382.0	11063.0	7946.0	920.0	631	0.0	

3 rows × 27 columns

In [4]:
```
# Providing the dimensions of the dataframe to understand size of data
df_g.shape, df_c.shape
```

Out[4]: ((12485, 27), (85, 52))

In [5]:
```
df_g.dtypes
```

Out[5]:
```
month                          object
state                          object
permit                        float64
permit_recheck                float64
handgun                       float64
long_gun                      float64
other                         float64
multiple                        int64
admin                         float64
prepawn_handgun               float64
prepawn_long_gun              float64
prepawn_other                 float64
redemption_handgun            float64
redemption_long_gun           float64
redemption_other              float64
returned_handgun              float64
returned_long_gun             float64
returned_other                float64
rentals_handgun               float64
rentals_long_gun              float64
private_sale_handgun          float64
private_sale_long_gun         float64
private_sale_other            float64
return_to_seller_handgun      float64
return_to_seller_long_gun     float64
return_to_seller_other        float64
totals                          int64
dtype: object
```

In [6]:
```
df_g.describe()
```

Out[6]:

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

8 rows × 25 columns

In [7]:
```
df_g.head()
```

Out[7]:

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

5 rows × 27 columns

In [8]:
```
# Exploring the census data from the csv
df_c.head(3)
```

Out[8]:

	Fact	Fact Note	Alabama	Alaska	Arizona	Arkansas	California	Colorado	Connecticut	D
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	
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	
2	Population, percent change - April 1, 2010 (es...	NaN	1.70%	4.50%	8.40%	2.50%	5.40%	10.20%	0.10%	

3 rows × 52 columns

In [9]:
```
df_c.dtypes.head()
```

Out[9]:
```
Fact          object
Fact Note     object
Alabama       object
Alaska        object
Arizona       object
dtype: object
```

Checking if both data sets have the same states

In [10]:
```
# Count of states in the census data
census_state = df_c.iloc[0,2:].index
len(census_state)
#census_state
```

Out[10]:
```
50
```

In [11]:
```
# Count of states in the gun data
gun_state = df_g.groupby('state').sum().index
len(gun_state)
#gun_state
```

Out[11]:
```
55
```

In [12]:
```
# Checking to see which states are missing
for s in gun_state:
    if s not in census_state:
        print(s)
```

```
District of Columbia
Guam
Mariana Islands
Puerto Rico
Virgin Islands
```

Determining if either data sets have duplicates

In [13]:
```
# check for duplicates in the data.
print("Number of duplicates in gun data is", sum(df_g.duplicated()))
print("Number of duplicates in census data is", sum(df_c.duplicated()))
```

```
Number of duplicates in gun data is 0
Number of duplicates in census data is 3
```

Checking for null values

In [14]:
```
#  Checking to see if any value is NaN in the DataFrame and in how many columns
print(df_g.isnull().any().any(), sum(df_g.isnull().any()))
```

```
True 23
```

In [15]:
```
#  Checking to see if any value is NaN in the DataFrame and if so, then how many
print(df_c.isnull().any().any(), sum(df_c.isnull().any()))
```

```
True 52
```

In [16]:
```
df_c.describe()
```

Out[16]:

	Fact	Fact Note	Alabama	Alaska	Arizona	Arkansas	California	Colorado	Connecticut	Delav
count	80	28	65	65	65	65	65	65	65	
unique	80	15	65	64	64	64	63	64	63	
top	FIPS Code	(c)	68.70%	7.30%	50.30%	50.90%	6.80%	3.30%	0.10%	51.
freq	1	6	1	2	2	2	2	2	2	

4 rows × 52 columns

Data Cleaning

Census Data

- Since the "Fact Note" column is mostly null and adds grouping challenges, it can be removed for cleaner alaysis
- For conveience, all column names will be converted to lowercase
- Converting datatypes from string to float for data in the columns labled as names of states.
- Dropping duplicate rows

In [17]:
```
# Dropping the column "Fact Note"

df_c = df_c.drop('Fact Note',axis=1)
```

In [18]:
```
# For convenience all column names will be converted to lowercase

df_c.rename(columns = lambda x: x.lower(), inplace = True)
```

In [19]:
```
# Converting datatypes from string to float for data in the columns labled as nar

columns = df_c.iloc[:,1:].columns
for column in columns:
    df_c[column] = df_c[column].str.extract('(\d+)').astype(float)
```

In [20]:
```
print(df_c.iloc[:,1:].isnull().any().any(), sum(df_c.iloc[:,1:].isnull().any()))
```

```
True 50
```

In [21]:
```
# The last 20 rows of the cencus data is unecessary to have.
# therefore I am going to drop them to make the data set cleaner
df_c[65:86]
```

Out[21]:

	fact	alabama	alaska	arizona	arkansas	california	colorado	connecticut	delaware	fl
65	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
66	NOTE: FIPS Code values are enclosed in quotes ...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
67	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
68	Value Notes	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
69	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
70	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [22]:
```
# Dropping last 20 rows
df_c = df_c.drop(df_c.index[65:86])
```

In [23]:
```
# Filling the remaining null values with the float value 0.0
df_c = df_c.iloc[:,:].fillna(0.0)
```

In [24]:
```
# drop duplicates and confirm changes

df_c.drop_duplicates(inplace=True)
sum(df_c.duplicated())
```

Out[24]: 0

In [25]:
```
# Checking census data final result
df_c.head(3)
```

Out[25]:

	fact	alabama	alaska	arizona	arkansas	california	colorado	connecticut	delaware	flor
0	Population estimates, July 1, 2016, (V2016)	4.0	741.0	6.0	2.0	39.0	5.0	3.0	952.0	2
1	Population estimates base, April 1, 2010, (V2...	4.0	710.0	6.0	2.0	37.0	5.0	3.0	897.0	1
2	Population, percent change - April 1, 2010 (es...	1.0	4.0	8.0	2.0	5.0	10.0	0.0	6.0	

3 rows × 51 columns

Gun Data

- Converting the datatype from string to datetime for column "month"

In [26]:
```
# Converting month from string to DateTime
df_g['month'] = pd.to_datetime(df_g['month'])
df_g['month'].head()
```

Out[26]:
```
0   2017-09-01
1   2017-09-01
2   2017-09-01
3   2017-09-01
4   2017-09-01
Name: month, dtype: datetime64[ns]
```

In [27]:
```
# For convenience all column names will be converted to lowercase
df_g.rename(columns = lambda x: x.lower(), inplace = True)
```

In [28]:
```
# Dropping the states that aren't in both samples
states_to_drop = ['District of Columbia','Guam','Mariana Islands','Puerto Rico',

df_g = df_g.loc[~df_g['state'].isin(states_to_drop)].reset_index(drop = True)
df_g
```

Out[28]:

	month	state	permit	permit_recheck	handgun	long_gun	other	multiple	a
0	2017-09-01	Alabama	16717.0	0.0	5734.0	6320.0	221.0	317	
1	2017-09-01	Alaska	209.0	2.0	2320.0	2930.0	219.0	160	
2	2017-09-01	Arizona	5069.0	382.0	11063.0	7946.0	920.0	631	
3	2017-09-01	Arkansas	2935.0	632.0	4347.0	6063.0	165.0	366	
4	2017-09-01	California	57839.0	0.0	37165.0	24581.0	2984.0	0	
5	2017-09-01	Colorado	4356.0	0.0	15751.0	13448.0	1007.0	1062	
6	2017-09-01	Connecticut	4343.0	673.0	4834.0	1993.0	274.0	0	

Exploratory Data Analysis

Research Question 1: Does Firearm shopping have a seasonal pattern?

Let's take a look at the seasonal trend by viewing the number of background checks by month on a line graph

Staging the data:

In [29]:
```
# Renaming 'month' column to 'ym' and creating two additional columns for conven
df_g.rename(columns={'month': 'ym'}, inplace=True)
df_g['year'] = pd.DatetimeIndex(df_g['ym']).year
df_g['months'] = pd.DatetimeIndex(df_g['ym']).month
df_g[df_g['year']==2012]
```

2875	2012-12-01	Montana	1285.0	NaN	6287.0	8919.0	187.0	412
2876	2012-12-01	Nebraska	8300.0	NaN	224.0	6715.0	30.0	10
2877	2012-12-01	Nevada	2053.0	NaN	9274.0	9429.0	450.0	749
2878	2012-12-01	New Hampshire	3722.0	NaN	8412.0	7366.0	135.0	2
2879	2012-12-01	New Jersey	0.0	NaN	3825.0	5947.0	275.0	0
...	...	...	...	...	...	...	...	...
3420	2012-01-01	Massachusetts	7171.0	NaN	4008.0	2157.0	128.0	133
3421	2012-01-01	Michigan	19146.0	NaN	1224.0	8653.0	166.0	39
	2012-							

In [30]:
```
df_gun_totals = df_g[['ym','totals']]
df_gun_totals.set_index('ym',inplace=True)
df_gun_totals = df_gun_totals[::-1]

gun_totals_by_ym = df_gun_totals.groupby('ym').sum()
gun_totals_by_ym.head(3)
```

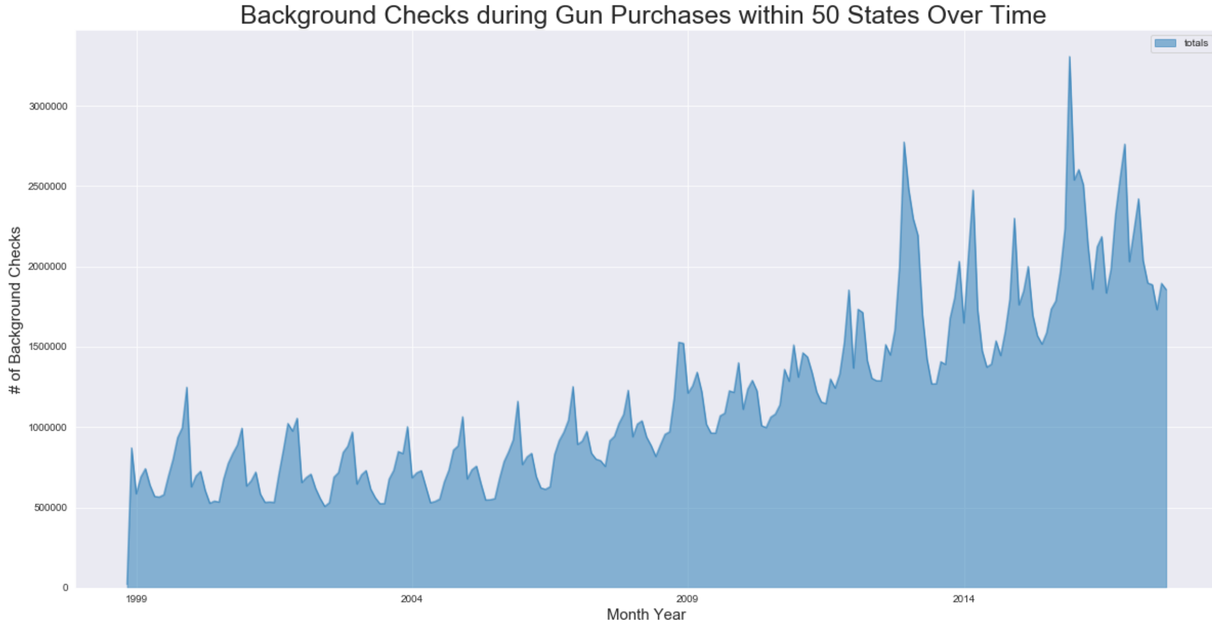
Out[30]:

	totals
ym	
1998-11-01	21174
1998-12-01	870202
1999-01-01	585569

Now that the data is in the desired format, we can plot our data on a line graph

In [31]:
```
ax = gun_totals_by_ym.plot.area(figsize=(20,10), alpha=0.5)
ax.set_title('Background Checks during Gun Purchases within 50 States Over Time
ax.set_xlabel('Month Year', fontsize=15)
ax.set_ylabel('# of Background Checks', fontsize=15)
```

Out[31]: Text(0, 0.5, '# of Background Checks')



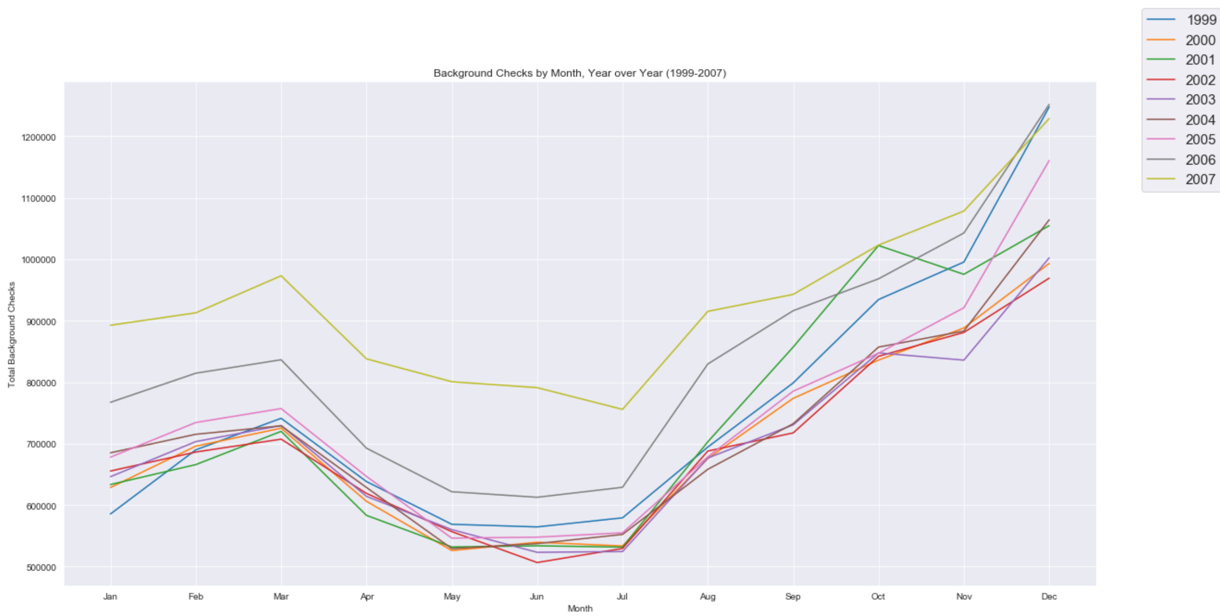
By looking at the graph above, we can see a defined "sawtooth" pattern that displays seasonal behaviors. To better understand the trends, let's take a closer by viewing the data at a monthly level and display the yearly trends separate from one another. And because there graph above has a relatively steep slope, the view will be broken up into two graphs:

1. Years ranging from 1999-2007
2. Years ranging from 2008-2016

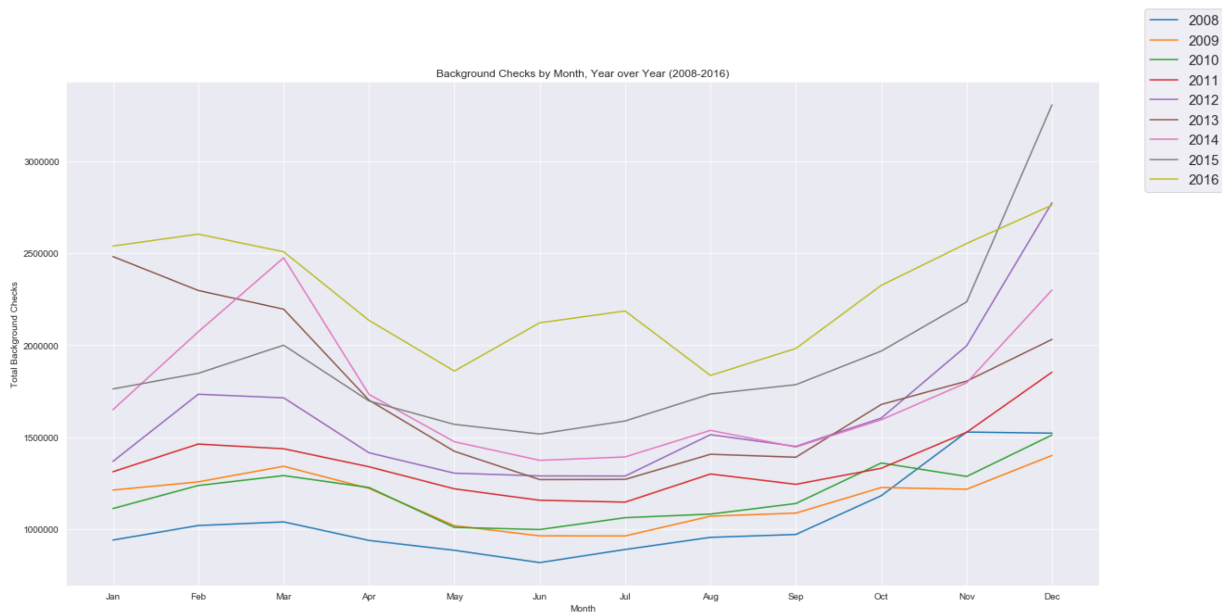
Note: Years 1998 and 2017 are excluded due to both years not having 12 months.

In [32]:
```
# The years 1998 and 2017 are excluded because they do not have 12 months worth o
years_to_drop = ['1998','2017']

df_g2 = df_g.loc[~df_g['year'].isin(years_to_drop)]
```

In [33]:
```
df_g_view = df_g2[['months','year','totals']]

labels = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec
legend_values = []
plt.figure(figsize=(20,10))

for i in range(1999,2008):
    df1 = df_g_view[(df_g_view['year']==i)].groupby(['months','year']).sum()
    legend_values.append(i)
    plt.plot(labels, df1)
    plt.legend(legend_values, bbox_to_anchor=(1, 1),bbox_transform=plt.gcf().tra
    plt.xlabel('Month')
    plt.ylabel('Total Background Checks')
    plt.title('Background Checks by Month, Year over Year (1999-2007)')
```



In [34]:
```
df_g_view = df_g2[['months','year','totals']]

labels = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec
legend_values = []
plt.figure(figsize=(20,10))

for i in range(2008,2017):
    df1 = df_g_view[(df_g_view['year']==i)].groupby(['months','year']).sum()
    legend_values.append(i)
    plt.plot(labels, df1)
    plt.legend(legend_values, bbox_to_anchor=(1, 1),bbox_transform=plt.gcf().tra
    plt.xlabel('Month')
    plt.ylabel('Total Background Checks')
    plt.title('Background Checks by Month, Year over Year (2008-2016)')
```



As we can see from both graphs above, there early and late part of the year tends to have a greater number of background checks than the middle of the year. Therefore, we can say that there is definitely a seasonal pattern for firearm shopping and background checks.

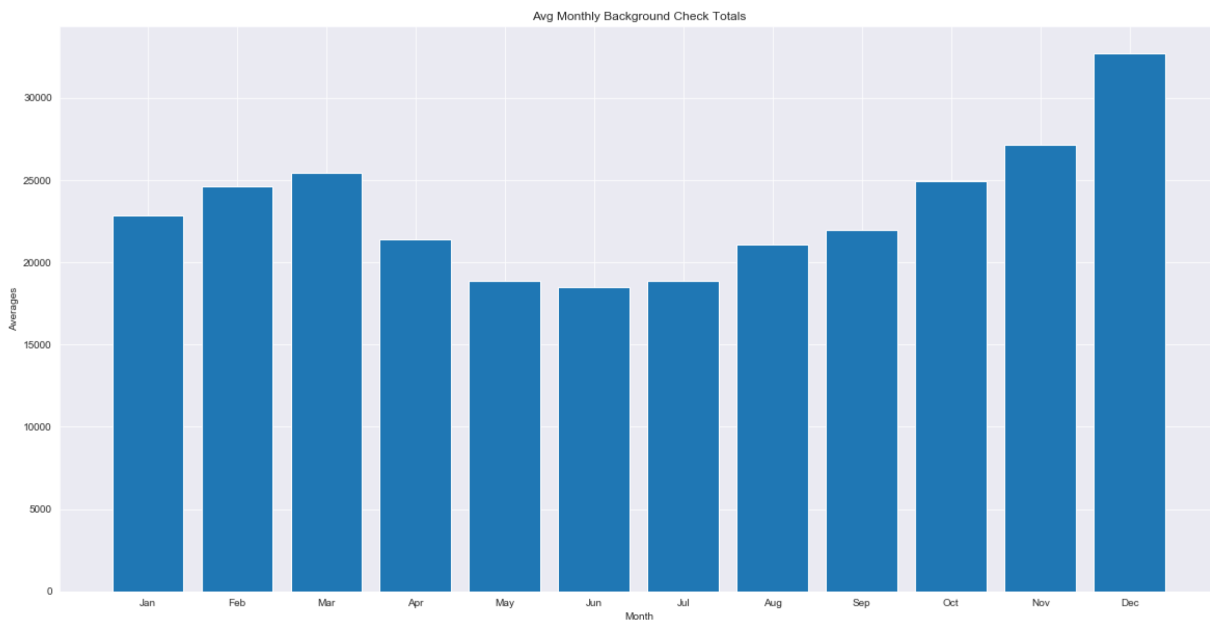
Research Question 2; What is the avg background checks completed by month from (1999- 2016)? And what is the month with the greatest avg?

In [35]:
```
df_g_view2 = df_g2[['months','totals']]
df_mean_by_month = df_g_view2.groupby(['months']).mean()
df_mean_by_month.sort_values(by=['totals'],ascending=False)
```

	totals
months	
12	32701.125556
11	27155.866667
3	25463.367778
10	24934.721111
2	24606.023333
1	22828.961111
9	21941.103333
4	21410.198889
8	21056.507778
5	18891.021111
7	18859.944444
[illegible]	[illegible]

In [36]:
```
# show the mean of each month in every year by plotting bar.

labels = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec
heights = df_mean_by_month['totals'].unique()
locations = df_mean_by_month.index.tolist()


plt.figure(figsize=(20,10))
plt.bar(locations, heights, tick_label=labels)
plt.xlabel('Month')
plt.ylabel('Averages')
plt.title('Avg Monthly Background Check Totals')
```

Out[36]:
```
Text(0.5, 1.0, 'Avg Monthly Background Check Totals')
```



By the chart above we can see that December is the month with the largest average number of background checks

Research Question 3; Which states had the highest growth in gun registrations?

In [37]:
```
# Grouping the dataframe 'df_g2' by ym and state
gun_all = df_g2.groupby(['ym', 'state'])['totals'].sum()
```

In [38]:
```
# Find out the earliest and latest registration date
latest_date = df_g2['ym'].max()
earliest_date = df_g2['ym'].min()
```

In [39]:
```
gun_all.loc[latest_date]
```

```
Idaho                17034
Illinois            137994
Indiana              95672
Iowa                 18090
Kansas               22282
Kentucky            397059
Louisiana            51493
Maine                10528
Maryland             17984
Massachusetts        19842
Michigan             47834
Minnesota            60056
Mississippi          36091
Missouri             63289
Montana              12610
Nebraska              8790
Nevada               14889
New Hampshire        14305
New Jersey           10577
New Mexico           17518
```

In [40]:
```
# The % growth of registed guns from latest year/month from the earliest year/mo
gun_growth_rate = (((gun_all.loc[latest_date] - gun_all.loc[earliest_date])/gun_

# Finding the greatest difference using idmax and pulling in the associated name
print('The state with greatest % growth in background checks is', gun_growth_rat
```

```
The state with greatest % growth in background checks is Massachusetts with a d
ifference of 2254.0 %
```

In [41]:
```
# Displaying the top 10 states
gun_growth_rate.sort_values(ascending=False).head(10)
```

Out[41]:
```
state
Massachusetts    2254.0
Kentucky         2117.0
New Hampshire     877.0
Minnesota         793.0
South Dakota      673.0
Florida           663.0
Wisconsin         632.0
Washington        581.0
Indiana           524.0
Utah              483.0
Name: totals, dtype: float64
```

By looking at the tale values for each state, and measuring the percent growth, we see that Massachusetts has the highest percent growth in background checks with Kentuckey right behind them. The interesting thing here is the difference between the top 2 vs the rest of the states. The top two are significantly different which and it is worth looking into Kentucky and Massachusetts at a yearly level to determine where the large increases occurred.

In [42]:
```
df_g_view3 = df_g2[['state','year','totals']]
df_mass = df_g_view3[df_g_view3['state']=='Massachusetts']
df_ken = df_g_view3[df_g_view3['state']=='Kentucky']
df_ken.head()
```

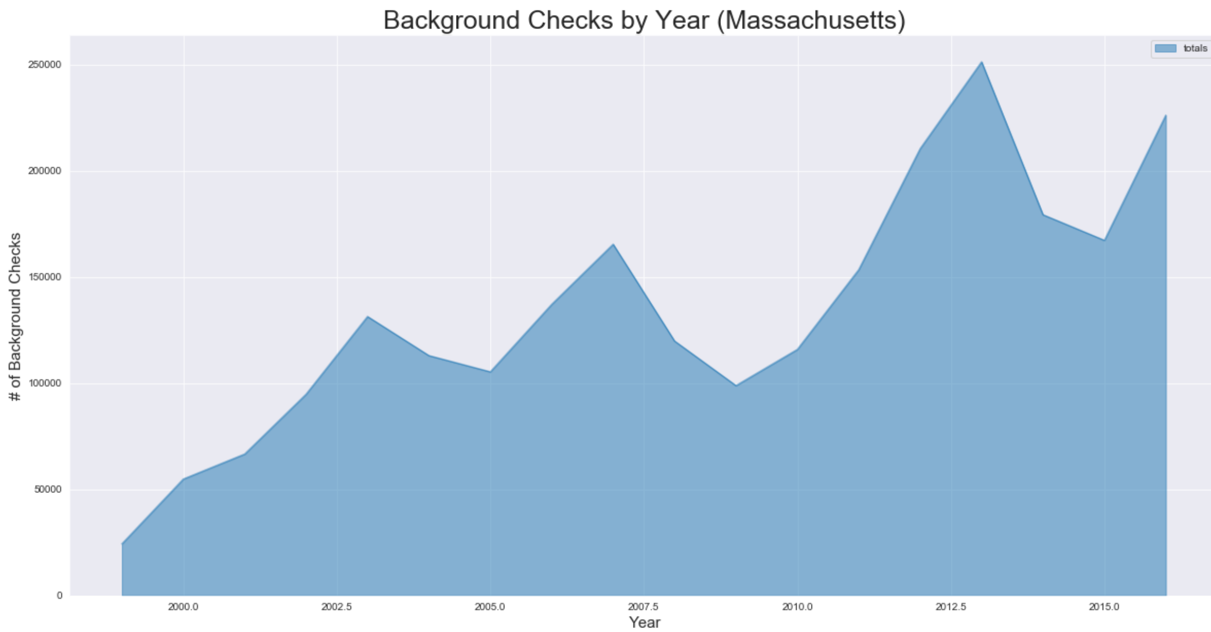
Out[42]:

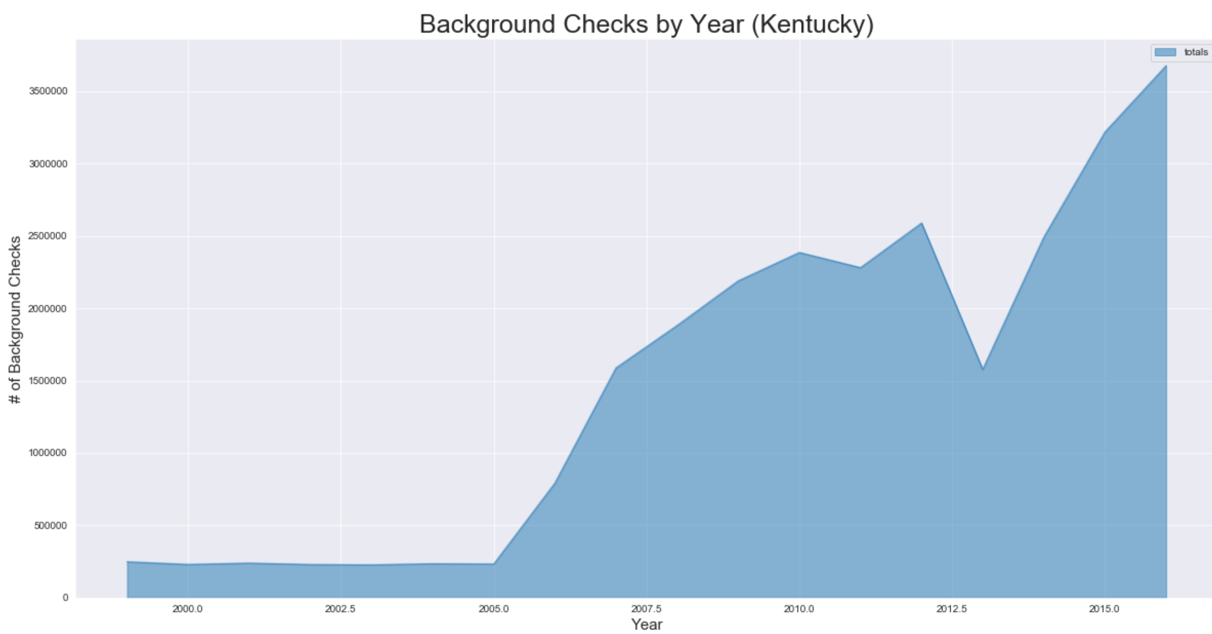
	state	year	totals
466	Kentucky	2016	397059
516	Kentucky	2016	330444
566	Kentucky	2016	378973
616	Kentucky	2016	298753
666	Kentucky	2016	29746

In [43]:
```
df_mass_sum = df_mass.groupby(['year']).sum()
df_ken_sum = df_ken.groupby(['year']).sum()
```

In [44]:
```
labels = df_mass['year'].unique().tolist()
ax2 = df_mass_sum.plot.area(figsize=(20,10), alpha=0.5)
ax2.set_title('Background Checks by Year (Massachusetts)', fontsize=25)
ax2.set_xlabel('Year', fontsize=15)
ax2.set_ylabel('# of Background Checks', fontsize=15)

labels = df_ken['year'].unique().tolist()
ax = df_ken_sum.plot.area(figsize=(20,10), alpha=0.5)
ax.set_title('Background Checks by Year (Kentucky)', fontsize=25)
ax.set_xlabel('Year', fontsize=15)
ax.set_ylabel('# of Background Checks', fontsize=15)
```

Out[44]:
```
Text(0, 0.5, '# of Background Checks')
```





Looking at the two graphs above, we can see that there was a major increase in background checks in years (2005-2012) for Kentucky and a large increase in Massachusetts from 1999 to 2003 and again from 2008 to 2014. The census data could be used here to look at relationships between housing, unemployment, persons in poverty, etc... and those increase in background checks (gun sales).

Conclusions

1. Does Firearm shopping have a seasonal pattern?

> Yes, friearm shopping and background checks have a seasonal pattern. From analyzing FBI gun data from 1999-2016 of all the states, there's a clear pattern that the background checks drop in the summer time and peaks in the winter and spring seasons.

2. What is the avg background checks completed by month? And what is the month with the greatest avg?

> It is apparent that gun sales and background checks are highest from September to March.

3. Which states had the highest growth in gun registrations?

> Massachusetts and Kentucky had the largest changes from 1999 to 2016. When plotting these results we can see certain year ranges in which the majority of background checks occurred. By identifying these time periods, we can use the census data to explore variables that are correlated to the growth. Due to time restrictions, we will not be exploring this further

In [45]:
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Investigate_FBI_Gun_Dataset_20180108.ipynb']
```

Out[45]: 4294967295