# Step 1: Business and Data Understanding

## 1. What decisions needs to be made?

Ans: Perform an analysis to recommend the city for Pawdacity's newest store, based on predicted yearly sales. But this project is just to get the data ready before the prediction.

: Awesome: Good job identifying the key decision to be made.

## 2. What data is needed to inform those decisions?

Ans: The manager gives 4 csv files with the following info:

1. monthly sales for all Pawdacity stores in 2010
2. NAICS(North American Industry Classification System) data of competitors
3. partially parsed data for population
4. demographic data in wyoming

: Awesome: These data mentioned should be sufficient to carry out the analysis.

# Step 2: Building the Training Set

| Column | Sum | Mean | median |
|---|---|---|---|
| Census Population | 213,862 | 19,442 | 12,359 |
| Total Pawdacity Sales | 3,773,304 | 290,254 | 273,024 |
| Households with Under 18 | 34,064 | 3096.73 | 2646.0 |
| Land Area | 33,071 | 3006.49 | 2748.85 |
| Population Density | 63 | 5.71 | 2.78 |
| Total Families | 62,653 | 5695.71 | 5556.49 |

: Awesome: The correct averages have been reported for each column.

The calucation details are as follows

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
% matplotlib inline
store = pd.read_csv("p2-2010-pawdacity-monthly-sales.csv")
nakes = pd.read_csv("p2-wy-453910-naics-data.csv")
popul = pd.read_csv("p2-partially-parsed-wy-web-scrape.csv")
demog = pd.read_csv("p2-wy-demographic-data.csv")
```

## calculate total Pawdacity sales and collect city names

In [2]:
```
print(store.iloc[:,5:].sum(axis=1).sum())  # 3773304
print(store.iloc[:,5:].sum(axis=1).median()) #  290254
```

```
3773304
273024.0
```

In [3]:
```
store["CITY"].value_counts()
```

Out[3]:
```
Cheyenne        2
Gillette        2
Riverton        1
Buffalo         1
Casper          1
Powell          1
Evanston        1
Sheridan        1
Douglas         1
Cody            1
Rock Springs    1
Name: CITY, dtype: int64
```

In [4]:
```
cities = store["CITY"].unique()  # nump array of shape 11
```

## calculate census population

In [5]:
```
popul.dropna(axis=0, how='any', thresh=None, subset=None, inplace=True)
```

In [6]:
```
popul['Pawdacity'] = False
```

In [7]:
```
popul.tail()
```

Out[7]:

| | City\|County | 2014 Estimate | 2010 Census | 2000 Census | Pawdacity |
|---|---|---|---|---|---|
| **94** | Wamsutter\|Sweetwater | <td>503</td> | <td>451</td> | <td>261</td> | False |
| **95** | Wheatland ?\|Platte | <td>3,659</td> | <td>3,627</td> | <td>3,548</td> | False |
| **96** | Worland ?\|Washakie | <td>5,366</td> | <td>5,487</td> | <td>5,250</td> | False |
| **97** | Wright\|Campbell | <td>1,847</td> | <td>1,807</td> | <td>1,347</td> | False |
| **98** | Yoder\|Goshen | <td>161</td> | <td>151</td> | <td>169</td> | False |

In [8]:
```
import re
s="|".join(cities)
s="^("+s+")"
pool = re.compile(s)

from bs4 import BeautifulSoup
def parsenum(text):
    if str(text).isnumeric():
        return text
    soup = BeautifulSoup(text,'html.parser')
    num =""
    for digit in soup.find("td").text:
        if digit.isnumeric():
            num += digit
        if digit =="[":
            break
    if num.isnumeric():
        return int(num)
    else:
        return 0

for i in range(popul.shape[0]):
    row = popul.iloc[i,:]
    popul.set_value(i,"2014 Estimate",parsenum(row["2014 Estimate"]))
    popul.set_value(i,"2010 Census",parsenum(row["2010 Census"]))
    popul.set_value(i,"2000 Census",parsenum(row["2000 Census"]))
    if  pool.search(row["City|County"]):
        popul.set_value(i,"Pawdacity",True)
```

In [9]:
```
census = popul.drop("City|County", axis=1).astype(int)
```

In [10]:
```
census.groupby("Pawdacity").sum()
```

Out[10]:

| | 2014 Estimate | 2010 Census | 2000 Census |
|---|---|---|---|
| **Pawdacity** | | | |
| **0** | 155443 | 151098 | 134817 |
| **1** | 227191 | 213862 | 184026 |

In [11]:
```
census.groupby("Pawdacity").median()
```

Out[11]:

| | 2014 Estimate | 2010 Census | 2000 Census |
|---|---|---|---|
| **Pawdacity** | | | |
| **0** | 529.5 | 526.5 | 446.0 |
| **1** | 12190.0 | 12359.0 | 11507.0 |

## calcualte demographic info

In [12]:
```
demog['Pawdacity']= False
for i in range(demog.shape[0]):
    if pool.search(demog.iloc[i,:]["City"]):
        demog.set_value(i,"Pawdacity",True)
```

In [13]:
```
demog.groupby("Pawdacity").sum()
```

Out[13]:

| | Land Area | Households with Under 18 | Population Density | Total Families |
|---|---|---|---|---|
| **Pawdacity** | | | | |
| **False** | 48884.539141 | 23589 | 50.64 | 51039.57 |
| **True** | 33071.380389 | 34064 | 62.80 | 62652.79 |

In [14]:
```
demog.groupby("Pawdacity").median()
```

Out[14]:

| | Land Area | Households with Under 18 | Population Density | Total Families |
|---|---|---|---|---|
| **Pawdacity** | | | | |
| **False** | 218.720727 | 124.5 | 0.285 | 274.735 |
| **True** | 2748.852900 | 2646.0 | 2.780 | 5556.490 |

# Step 3: Dealing with Outliers

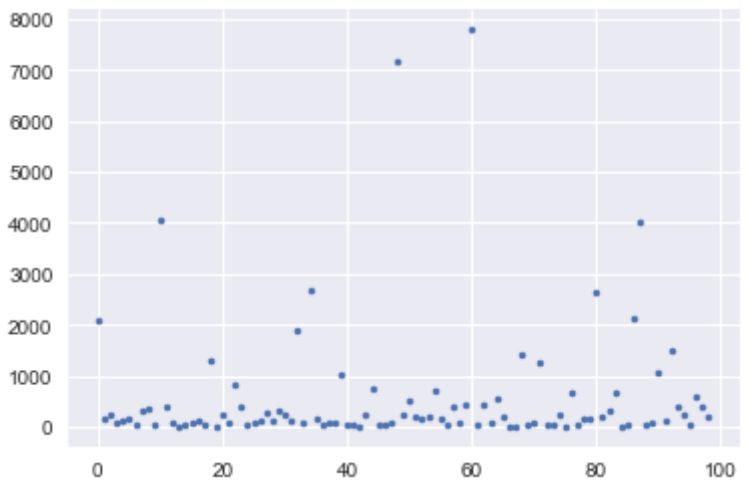
### Are there any cities that are outliers in the training set? Which outlier have you chosen to remove or impute?

Ans: The outlier is the City "Cheyenne", whose "Population Density" and "Total Families" much larger than average.

: Required: Please note that the rubric requires you to mention all the outliers in the dataset. There is at least one more outlier in the dataset. Also, note that you need to discuss which outlier you choose to remove or impute. Remember that you are allowed to remove only one outlier, so be sure to appropriately justify your final decision.

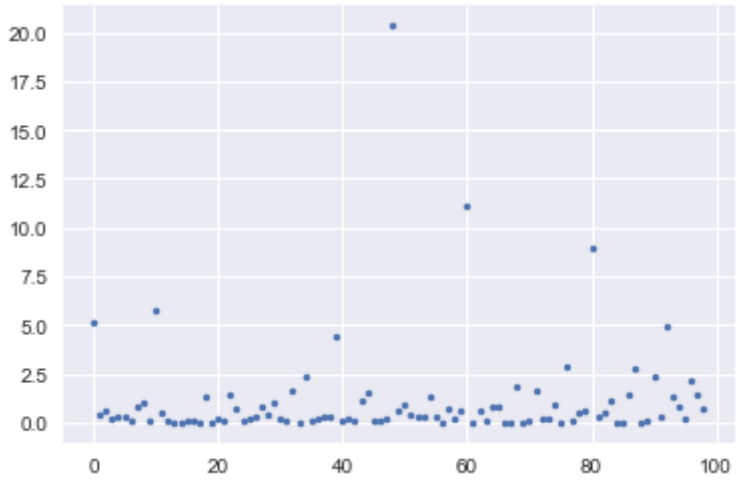
In [16]:
```
plt.plot(demog["Households with Under 18"],'.')
```

Out[16]: `[<matplotlib.lines.Line2D at 0x1161998d0>]`



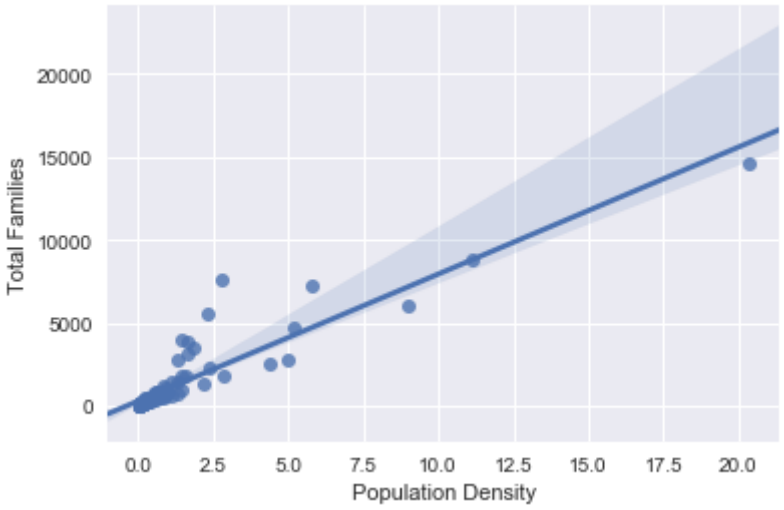
In [19]:
```
plt.plot(demog["Population Density"],'.')
```

Out[19]:
```
[<matplotlib.lines.Line2D at 0x1175f3b38>]
```




In [25]:
```
sns.regplot(demog["Population Density"],demog["Total Families"])
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x116c274e0>
```




In [30]:
```
demog[demog["Households with Under 18"]>7000]
```

Out[30]:

| | City | County | Land Area | Households with Under 18 | Population Density | Total Families | Pawdacity |
|---|---|---|---|---|---|---|---|
| **48** | Cheyenne | Laramie | 1500.1784 | 7158 | 20.34 | 14612.64 | True |
| **60** | Casper | Natrona | 3894.3091 | 7788 | 11.16 | 8756.32 | True |

## Next up

: Comment: You'll be doing this in the next project - "Recommend a City"

I expect there is a follow-up project to do the actual prediction. But it seems to be removed. So my steps for such prediciton is:

1. With the demographic statistics, figure out which city is above the average but without a Pawdacity store
2. Dig into NAICS to see whether competitors are already in the market of target cities.
3. Use existing data to predict the sales in the target cities, figure out which one will have the best sales.

In [38]:
```
nakes["SALES VOLUME"].describe()
```

Out[38]:
```
count        32.000000
mean     173630.875000
std      222548.908781
min           0.000000
25%       65500.000000
50%       86000.000000
75%      147743.250000
max      890000.000000
Name: SALES VOLUME, dtype: float64
```

In [39]:
```
nakes[nakes["SALES VOLUME"]>150e3]
```

Out[39]:

| | BUSINESS NAME | PHYSICAL CITY NAME | SALES VOLUME | CASS_LastLine |
|---|---|---|---|---|
| **0** | Mile High Mobile Pet LLC | Cheyenne | 300000 | Cheyenne, WY 82007-3528 |
| **1** | Pets City Inc | Cheyenne | 640000 | Cheyenne, WY 82009-4851 |
| **7** | Don Bruner Sales LLC | Torrington | 750000 | Torrington, WY 82240-3516 |
| **19** | L and C Pets and Gifts LLC | Evansville | 210000 | Evansville, WY 82636 |
| **20** | All Gods Creatures | Gillette | 450000 | Gillette, WY 82716-2919 |
| **21** | Camelot Pet Castle | Gillette | 230000 | Gillette, WY 82716-1704 |
| **23** | Pet Food Outlet | Gillette | 450000 | Gillette, WY 82718-6330 |
| **26** | Zoobecks Inc | Rock Springs | 890000 | Rock Springs, WY 82901-5105 |

In [ ]:
```

```