Fueldup Consumer Behavior

MSBA 5303 - Programming for Analytics

University of Central Oklahoma

Project Report

Team Dynamic TRiO:

Sahil Agarwal

Tanay Patel

Aakash Patel

Due Date: 04 December 2019

Table of Contents

Executive Summary

Technology has changed the way we do business, how we communicate, and has had a transformational impact on the concept of the modern office. Using technological innovations can have a beneficial effect on how a business functions and how successful an organization is. While the future of work is remote, This report sheds light on assisting one such tech company with their business problems, that is the Fueldup company whose motto is to save their customer's time and make them hassle free by delivering them gas at their convenience. By providing rigorous and statistical analysis, key findings are highlighted to provide the Fueldup company with beneficial evidence for their target customers and areas. The graphs and tables that are attained through numerous analysis are provided throughout the report to support the claims. Overall, this report provides detailed results gained by performing analysis on numerous Oklahoma datasets which could turn out to be beneficial for the Fueldup company in determining their target customers and areas in Oklahoma.

Business Problem

Fueldup, a startup company based in Oklahoma was looking for volunteers to support them in analyzing their target customers and areas in Oklahoma which could help them generate most revenue. A goal of Fueldup company is to deliver enough amount of gas with the equal or less price to people's homes or inside the community. The company aims to target corporate, where the company will pay the delivery fees to Fueldup in exchange for their service and save their employee's time. Moreover, the company also aims to target families who do not have time to go to the gas station and refill their gas because of their hectic schedule. Apart from this, through our research, our aim was to find groups of people who would be more willing to utilize this fuel

delivery service, such as, people who utilizes delivery services like GrubHub, Doordash and other convenience services. The company was already delivering its service to the corporates but later on, they wanted to expand their service. The reason for expanding their network was to generate more revenue and help people with their daily life tasks. This kind of service is already present in different states of the United States i.e. California, Florida, etc. This kind of service got a great response in different states and making a great amount of money at fair prices.

Company Description

In today's fast-paced world, with very little spare time in hand, where everyone is busy in their life whether that is going to work, attending meetings, taking kids to their activities or family outings, things like Fueldup is what everyone needs to take some load off your back. Fueldup is a fuel delivery service that brings fuel directly to you whether that is at your workplace, home or anywhere else. They serve an area on a weekly or bi-weekly basis. They currently have three gas trucks that fill up gas from their storage facility (which can hold up to 10,000 gallons of gas) and deliver to their customers ("We get you fueldup so you can be freedup", 2019).

With the advancement in technology, delivering goods to customers' home started a while ago. The trend then moved to the food industry, and apps like Grubhub, Doordash and Uber Eats came into existence. These apps deliver food from restaurants right at the customer's doorstep for a little convenience fee ("Cheapest Food Delivery Service?", 2019). The success of these apps influenced the idea to deliver fuel to customers' cars, so customers don't have to go to gas stations to fill up gas. Fueldup which is a startup company has an innovative idea of delivering gas to the people rather than having to wait in a long line just to fill up your car. They provide gas at the same or even lower rate in comparison to gas stations in your neighborhood ("We get you fueldup so you can be freedup", 2019). Fueldup will provide such service in the Oklahoma area, but the problem is that the idea is very new in this market and they currently don't know who their target customers are? Since the food delivery industry is booming and the idea of delivering gas is very new, it was very interesting for our team to see how well does Fueldup performs in the Oklahoma market.

Fueldup is first of its kind in Oklahoma, so finding their potential customers can be both very challenging and exciting. Our plan was to gather data related to taxes paid by gas stations and combine it with demographics data to provide a solution to this problem. The company needed a

hand to analyze their target customers, so, by analyzing the customer demographics information like income, age, marital status and more, this report provides the company with some valuable customer information which will aid them in targeting their customers. We are not only able to provide customer statistics but also the populated areas that they should target where people usually tend to fill up their cars.

We broke down our analysis on several basis and assembled questions through which we can base our analysis on, questions such as:

1. Which county has the largest number of registered vehicles and the amount of gas taxes collected?
2. Which county according to the gender has the driving percentage?
3. Which county has the maximum percentage of FamilyWork and their income?
4. Which county has individuals whose income is more than the average income of an individual in Oklahoma?
5. Which county has the highest number of people preferring to drive their own car for daily commutes over other transportation services?
6. Which county has the highest mean commute time of people commuting for work?
7. Which county has the highest number of employed individuals?

Data understanding

Initial data was collected from several different sources. Through our research, our goal was to give recommendations to the company to target customers in Oklahoma. Data was collected in different forms:

- Demographic information.
- Vehicle registered dataset.
- Gas Excise tax information.
- Zip Code information of each city in Oklahoma.
- FIPS (Federal Information Processing Standard) Zip Code information.

Data description & sources

I. Demographic Information

We have collected two datasets of 2015 and 2017 year. Both of the datasets have the necessary information about the demographics of Oklahoma's residents.

Source: Kaggle

Link : https://www.kaggle.com/muonneutrino/us-census-demographic-data

File_2015 : Rows - 1046, Columns - 37

	TractId	State	County	TotalPop	Men	Women	Hispanic	White	Black	Native	Asian	Pacific	VotingAgeCitizen	Income	IncomeErr	Incomel
0	40001376600	Oklahoma	Adair	2575	1343	1232	5.3	60.9	0	23.1	0.7	0	1915	34856	4219	
1	40001376700	Oklahoma	Adair	5420	2526	2894	4.6	49.9	0.6	24.8	0.4	0.1	3816	35625	3461	
2	40001376800	Oklahoma	Adair	4584	2380	2204	3.9	40.2	0.4	46.2	0.4	0	3422	35159	3612	

File_2017 : Rows - 1046, Columns - 37

	TractId	State	County	TotalPop	Men	Women	Hispanic	White	Black	Native	Asian	Pacific	VotingAgeCitizen	Income	IncomeErr	Incomel
0	40001376600	Oklahoma	Adair	2512	1248	1264	5.1	58.9	0	26.9	1.3	0	1945	37195	4471	
1	40001376700	Oklahoma	Adair	5053	2381	2672	5.1	48.6	0.1	33.5	0.1	0.9	3640	34633	3043	
2	40001376800	Oklahoma	Adair	4712	2430	2282	4.1	39	0.4	48.2	0.6	0	3532	36217	4042	

Figure 1. Demographics dataset of 2015 & 2017

Column Definitions:

- ➔ TractId: Census tract ID
- ➔ State: State, DC, or Puerto Rico
- ➔ County: County of United States
- ➔ TotalPop: Total population
- ➔ Men: Number of men
- ➔ Women: Number of women
- ➔ Hispanic: % of population that is Hispanic/Latino
- ➔ White: % of population that is white
- ➔ Black % of population that is black
- ➔ Native: % of population that is Native American or Native Alaskan
- ➔ Asian: % of population that is Asian
- ➔ Pacific: % of population that is Native Hawaiian or Pacific Islander
- ➔ Citizen: Number of citizens
- ➔ Income: Median household income ($)
- ➔ Poverty: % under poverty level
- ➔ Drive: % commuting alone in a car, van, or truck
- ➔ Carpool: % carpooling in a car, van, or truck
- ➔ Transit: % commuting on public transportation
- ➔ Walk: % walking to work
- ➔ OtherTransp: % commuting via other means
- ➔ MeanCommute: Mean commute time (minutes)
- ➔ Employed: Number of employed (16+)
- ➔ FamilyWork: % in unpaid family work
- ➔ Unemployment: Unemployment rate (%)

II. Vehicle Registration

The goal was to retrieve information about the vehicles in the state of Oklahoma. This dataset provides the number of vehicles registered in each county. The dataset was crucial in finding information about the target customer.

Source: Government website.

Link: https://www.ok.gov/tax/Individuals/Motor_Vehicle/

	county_name	number_registered
0	Adair	11291
1	Alfalfa	6790
2	Atoka	13575

Figure 2. Vehicle registration dataset

Column Definitions:

- ➔ County_Name: Name of Oklahoma county
- ➔ Number_registered: Number of registered vehicles for a county

III. Gas Excise Tax

The gas excise tax dataset provides information about the cities that are paying taxes on gas in the Oklahoma state. This dataset helped in achieving the region, who are paying more taxes on the gas. Information can be used to make the initial move in targeting customers in those cities.

Source: Government website.

Link: https://www.ok.gov/tax/Forms_&_Publications/Forms/Motor_Fuel/#

	Copo	Name	Distribution Date	Amount
0	105.0	Town of Stilwell	10-Jul-18	652.71
1	105.0	Town of Stilwell	10-Aug-18	643.07
2	105.0	Town of Stilwell	11-Sep-18	631.59

Figure 3. Gas excise tax information

Column Definitions:

- ➔ Name: Name of the city in Oklahoma
- ➔ Distribution Date: Date on which the gas tax amount was paid
- ➔ Amount: Gas tax amount paid ($)

IV. Oklahoma Zip Code

This dataset provides information about the zip codes based on cities and counties. We found this dataset on google. This dataset helped link each city zip code based on county name and city column.

Source: Google.

Link: https://www.zipcodestogo.com/Oklahoma/

	zip	city	state_id	state_name	county_name
0	73002	Alex	OK	Oklahoma	Grady
1	73003	Edmond	OK	Oklahoma	Oklahoma
2	73004	Amber	OK	Oklahoma	Grady

Figure 4. Oklahoma zip code

Column Definitions:

- ➔ Zip: Zip code
- ➔ City: City name
- ➔ State_Id: State abbreviation
- ➔ State_Name: State name
- ➔ County_Name: County name

V. FIPS Zip Code

The purpose of using the Fips Zip Code dataset was to get the information about the uniquely identified counties and county equivalents in the United States. Moreover, the dataset has a unique code which individually identifies counties and their assigned zip codes.

Source: Kaggle.

Link: https://www.kaggle.com/danofer/zipcodes-county-fips-crosswalk

	ZIP	COUNTYNAME	STATE	STCOUNTYFP	CLASSFP
0	36003	Autauga County	AL	1001	H1
1	36006	Autauga County	AL	1001	H1

Figure 5. Fips Zip Code

Column Definitions:

- ➔ ZIP: Zip code
- ➔ County name: US County Name
- ➔ State: US State
- ➔ St County FP: US State & County FIPS ID
- ➔ Class FP: FIPS Class Code, as defined by the Census

Describe & Explore Data

I. Demographic Information

The dataset has three data types and their respective summary statistics have been shown below. From the statistics, County has 79 unique counties available in the dataset. Additionally, the data types i.e. float & integer showing the mean and standard deviation of the variables. We would like to mention some finding from the data types such as Average Income in the Oklahoma state is $48,790.

Object DataTypes

	count	unique	top	freq
State	2090	1	Oklahoma	2090
County	2090	79	Oklahoma	480

Float DataTypes

	mean	std
Hispanic	277.522	732.89
White	309.478	720.107
Black	276.939	733.176
Native	271.188	735.085
Asian	265.869	737.125
Pacific	263.453	738.06
Poverty	281.858	731.08
ChildPoverty	286.979	729.163
Professional	290.948	727.527
Service	277.546	732.506
Office	280.951	731.169

Integer DataTypes

	mean	std	min	max
TractId	3.0073e+10	1.85478e+10	2090	4.0154e+10
TotalPop	4146.34	4575.28	99	14928
Men	2219.65	2313.94	44	7647
Women	2186.99	2190.71	19	7281
VotingAgeCitizen	3024.88	3165.52	66	10453
Income	48790.6	52859.3	2090	170278
IncomeErr	11349.2	18857.2	841	57603
IncomePerCap	29187.5	39217.6	1274	122826
IncomePerCapErr	8219.2	16252.8	638	48371
Employed	2106.09	2303	13	7540

Figure 6. Summary Statistics of different data types

Our team sorted 5 counties from 79 unique counties based on the employed column. Our final analysis contained only these counties and their respective cities to get precise output. Employed column defined the number of employed people in each county. Furthermore, we noticed that the higher employed showed a great number of people who drive too.

County	Employed	TotalPop	Mean_Income	Professional	FamilyWork	Drive
Oklahoma	717041	1528683	50233.0	15853.8	72.7	38542.9
Tulsa	612206	1260458	53877.2	12255.4	57.6	28612.2
Cleveland	271664	542638	58547.2	4939.8	16.1	10191.4
Canadian	130217	259115	65715.3	1985.7	11.2	4929.1
Comanche	99056	248597	45266.2	2016.1	7.2	4650.1

Figure 7. Top 5 county evaluated based on sum of employed population

II. Vehicle Registration

Vehicle registration dataset has only two columns through which we were able to get the number of vehicles registered in each county. Figure 8 below shows the top 4 counties that have the largest number of vehicles registered. Additionally, if we compare Figure 7 and Figure 8, it shows the same county names. We were more confident in using these counties for our further analysis.

county_name	number_registered
Oklahoma	840960
Tulsa	554962
Cleveland	238680
Canadian	131894

Figure 8. Top 4 vehicles registered in the county

III. Gas Excise Tax

Gas Excise Tax dataset has two columns through which we were able to get the total amount of gas excise tax paid by each county. The largest amount of taxes paid by the top three

cities are Oklahoma City, Tulsa, and Norman. The amount is more than or equal to $200,000 and the rest of the cities have less than $200,000.

City_Name	Amount
Oklahoma City	1182163.0
Tulsa	737971.0
Norman	225621.0

Figure 9. Top 3 cities paying highest Gas taxes

Figure 10 below shows the monthly trend of one-year period in paying the taxes. (Left image) shows the amount paid each month and (Right image) shows the visualization of the dataset. We have noticed that the amount paid by the cities is lower in October, March, and May. One of the reasons for having low taxes on gas is the price inflation in these three months.

Distribution Date	Amount
2018-07-31	492564.91
2018-08-31	485289.01
2018-09-30	476621.57
2018-10-31	414687.69
2018-11-30	512198.78
2018-12-31	468349.77
2019-01-31	454482.85
2019-02-28	461568.05
2019-03-31	377534.12
2019-04-30	455299.27
2019-05-31	382649.26
2019-06-30	565601.46

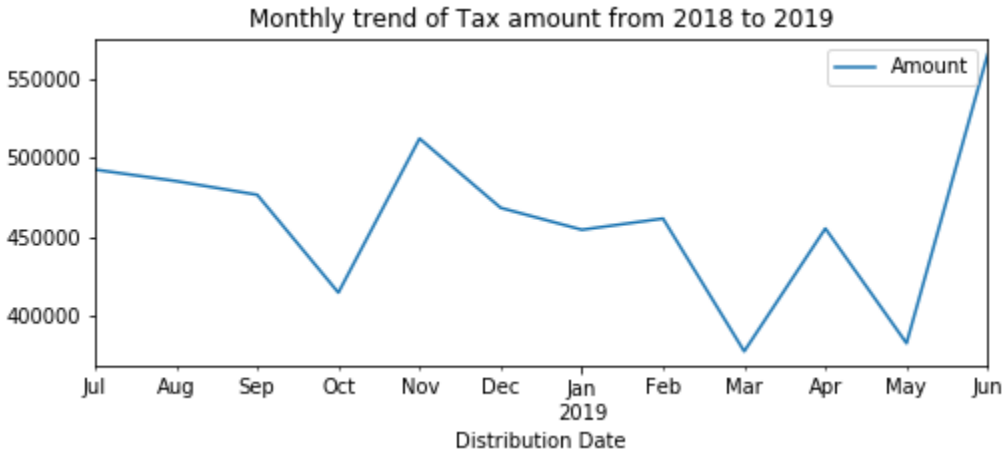

Figure 10. Left - Monthly Amount of taxes paid by the cities, Right - Visualized the dataset

IV. Oklahoma Zip Code

This dataset has three object data types and one integer column. Below we have shown the count of unique zip codes, cities, state and county name. This dataset was helpful in getting zip codes of each city and county.

	Unique Values
zip	648
city	557
state_id	1
county_name	77

Figure 11. Unique values in each column

V. FIPS Zip Code

This dataset has three object data type and two integer columns. Zip code dataset has information about all the available counties and cities in the United States based on Federal Information Processing Standard (FIPS). Overall, there were 1958 counties and 39456 zip codes.

	Unique Values
ZIP	39456
COUNTYNAME	1958
STATE	54
STCOUNTYFP	3223
CLASSFP	5

Figure 12. Unique values in each column

Verify Data Quality

To maintain the data quality, our team decided to reduce and change the data types of each column in the datasets. In our analysis, less memory usage helped in merging each dataset much faster as compared to default data type assigned by Python.

I. Demographic Information

After reducing bytes of 10 integer data types from ‘ínt64’ to ‘int32’. Demographic dataset reduced the memory usage from 620.5+ KB to 538.8+ KB.

	Count of Data Types
float64	25
int32	10
object	2

Figure 13. Data type count

II. Vehicle Registration

Below, left table shows number registered with ‘int64’ data type. Right table shows number registered with ‘int32’ data type. The memory was reduced from 1.3+ KB to 1.0+ KB.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 77 entries, 0 to 76
Data columns (total 2 columns):
county_name          77 non-null object
number_registered    77 non-null int64
dtypes: int64(1), object(1)
memory usage: 1.3+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 77 entries, 0 to 76
Data columns (total 2 columns):
county_name          77 non-null object
number_registered    77 non-null int32
dtypes: int32(1), object(1)
memory usage: 1.0+ KB
```

Figure 14. (Left) Original data types and (Right) Assigned data type to reduce the memory usage.

III. Gas Excise Tax

In this dataset, there was no column which was needed to be changed to minimize the memory usage. Below is a screenshot of the original dataset which contains count of values, data types and memory usage.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7008 entries, 0 to 7007
Data columns (total 4 columns):
Copo                7008 non-null float64
CityName            6996 non-null object
Distribution Date   7008 non-null datetime64[ns]
Amount              7008 non-null object
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 273.8+ KB
```

Figure 15. Information of Original dataset

IV. Oklahoma Zip Code

Zip column has been changed from ‘int64’ to ‘int32’ which reduced the memory usage from 20.4+ KB to 17.8+ KB. Below tables is the snapshot of the count of values in each column, data types and memory usage.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 648 entries, 0 to 647
Data columns (total 4 columns):
zip            648 non-null int64
city           648 non-null object
state_id       648 non-null object
county_name    648 non-null object
dtypes: int64(1), object(3)
memory usage: 20.4+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 648 entries, 0 to 647
Data columns (total 4 columns):
zip            648 non-null int32
city           648 non-null object
state_id       648 non-null object
county_name    648 non-null object
dtypes: int32(1), object(3)
memory usage: 17.8+ KB
```

Figure 16. (Left) is a snapshot of the original dataset and (Right) is a snapshot of the modified dataset

V. FIPS Zip Code

Dataset has been changed to reduce the memory usage. As there are two integer columns which has been changed to ‘int32’ to reduce the memory usage to 0.4 MB.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52889 entries, 0 to 52888
Data columns (total 5 columns):
ZIP           52889 non-null int64
COUNTYNAME    52889 non-null object
STATE         52889 non-null object
STCOUNTYFP    52889 non-null int64
CLASSFP       52889 non-null object
dtypes: int64(2), object(3)
memory usage: 2.0+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52889 entries, 0 to 52888
Data columns (total 5 columns):
ZIP           52889 non-null int32
COUNTYNAME    52889 non-null object
STATE         52889 non-null object
STCOUNTYFP    52889 non-null int32
CLASSFP       52889 non-null object
dtypes: int32(2), object(3)
memory usage: 1.6+ MB
```

Figure 17. (Left) is a snapshot of the original dataset and (Right) is a snapshot of the modified dataset.

Data Preparation

All of the data files our team used for this project, needed to be cleaned before merging it into the main dataset. All of the cleaning, merging and analyzing for this project is done using python. Our data preparation process include: importing different data files, renaming column names, removing columns and rows whose 60% or more values include null, N/A, or NAN, replacing some null values with 0.0 to assist in the analysis process, and merging the data.

Importing data files:

We used four different datasets to analyze the Oklahoma region for evaluating target customers for FueldUp.

file_2015.csv & file_2017.csv - Contains demographic information about Oklahoma state residents.

Vehicle_Registrations.csv - Contains information about the number of vehicles registered in each county.

ZIP-COUNTY-FIPS_2017-06.csv - Contains zip codes and its corresponding Fips code to assist with visualizations containing maps.

YTD_Gas_Excise.csv - Contains Gas Tax excise information based on City and Distribution date

Reading the files:

```
    # Demographic(file_2015.csv & file_2017.csv),
    # Vehicle Registrations(Vehicle_Registrations.csv)
    # Gas Excise(YTD_Gas_Excise.csv)
    # Zip Code(ZIP-COUNTY-FIPS_2017-06.csv)
file_2015 = pd.read_csv('acs2015.csv')
file_2017 = pd.read_csv('acs2017.csv')
vehicle_reg = pd.read_csv('Vehicle_Registrations.csv')
gas_excise = pd.read_csv('YTD_Gas_Excise.csv')
```

```
fips_zip_code = pd.read_csv('ZIP-COUNTY-FIPS_2017-06.csv')
ok_zip = pd.read_csv('OK_zipcode.csv')
```

Retrieving only 'Oklahoma' state data from the dataset:

```
# data_2015
filt1 = file_2015["State"] == "Oklahoma"
file_2015 = file_2015.loc[filt1].reset_index()
file_2015.drop(columns='index',inplace=True)
# data_2017
filt3 = file_2017["State"] == "Oklahoma"
file_2017 = file_2017.loc[filt3].reset_index()
file_2017.drop(columns='index',inplace=True)
```

Check to see if data types match in both of the tables:

```
a = file_2015.dtypes.value_counts().to_frame("Count")
b = file_2017.dtypes.value_counts().to_frame("Count")
a1 = a.style.set_table_attributes("style='display:inline'").set_caption("File_2015")
b1 = b.style.set_table_attributes("style='display:inline'").set_caption("File_2017")
display_html(a1._repr_html_()+" "+b1._repr_html_(), raw=True)
```

Removing columns that contains all null values or columns that are not useful in analysis:

```
# Checking if whole row or column is missing or not.
file_2015.dropna(how='all',inplace=True)  # Checking whole row.
file_2015.dropna(how='all',axis='columns',inplace=True)   # Checking whole column.
file_2017.dropna(how='all',inplace=True)
file_2017.dropna(how='all',axis='columns',inplace=True)
```

Removing some rows for better analysis results:

Rows which has more than 60% of empty or NULL values, were removed to provide better results while analyzing this dataset.

```
dem_data.dropna(thresh = dem_data.shape[1]*0.6,how='all',inplace=True)
```

Dropping the last row in YTD_Gas_Excise.csv because that row contains the total sum of the amount column.

```
gas_excise.dropna(inplace=True)
```

Formatting Data:

Some of the columns in the dataset were renamed to for better data understanding purposes. Datatypes of some columns were also changed to assist in analysis of those columns. Values in some columns had to be trimmed to create meaningful results.

```
# Removing the "County" in each value which was attached after each county names:
file_2017['County'] = file_2017['County'].str.split(expand=True)[0]

# Changing two column names which are similar but column names are different:
file_2015.rename(columns={'CensusTract':"TractId","Citizen":"VotingAgeCitizen"},inplace=Tr
ue)

# Changing the data type of Income columns into 'Integer' datatype:
int_list = ['Income','IncomeErr','IncomePerCap','IncomePerCapErr']
dem_data[int_list] = dem_data[int_list].astype('int64')


# In the 'Name' column, we are separating cities name and inserting into new column:

a = gas_excise['Name'].str.split(expand=True)

b = a.loc[:,[2,3]]

b[3] = b[3].fillna("")

city_name = b[2]+" "+b[3]

# Concatenate the new column with the existing dataframe and replacing the 'Name' column with
'City_Name' column.

concat_city_name = pd.concat([gas_excise,city_name],axis=1)

concat_city_name.rename(columns={0:'City_Name'},inplace=True)

concat_city_name.drop(labels='Name',axis=1,inplace=True)

concat_city_name['City_Name'] = concat_city_name['City_Name'].str.strip().str.lower().str.title()

concat_city_name['City_Name'].replace({'Ft Cobb':"Fort Cobb",'Ft Gibson':'Fort
Gibson'},inplace=True)

new_col_order = ['Copo','City_Name','Distribution Date','Amount']

new_gas_excise = concat_city_name[new_col_order]
```

```
# Changing datatype of ["Amount", "Copo"] column from [object, float] to [float, int]:
new_gas_excise['Amount'] = new_gas_excise['Amount'].str.replace(",","")
new_gas_excise['Copo'] = new_gas_excise['Copo'].astype('int64')
# Forcefully changing the datatype of the 'Amount' column.
new_gas_excise['Amount'] = pd.to_numeric(new_gas_excise['Amount']).astype(int)
```

Derived Attributes

Created new column named DrivingAgeCitizen to get only citizens who are able to drive. It is achieved by subtracting VotingAgeCitizen from TotalPop:

```
dem_data['DrivingAgeCitizen'] = dem_data['TotalPop'] - dem_data['VotingAgeCitizen']
```

Methodology, Analysis and Findings

Descriptive Analysis

The basis on which our whole research was conducted on was to provide numerous insights to the Fueldup company, such as who and what type of group would be willing to use their gas delivery service and which area they should target. Therefore, the base of our analysis was exploratory data analysis, to summarize main characteristic of our dataset. We used this methodology after the data preparation phase to find the variables which best predict our research questions. We created visualizations to help better understand the findings:

1. Using demographics data, firstly we decided to find the top five counties based on driving age population. It will help determine what areas contain most population that drives a vehicle, so Fueldup can consider targeting customers there. Derived attribute DrivingAgeCitizen is calculated by subtracting VotingAgeCitizen from TotalPop. Based on the analysis and graph in Figure 18, we came up with the following five top counties for most driving age population: Oklahoma, Tulsa, Cleveland, Canadian, Comanche.

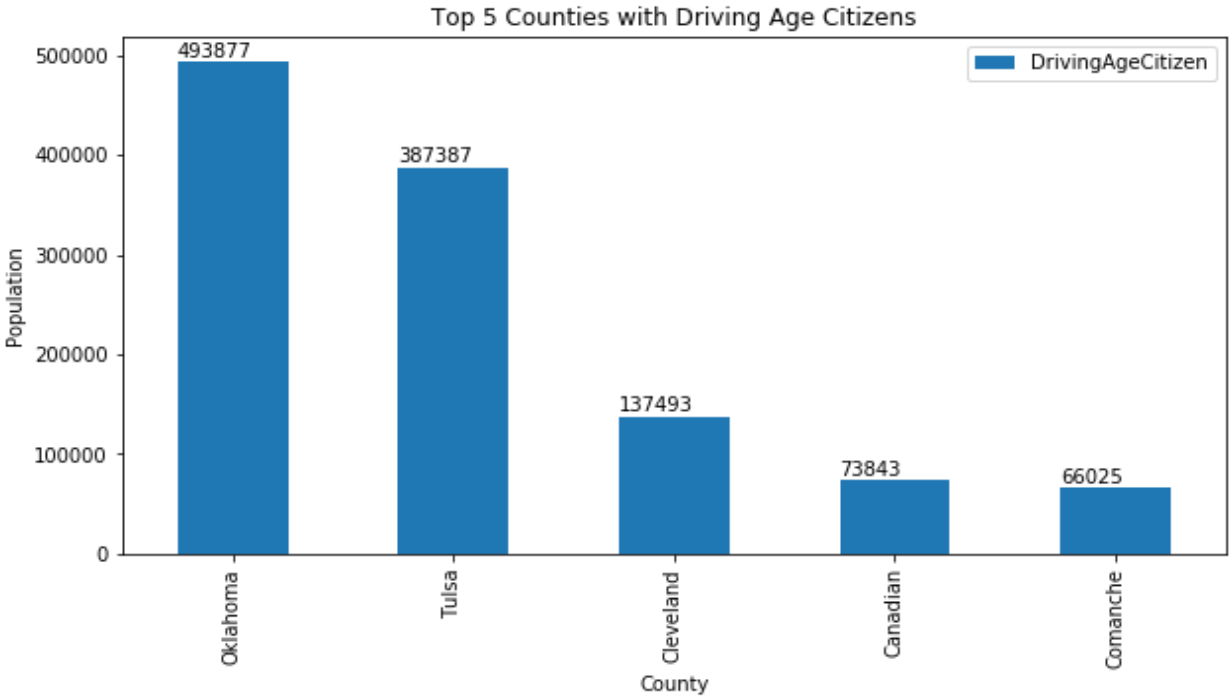

Figure 18. Top 5 Counties by Driving Age Citizens

2. Using demographics data, we then found the top five counties based on average commute time of people commuting for work. This information helped determine people from which county are commuting most for their daily work and therefore, are more likely to consume more gas on a daily basis and would most likely prefer gas delivery service over waiting in long lines at gas stations because of their hectic commute each day. Figure 19 below show the top five counties with the highest average commute time.

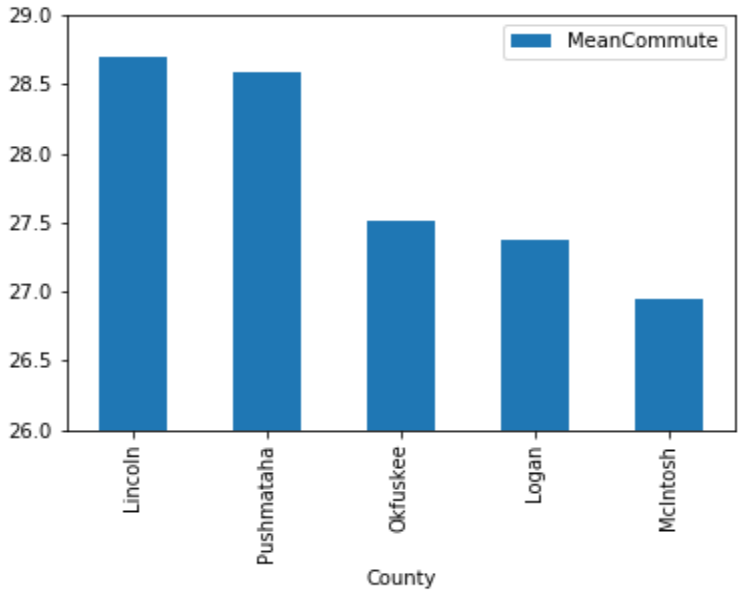

Figure 19. Top 5 Counties by Average Commute Time

3. Next to check what county has individuals who can afford and would be willing to use the gas delivery service, we found the number of employed individuals in each county who has income more than the average income of an individual in Oklahoma with the help of demographics data. Based on our analysis, the top five counties that meets this criteria are shown in Figure 20 below. To our surprise Oklahoma county didn't make it in the top five.

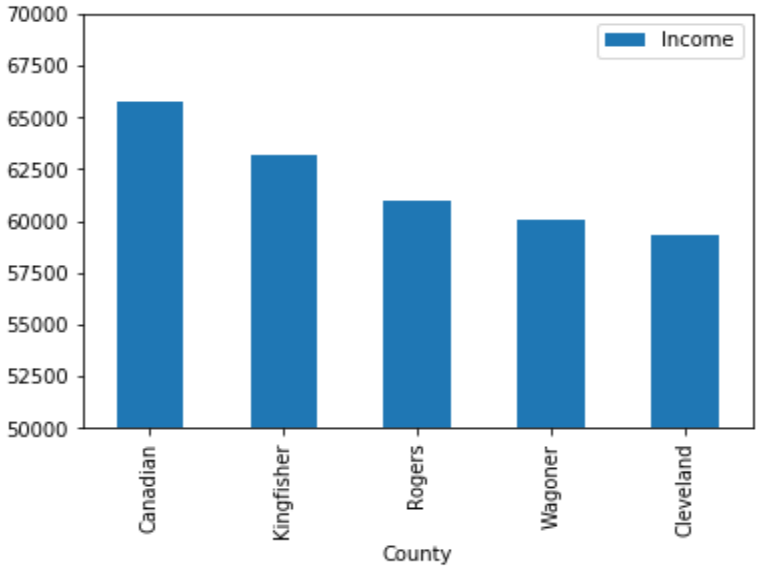

Figure 20. Top 5 Counties by Average Income

4. To better assist with targeting specific gender, we further analyzed the demographics data by finding gender disparity in the top five most populated counties of Oklahoma: Canadian, Oklahoma, Cleveland, Tulsa and Comanche. It was interesting to see that the population in all of these counties were almost evenly distributed between male and female gender. The result is shown below in Figure 21.

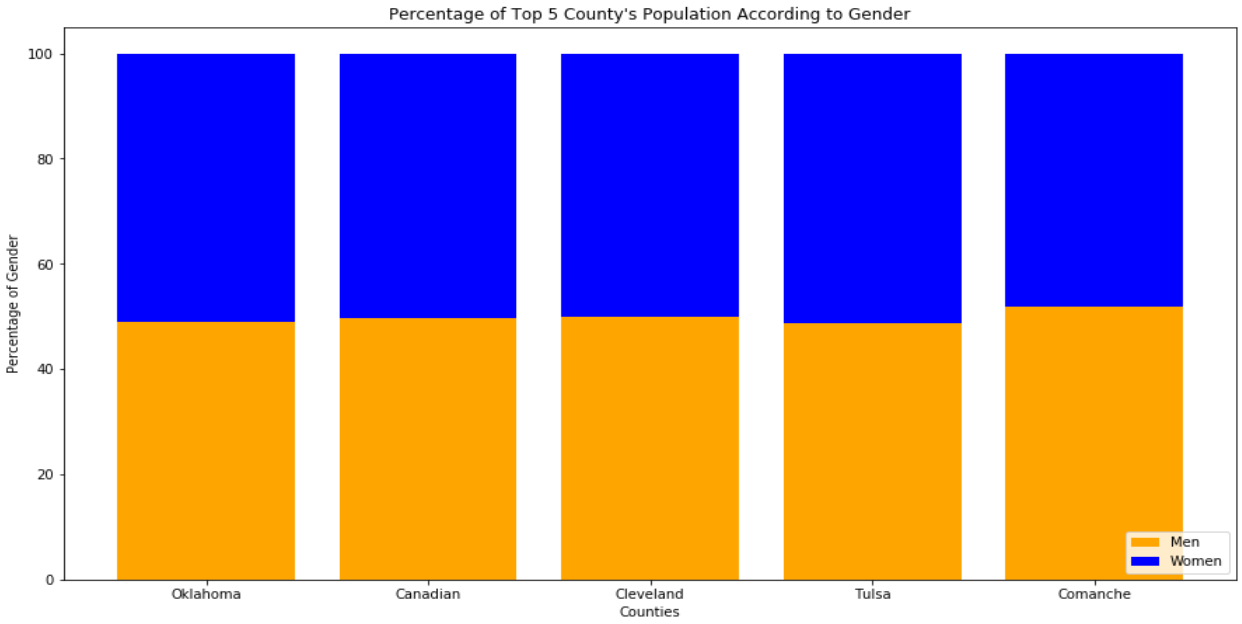

Figure 21. Gender Disparity in Top 5 Oklahoma Populated Counties

5. Further to analyze the gas excise tax collected by each counties of Oklahoma to see the hot spots where people usually refuel their vehicle, we firstly examined the vehicle registrations data to see the number of registered vehicles per county, which can be seen in Figure 22 and then using that information we went further and analyzed the gas excise data to get an estimate of the gas excise tax collected by the cities of these high vehicle registered counties. Some of the high paying gas excise tax cities can be seen in Figure 23.

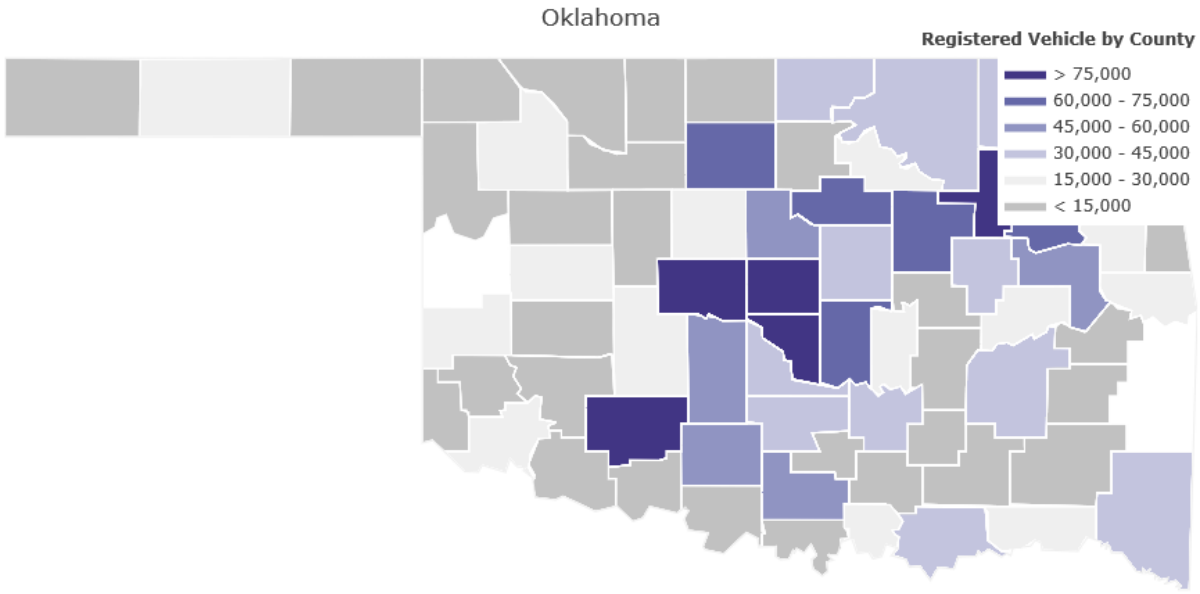

Figure 22. Number of Registered Vehicles per County

Figure 23. High Paying Gas Excise Tax Cities

Statistical Analysis

Statistical Analysis was used to support our claim that in the state of Oklahoma, top five counties are Oklahoma, Tulsa, Cleveland, Comanche and Canadian.

Scatterplot

I.

The scatterplot shown in Figure 24 below is about vehicle registered in the state of oklahoma and the amount of tax paid by the counties. Those two variables are important in deciding the vehicle usage and driven in particular county. We noticed that seven of the counties have been scattered differently as compared to others.

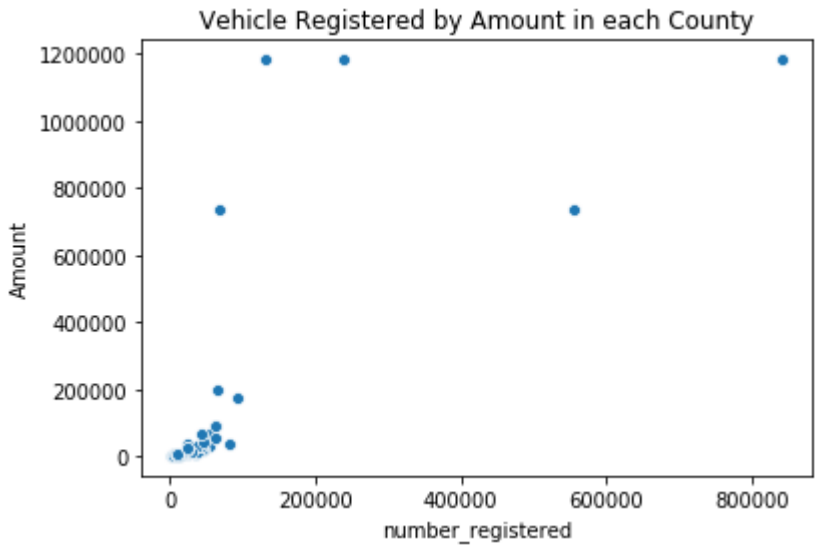

Figure 24. Vehicle registered by amount of counties

The above scatterplot dataset has been shown below to show which counties has been spread differently.

Sorted by number registered

County	number_registered	Amount
Oklahoma	840960	1.18216e+06
Tulsa	554962	737971
Cleveland	238680	1.18216e+06
Canadian	131894	1.18216e+06
Comanche	93221	172121

Sorted by Amount

County	number_registered	Amount
Canadian	131894	1.18216e+06
Cleveland	238680	1.18216e+06
Oklahoma	840960	1.18216e+06
Creek	68225	737971
Tulsa	554962	737971

Figure 25. Information of counties which has been shown uniquely above

II.

The scatterplot shown in Figure 26 below is how total population and income has been scattered in counties. This plot also gives insights about some counties which is exceptionally different in terms of population and income. We noticed that there are five counties which are exceptionally different place in the plot.

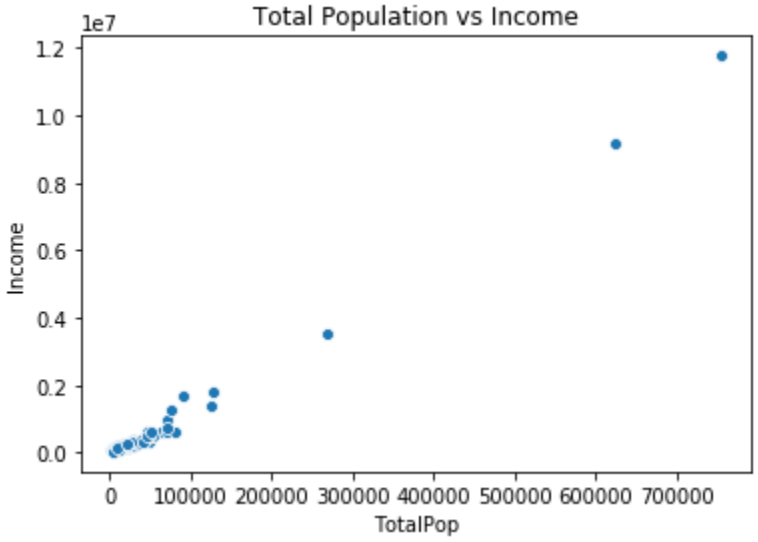

Figure 26. Total population vs income of counties

Above plot gives the insights about differently placed counties. Figure 27 gives the information about the county which are spread around on the plot as compared to others.

Sorted by Total Population

County	TotalPop	Income
Oklahoma	754480	11768506
Tulsa	623335	9196287
Cleveland	268614	3537547
Canadian	126193	1831119
Comanche	125531	1378625

Sorted by Income

County	TotalPop	Income
Oklahoma	754480	11768506
Tulsa	623335	9196287
Cleveland	268614	3537547
Canadian	126193	1831119
Rogers	89190	1672684

Figure 27. Information of counties which has been shown uniquely above

III.

The scatterplot in Figure 28 shows how counties have been distributed based on total population and number of vehicles registered. This plot also shows five counties which are placed differently.

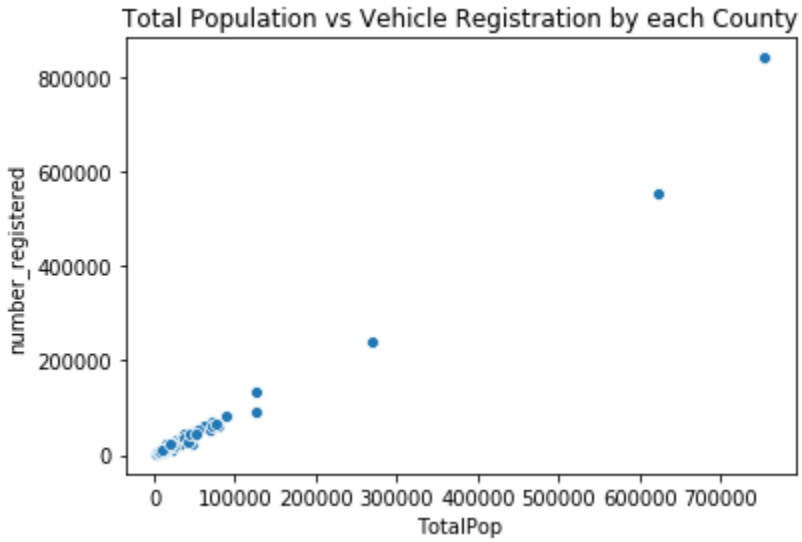

Figure 28. Total Population vs vehicle registered of county

Figure 29 shows the values which uniquely placed in the plot above. Here we have collected the counties name which stands in the top 5.

Sorted by Total Population

County	TotalPop	number_registered
Oklahoma	754480	840960
Tulsa	623335	554962
Cleveland	268614	238680
Canadian	126193	131894
Comanche	125531	93221

Sorted by Number Registered

County	TotalPop	number_registered
Oklahoma	754480	840960
Tulsa	623335	554962
Cleveland	268614	238680
Canadian	126193	131894
Comanche	125531	93221

Figure 29. Information of counties which has been shown uniquely above

IV.

The scatterplot shown in Figure 30 below gives information about total population and gas taxes paid by the counties in the state of Oklahoma. The five outlying counties can be seen in Figure 30 below.

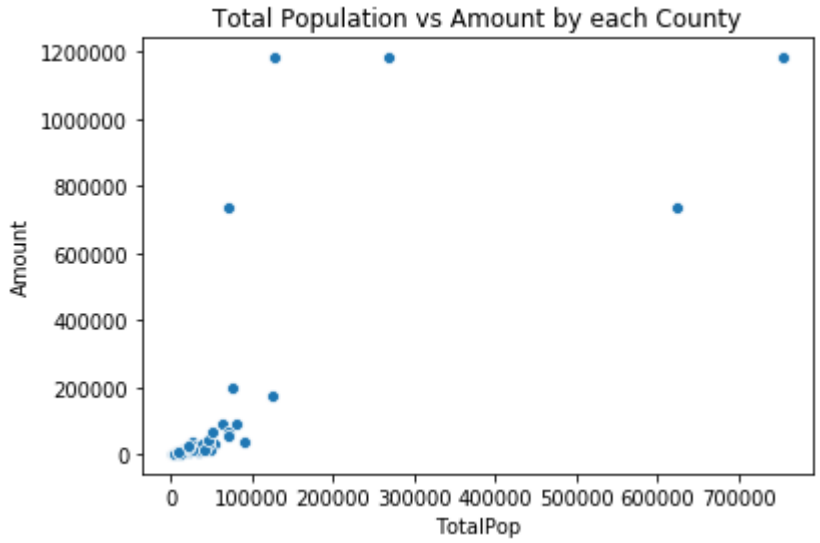

Figure 30. Total population vs gas tax amount paid by each county

Figure 31 shows the information about the above scatterplot. We can clearly determine the counties which are performing uniquely as compared to other counties.

Sorted by Total Population

County	TotalPop	Amount
Oklahoma	754480	1.18216e+06
Tulsa	623335	737971
Cleveland	268614	1.18216e+06
Canadian	126193	1.18216e+06
Comanche	125531	172121

Sorted by Amount

County	TotalPop	Amount
Canadian	126193	1.18216e+06
Cleveland	268614	1.18216e+06
Oklahoma	754480	1.18216e+06
Creek	70761	737971
Tulsa	623335	737971

Figure 31. Information of counties which has been shown uniquely above

V.

The scatter plot in Figure 32 is showing the values of driving percentage vs professional in each county. Here, we can say that the driving percentage increases the chances to have professionals in that county. This plot also gives unique counties which can be seen separately from other counties.

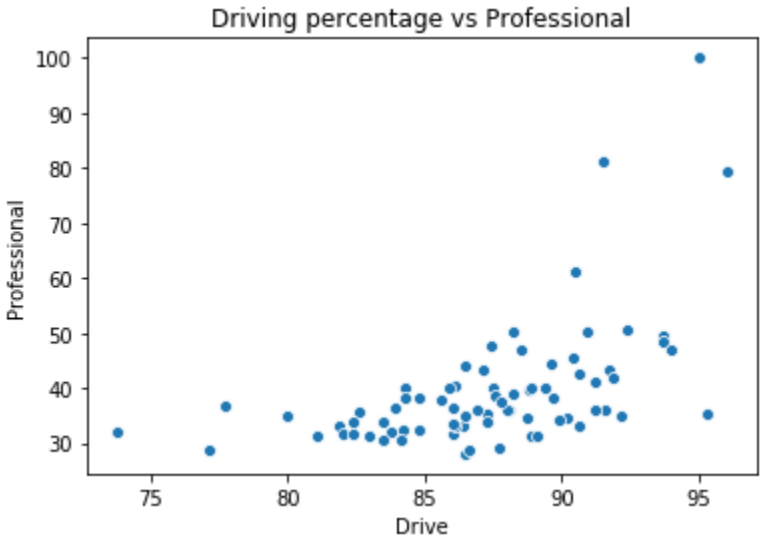

Figure 32. Driving vs Professionals percentage in each county

Table below clarifies the counties which are placed differently in the plot above. Cleveland county has 100% professionals and 95% driver in that county. This county is mainly contains highest professional as compared to other counties.

Sorted by Drive

County	Drive	Professional
Oklahoma	96	79.2
Garvin	95.3	35.4
Cleveland	95	100
Garfield	94	47
Comanche	93.7	48.3

Sorted by Professional

County	Drive	Professional
Cleveland	95	100
Tulsa	91.5	81.2
Oklahoma	96	79.2
Payne	90.5	61.2
Rogers	92.4	50.7

Figure 33. Information of counties which has been shown uniquely above

Pairplot

The plot in Figure 34 shows the additional information of the important column in our dataset. We can clearly distinguish counties based on the distribution in different plots. Moreover, we selected most frequent counties which were showing in each graph.

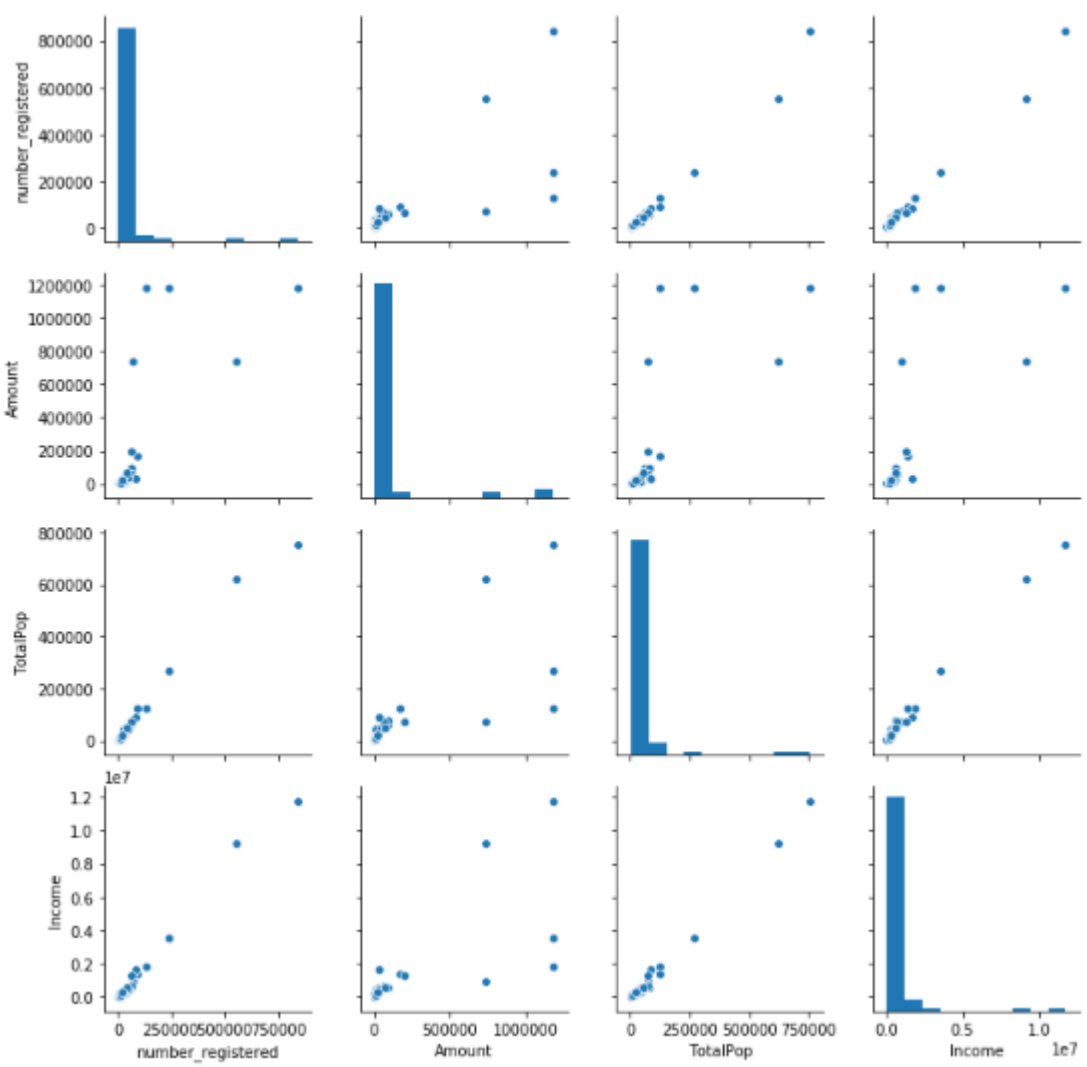

Figure 34. Distribution of important variables in the dataset

Linear Regression

Title: Predicting the income with the help of population variable.

Summary Statistics:

Variables	N (%)	Mean	Std. Dev	Range
Income	76 (100)	654600.38	1714039.61	31250 - 11768506
Total Population	76 (100)	49997.82	112874.41	2341 - 754480

Model:

Estimated Income = - 100959.74 + 15.11 * Total population

Fit of the model:

Adjusted R-square - 0.990

Thus, 99.0 % of variability in income is explained by total population.

Python Output:

OLS Regression Results

Dep. Variable:	Income	**R-squared:**	0.990
Model:	OLS	**Adj. R-squared:**	0.990
Method:	Least Squares	**F-statistic:**	7588.
Date:	Sat, 30 Nov 2019	**Prob (F-statistic):**	2.56e-76
Time:	20:41:29	**Log-Likelihood:**	-1021.9
No. Observations:	76	**AIC:**	2048.
Df Residuals:	74	**BIC:**	2053.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-1.01e+05	2.13e+04	-4.740	0.000	-1.43e+05	-5.85e+04
TotalPop	15.1119	0.173	87.110	0.000	14.766	15.458

Omnibus:	5.366	**Durbin-Watson:**	2.218
Prob(Omnibus):	0.068	**Jarque-Bera (JB):**	5.135
Skew:	-0.395	**Prob(JB):**	0.0767
Kurtosis:	3.999	**Cond. No.**	1.34e+05

Interpretation:

Each person is contributing $15 in income of the state.

Python code:

```
model = smf.ols('Income ~ TotalPop',data=join_income_pop)

result = model.fit()

result.summary()

result.params
```

Title: Predicting professionals which are in the driving variable.

Summary Statistics:

Variables	N (%)	Mean	Std. Dev	Range
Professional	76 (100)	39.55	11.63	28.1 - 100.0
Drive	76 (100)	87.30	4.20	73.8 - 96.0

Model:

Estimated professionals = -90.55 + 1.49 * Drive

Fit of the model:

Adjusted R-square - 0.289

Thus, 28.9 % of variability in professional is explained by drive.

Python output:

OLS Regression Results

Dep. Variable:	Professional	**R-squared:**	0.289
Model:	OLS	**Adj. R-squared:**	0.279
Method:	Least Squares	**F-statistic:**	30.08
Date:	Sat, 30 Nov 2019	**Prob (F-statistic):**	5.50e-07
Time:	20:41:31	**Log-Likelihood:**	-280.86
No. Observations:	76	**AIC:**	565.7
Df Residuals:	74	**BIC:**	570.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-90.5460	23.746	-3.813	0.000	-137.860	-43.232
Drive	1.4902	0.272	5.485	0.000	0.949	2.032

Omnibus:	61.513	**Durbin-Watson:**	1.883
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	330.261
Skew:	2.469	**Prob(JB):**	1.93e-72
Kurtosis:	11.939	**Cond. No.**	1.83e+03

Interpretation:

To achieve one percent professionals, state needs to have approximately 60% drivers. Because the intercept in the equation is -90.55. If, drive = 0, no driver in any state then the estimated professional value is -90.

Python code:

```
model2 = smf.ols('Professional ~ Drive',data=final_drive_prof)
result2 = model2.fit()
result2.summary()
```

The plot is showing the relation between drive and family work in top 5 counties which was selected from past findings.

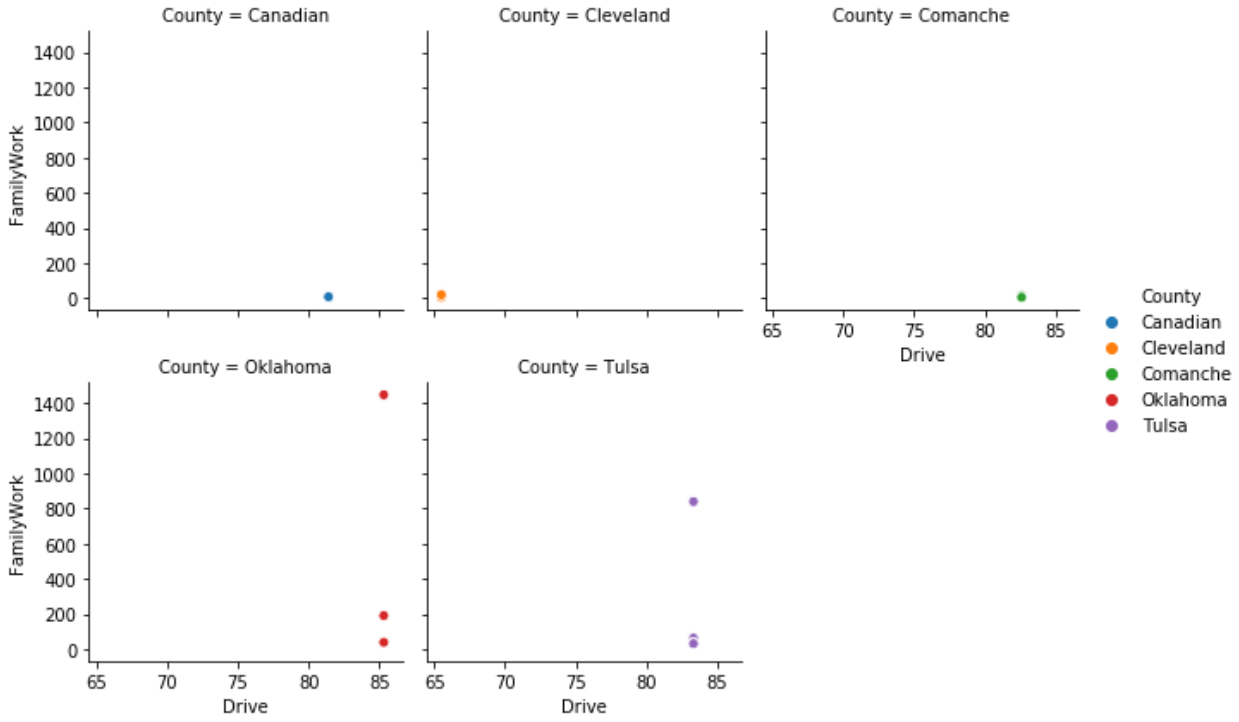

Figure 35. Relational plot between drive and family work

This plot provided an idea about how drive and family work has been distributed in five counties. All of the counties have more than 80% driving percentage. But they have different family work percentage which is making them differ from others. Oklahoma has most family work percentage than Tulsa.

Below graph is the visualization of the existing counties with more than 1% family work in their daily life. Oklahoma has more family work percentage than Comanche but both counties has almost the same number of cities in their respective county.

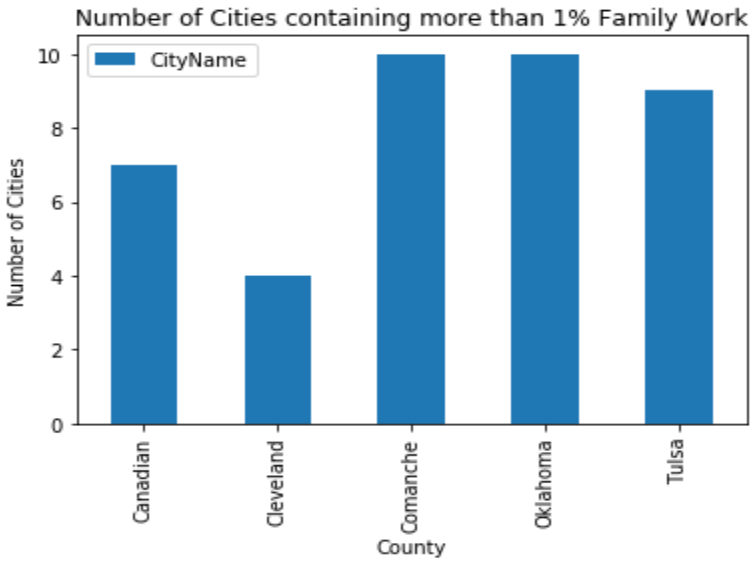

Figure 36. Each county has count of cities who has over 1% family work

Below graph provided the idea about family work and their profile. In this graph, whichever cities has a higher percentage of family work those cities have more professionals. Moreover, we can say that the higher post people have more family work.

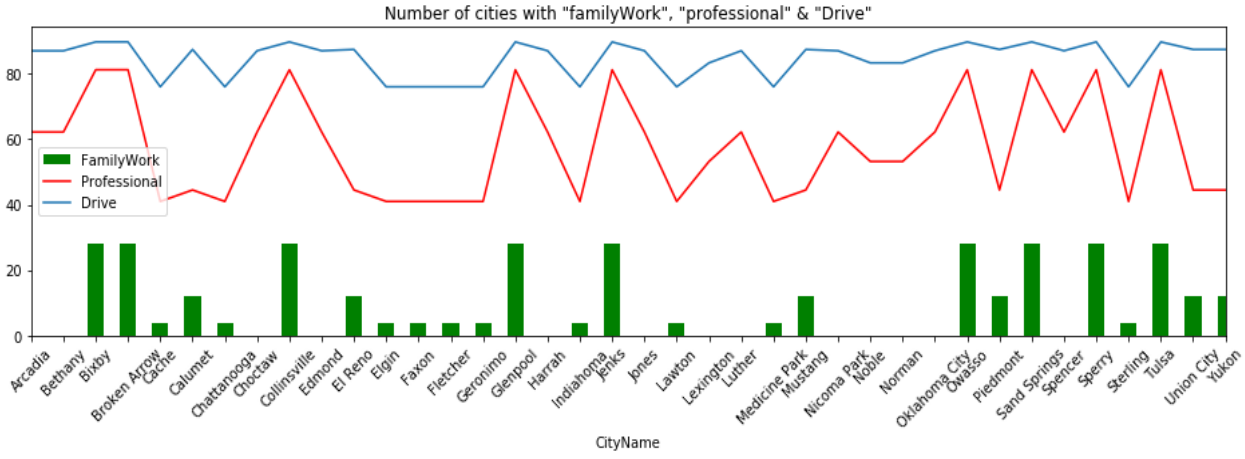

Figure 37. Cities having percentage graph in drive, family work and professionals.

Conclusion and Recommendations

Fueldup which is a startup company, provides fuel delivery service that brings fuel directly to you whether that is at your workplace, home or anywhere else. The only problem is that the idea is very new in the Oklahoma market and they currently don't know who their target customers are and what areas to target. So, the objective of this research was to explore numerous Oklahoma data, such as, Oklahoma demographic data, gas excise tax data and vehicle registration data, to determine areas that are generating high gas excise tax, areas where high income individuals reside, areas with high registered vehicles and others. Understanding our analysis will help the Fueldup company figure out their target customers and the areas that they should target which will generate most revenue.

The analysis conducted on the datasets mentioned above revealed five Oklahoma counties that stood out the most during our research: Oklahoma county, Tulsa county, Canadian county, Cleveland county, and Comanche county. Therefore, the following are our recommendations that can help the Fueldup company solve their business problems:

- During the analysis on the demographics data, we discovered that Oklahoma, Tulsa, Cleveland, Canadian, and Comanche county are the five counties in Oklahoma that has the most driving age population, therefore, Fueldup might focus on targeting customers in this area.
- We also discovered five counties from average time of people commuting for work is longest, which means their vehicle will consume more gas on a daily basis and they are more likely to utilize the Fueldup service because who would want to go fill their own gas at the end of the day after a long tiring commute to work. The five counties with high average commute time are: Lincoln, Pushmataha, Okfuskee, Logan, and McIntosh.

Therefore, we would recommend Fueldup to advertise their service more in these counties and target customers from these counties as they are more likely to utilize their service due to long commute time over waiting in long lines at gas stations at the end of a tiring day when they are in a hurry to return home.

- The last thing we would like to recommend to the Fueldup company is to also target areas that are resident to high income individuals as they will be more willing to pay a service fee for having gas delivered to them instead of going to gas stations. The counties we found with people who have an income more than the average income of individual in Oklahoma are: Canadian, Kingfisher, Rogers, Wagoner, and Cleveland.

Overall, the data we explored allowed us to gain some insight into Oklahoma counties,such as, where most vehicles are running, where high income society is, where most gas excise tax is generated, etc. Based on this information we were able to come up with some recommendations that might help the Fueldup company come up with solutions to their business problems. Further in-depth analysis can be conducted if we receive data for the current year in the future that might help further understand the business problems of Fueldup.

Appendix

Python Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.display import display_html
from IPython.display import display
pd.set_option('max_columns',51)

# Reading the files:
   # Demographic(file_2015.csv & file_2017.csv),
   # Vehicle Registrations(Vehicle_Registrations.csv)
   # Gas Excise(YTD_Gas_Excise.csv)
   # Zip Code(ZIP-COUNTY-FIPS_2017-06.csv)
file_2015 = pd.read_csv('acs2015.csv')
file_2017 = pd.read_csv('acs2017.csv')
vehicle_reg = pd.read_csv('Vehicle_Registrations.csv')
gas_excise = pd.read_csv('YTD_Gas_Excise.csv')
fips_zip_code = pd.read_csv('ZIP-COUNTY-FIPS_2017-06.csv')
ok_zip = pd.read_csv('OK_zipcode.csv')
```

Merging and Cleaning Demographic files (acs2015.csv, acs2017.csv)

```
# Retrieving only State = 'Oklahoma' data from the dataset
# data_2015
filt1 = file_2015["State"] == "Oklahoma"
file_2015 = file_2015.loc[filt1].reset_index()
file_2015.drop(columns='index',inplace=True)
# data_2017
filt3 = file_2017["State"] == "Oklahoma"
file_2017 = file_2017.loc[filt3].reset_index()
file_2017.drop(columns='index',inplace=True)
# Checking the shape of the file_2015 & file_2017.
print("File 2015: ",file_2015.shape)
print("File 2017: ",file_2017.shape)
# Checking data types present in both of the tables.
a = file_2015.dtypes.value_counts().to_frame("Count")
```

```
b = file_2017.dtypes.value_counts().to_frame("Count")
a1 = a.style.set_table_attributes("style='display:inline'").set_caption("File_2015")
b1 = b.style.set_table_attributes("style='display:inline'").set_caption("File_2017")
display_html(a1._repr_html_()+" "+b1._repr_html_(), raw=True)

# Checking if whole row or column is missing or not.
file_2015.dropna(how='all',inplace=True)  # Checking whole row.
file_2015.dropna(how='all',axis='columns',inplace=True)   # Checking whole column.
file_2017.dropna(how='all',inplace=True)
file_2017.dropna(how='all',axis='columns',inplace=True)

# Removing the "County" in each value which was attached after each county names.
file_2017['County'] = file_2017['County'].str.split(expand=True)[0]

# Changing two column names which are similar but column names are different.
file_2015.rename(columns={'CensusTract':"TractId","Citizen":"VotingAgeCitizen"},inplace=Tr
ue)

# Displaying both of the dataframes together.
a2 = file_2015.head(3).style.set_table_attributes("style='display:inline'").set_caption("File_2015
: Rows - 1046, Columns - 37")
b2 = file_2017.head(3).style.set_table_attributes("style='display:inline'").set_caption("File_2017
: Rows - 1046, Columns - 37")
display_html(a2._repr_html_()+" "+b2._repr_html_(), raw=True)

# Checking each columns is equal or not.
file_2015.columns == file_2017.columns

# Making the index in proper format using reset_index()
demographic_data = pd.concat([file_2015,file_2017],sort=False).reset_index()
demographic_data.drop(columns='index',inplace=True)
demographic_data.head()

# Remove rows with more than 60% of the dataset empty or NULL.
demographic_data.dropna(thresh = demographic_data.shape[1]*0.6,how='all',inplace=True)

# Fill NaN values with the minimum value present in their column
placing_null_1 = demographic_data['Income'].min()
placing_null_2 = demographic_data['IncomeErr'].min()
placing_null_3 = demographic_data['ChildPoverty'].min()
```

```
placing_null_4 = demographic_data['MeanCommute'].min()
demographic_data['Income'].fillna(value=placing_null_1,inplace=True)
demographic_data['IncomeErr'].fillna(value=placing_null_2,inplace=True)
demographic_data['ChildPoverty'].fillna(placing_null_3,inplace=True)
demographic_data['MeanCommute'].fillna(placing_null_4,inplace=True)
demographic_data.isna().sum().sum()

# Changing the data type of Income columns into 'Integer' datatype.
int_list = ['Income','IncomeErr','IncomePerCap','IncomePerCapErr']
demographic_data[int_list] = demographic_data[int_list].astype('int64')

# Successfully merged two demographic datasets.
Demographic_data.shape

# Final demographic data
demographic_data.head(3)
```

Merging and cleaning zip code files (Ok_zipcode.csv, ZIP-COUNTY-FIPS_2017-06.csv)

```
ok_zip.head(2)
Ok_zip.shape
fips_zip_code.head(2)
Fips_zip_code.shape

# Merge ok_zip and fips_zip_code on zip by inner join to get only OK zip code
# Drop unnecessary/duplicate columns
zip_code = ok_zip.merge(fips_zip_code, left_on='zip', right_on='ZIP', how='inner')
zip_code.drop(columns=['ZIP', 'COUNTYNAME', 'STATE', 'CLASSFP'], inplace=True)
zip_code.rename(columns={'zip':'ZipCode', 'city':'City', 'state_id':'StateID',
'state_name':'StateName', 'county_name':'CountyName', 'STCOUNTYFP':'CountyFipsCode'},
inplace=True)
zip_code.head()
```

Merging and cleaning zip code and gas excise files (zip_code, YTD_Gas_Excise.csv)

```
# Removing the last row, which is the total amount.
gas_excise.drop(labels=7008,inplace=True)

# In the 'Name' column, we are separating cities name and inserting into new column.
a = gas_excise['Name'].str.split(expand=True)
```

```
b = a.loc[:,[2,3]]
b[3] = b[3].fillna("")
city_name = b[2]+" "+b[3]

# Concatenate the new column with the existing dataframe and replacing the 'Name' column with 'City_Name' column.
concat_city_name = pd.concat([gas_excise,city_name],axis=1)
concat_city_name.rename(columns={0:'CityName'},inplace=True)
concat_city_name.drop(labels='Name',axis=1,inplace=True)
concat_city_name['CityName'] = concat_city_name['CityName'].str.strip().str.lower().str.title()
concat_city_name['CityName'].replace({'Ft Cobb':"Fort Cobb",'Ft Gibson':'Fort Gibson'},inplace=True)
new_col_order = ['Copo','CityName','Distribution Date','Amount']
new_gas_excise = concat_city_name[new_col_order]
new_gas_excise.head(2)
new_gas_excise.shape
zip_code.head(2)
zip_code.shape
county_gas_excise = new_gas_excise.merge(zip_code, left_on='CityName', right_on='City', how='left')
county_gas_excise.drop(columns=['City'], inplace=True)
county_gas_excise.rename(columns={'Distribution Date':'DistributionDate'}, inplace=True)
order = ['Copo', 'CityName', 'ZipCode', 'CountyName', 'CountyFipsCode', 'StateID', 'StateName', 'DistributionDate', 'Amount']
county_gas_excise = county_gas_excise[order]
county_gas_excise.head(2)

# Changing datatype of ["Amount", "Copo"] column from [object, float] to [float, int] .
county_gas_excise['Amount'] = county_gas_excise['Amount'].str.replace(",","")
county_gas_excise['Amount'] = county_gas_excise['Amount'].astype(float)
county_gas_excise['Copo'] = county_gas_excise['Copo'].astype('int64')
County_gas_excise.dtypes

# Remove rows with more than 60% of the dataset empty or NULL.
county_gas_excise.dropna(thresh = county_gas_excise.shape[1]*0.6,how='all',inplace=True)
```

Merging and cleaning zip code and vehicle registration files (zip_code, Vehicle_Registrations.csv)

```
zip_code.head(2)
vehicle_reg.head(2)
county_vehicle_reg = zip_code.merge(vehicle_reg, left_on='CountyName',
right_on='county_name', how='outer')
county_vehicle_reg.drop(columns=['county_name'], inplace=True)
county_vehicle_reg.head(2)

# Some rows in vehicle registration dataset didn't contain county name so drop those columns
county_vehicle_reg.dropna(inplace=True)
```

Merging all datasets (demographic_data, county_gas_excise, county_vehicle_reg)

```
demographic_data.shape
county_vehicle_reg.shape
demographic_vehiclereg = demographic_data.merge(county_vehicle_reg, left_on='County',
right_on='CountyName', how='left')
demographic_vehiclereg.head(2)
Demographic_vehiclereg.shape

# Drop 25 rows that were null and drop duplicate column
demographic_vehiclereg.dropna(inplace=True)
demographic_vehiclereg.drop(columns=['StateID', 'StateName', 'CountyName'], inplace=True)
demographic_vehiclereg.head(2)
county_gas_excise.shape
county_gas_excise.head(2)

final_dataset =
pd.merge(groupby_sorted_demographic_vehiclereg,groupby_sorted_county_gas_excise,\
                    left_on=['County','City','ZipCode'],\
                       right_on=['CountyName','CityName','ZipCode'])
```

Get the top five counties to see the relation between unemployment and meancommute

```
# Created new column named DrivingAgeCitizen to get only citizen who are able to drive
dem_data['DrivingAgeCitizen'] = dem_data['TotalPop'] - dem_data['VotingAgeCitizen']
county_group = dem_data.groupby('County').agg({'DrivingAgeCitizen':'sum',
'Unemployment':'mean', 'MeanCommute': 'mean'}) # group by counties
bar_chart = county_group.sort_values(by='DrivingAgeCitizen',
ascending=False).head().reset_index()

#Sort values by Unemployment rate to see relation between Unemployment and MeanCommute
bar_chart.sort_values(by='Unemployment')
fig, ax = plt.subplots(figsize=(18, 5))
sns.barplot(x='Unemployment', y='MeanCommute', data=bar_chart, hue='County',
        estimator=np.mean, ci=None, ax=ax);

# Created new column named DrivingAgeCitizen to get only citizens who are able to drive
#Check to see if poverty has any influence on driving
dem_data['DrivingAgeCitizen'] = dem_data['TotalPop'] - dem_data['VotingAgeCitizen']
county_group = dem_data.groupby('County').agg({'DrivingAgeCitizen':'sum', 'Poverty':'mean',
'Drive': 'mean'}) # group by counties
bar_chart = county_group.sort_values(by='DrivingAgeCitizen',
ascending=False).head().reset_index()

#Sort values by Unemployment rate to see the relation between Unemployment and
MeanCommute
bar_chart.sort_values(by='Poverty')

# Created new column named DrivingAgeCitizen to get only citizens who are able to drive
# Check to see if income has any influence on driving
dem_data['DrivingAgeCitizen'] = dem_data['TotalPop'] - dem_data['VotingAgeCitizen']
county_group = dem_data.groupby('County').agg({'DrivingAgeCitizen':'sum',
'IncomePerCap':'sum', 'MeanCommute': 'mean'}) # group by counties
bar_chart = county_group.sort_values(by='DrivingAgeCitizen',
ascending=False).head().reset_index()

#Sort values by Unemployment rate to see the relation between Unemployment and
MeanCommute
bar_chart.sort_values(by='IncomePerCap')
```

Which county has the highest number of employed individuals?

```
county_group = dem_data.groupby('County').agg({'TotalPop':'sum', 'Employed':'sum'}) # group by counties
bar_chart = dem_data.sort_values(by='TotalPop', ascending=False).head().reset_index()

#Sort values by employment rate to find counties with highest employment
cols = ['County', 'TotalPop', 'Employed']
bar_chart.loc[:,cols].sort_values(by='Employed', ascending=False)
```

What are the top 5 counties with driving age citizens?

```
# Created new column named DrivingAgeCitizen to get only citizen who are able to drive
demographic_data['DrivingAgeCitizen'] = demographic_data['TotalPop'] - demographic_data['VotingAgeCitizen']
countypop_group = demographic_data.groupby('County').agg({'DrivingAgeCitizen':'sum'}) # group by counties
top5_pop = countypop_group.sort_values(by='DrivingAgeCitizen', ascending=False).head().plot(kind='bar',title='Top 5 Counties with Driving Age Citizens', figsize=(10,5))

# Add labels to chart
top5_pop.set_ylabel('Population')
for p in top5_pop.patches:
    top5_pop.annotate(str(p.get_height()), xy=(p.get_x(), p.get_height()+5000))
countypop_group.sort_values(by='DrivingAgeCitizen', ascending=False).head()
```

Which county has the highest mean commute time of people commuting for work?

```
countycommute_group = demographic_data.groupby('County').agg({'MeanCommute':'mean'})
ax = countycommute_group.sort_values(by='MeanCommute', ascending=False).head().plot(kind='bar')
ax.set_ylim(26,29)
```

Which county has employed people who have a salary more than the average salary?

```
okla_avg_income = demographic_data['Income'].mean()
county_avg_income = demographic_data.groupby('County').agg({'Income':'mean'})
avg_high_income = county_avg_income > okla_avg_income
income_without_nan = county_avg_income[avg_high_income].dropna()
income_without_nan.sort_values(by='Income',
ascending=False).head(10).plot(kind='bar').set_ylim(50000, 70000)
income_without_nan.sort_values(by='Income', ascending=False).head(10)

# Removing the last row, which is the total amount.
ytd_gas.drop(labels=7008,inplace=True)

# In the 'Name' column, we are separating cities name and inserting into new column.
a = ytd_gas['Name'].str.split(expand=True)
b = a.loc[:,[2,3]]
b[3] = b[3].fillna("")
city_name = b[2]+" "+b[3]

# Concatenate the new column with the existing dataframe and replacing the 'Name' column with
'City_Name' column.
concat_city_name = pd.concat([ytd_gas,city_name],axis=1)
concat_city_name.rename(columns={0:'City_Name'},inplace=True)
concat_city_name.drop(labels='Name',axis=1,inplace=True)
concat_city_name['City_Name'] = concat_city_name['City_Name'].str.strip().str.lower().str.title()
concat_city_name['City_Name'].replace({'Ft Cobb':"Fort Cobb",'Ft Gibson':'Fort
Gibson'},inplace=True)
new_col_order = ['Copo','City_Name','Distribution Date','Amount']
new_ytd_gas = concat_city_name[new_col_order]

# Removing the "County" in each value which was attached after each county names.
ok_fips['COUNTYNAME'] = ok_fips['COUNTYNAME'].str.split(expand=True)[0]

# Merge vehicle registration data with ok_fips(similar to ok zip but with county id) data
ok_zip_veh_reg = pd.merge(veh_registration, ok_fips, left_on=['county_name'],
right_on=['COUNTYNAME'], how='outer')
ok_zip_veh_reg = ok_zip_veh_reg.drop(['COUNTYNAME'], axis=1)
ok_zip_veh_reg.dropna(how='any', axis='index', inplace=True)
ok_zip_veh_reg['ZIP'] = ok_zip_veh_reg['ZIP'].astype(int)
ok_zip_veh_reg['STCOUNTYFP'] = ok_zip_veh_reg['STCOUNTYFP'].astype(int)
```

```
ok_zip_veh_reg.head()

# Created graph for number of registered vehicle by Oklahoma counties
num_reg = ok_zip_veh_reg['number_registered'].tolist()
fips = ok_zip_veh_reg['STCOUNTYFP'].tolist()
fig = ff.create_choropleth(
   fips=fips, values=num_reg, scope=['OK'], county_outline={'color': 'rgb(255,255,255)', 'width':
0.5},
   round_legend_values=True, legend_title='Registered Vehicle by County', title='Oklahoma')
fig.layout.template = None
fig.show()

filt1 = dem_data['County'].isin(['Oklahoma','Tulsa','Cleveland','Canadian'])
imp_counties = dem_data.loc[filt1]
imp_counties.groupby('County').agg({'Employed':'sum','TotalPop':'sum','Income':'mean','Professi
onal':'sum',\
                                    'FamilyWork':'sum','Drive':'sum'})\
.sort_values(['Employed','FamilyWork'],ascending=[False,False]).round(1).rename(columns={'In
come':'Mean_Income'})
```

Manipulating the Vehicle Registration dataset

```
# Checking the four largest counties who has maximum number of vehicles registered
veh_regs.nlargest(4,'number_registered')

veh_regs.nlargest(10,'number_registered').plot(kind='barh',title='Number of vehicle registered
over 100k')

# Collecting Top 3 cities who paid more than $ 200,000 taxes on gas.
new_ytd_gas.groupby('City_Name').agg({'Amount':'sum'}).nlargest(3,'Amount').round(0)

# Top 10 cities paid the highest taxes in the recent date.
col_list = ['City_Name','Distribution Date','Amount']
new_ytd_gas[col_list].sort_values('Distribution
Date',ascending=False).drop_duplicates(subset='City_Name')\
        .drop(columns='Distribution
Date').sort_values('Amount',ascending=False).head(10).set_index('City_Name')\
        .plot(kind='line',figsize=(10,4),title='According to recent Distribution Date, taxes paid by
the cities')
```

```
# Separating recent and first month of gas excise tax paid by the cities in Oklahoma.
a = new_ytd_gas.sort_values('Distribution Date').drop_duplicates(subset='City_Name')\
        .sort_values('City_Name').reset_index()
a.rename(columns={'Amount':'Initial_Amt','City_Name':'Cities'},inplace=True)

b = new_ytd_gas.sort_values('Distribution
Date',ascending=False).drop_duplicates(subset='City_Name')\
            .sort_values('City_Name').reset_index()
b.rename(columns={'Amount':'Recent_Amt'},inplace=True)

c = pd.concat([a,b],axis=1)

# Plotting the graph which is showing the jump in one year excise tax.
c['first_last_payment_diff'] = ((c['Recent_Amt'] - c['Initial_Amt'])*100/73052)
c.sort_values('first_last_payment_diff',ascending=False).drop(columns=['index','Copo','City_Na
me']).set_index('Cities')\
      .head(10)['first_last_payment_diff'].plot(kind='bar',figsize=(14,4),title='One year % diff. of
gas tax contribution of cities')

# Plotting the graph which is showing the comparison of latest and previous month comparison
in percentage
new_ytd_gas['Amount_diff_from_last_month'] = new_ytd_gas['Amount'].pct_change()*100
new_ytd_gas.drop_duplicates(subset='City_Name',keep='last').sort_values('Amount',ascending=
False).set_index('City_Name')\

.head(10)['Amount_diff_from_last_month'].plot(kind='line',figsize=(15,4),title='Previous month
% diff. in taxes')

# Monthly trend on paying the gas taxes
print(new_ytd_gas.resample('M',on='Distribution Date').agg({'Amount':'sum'}))

print(new_ytd_gas.resample('M',on='Distribution Date').agg({'Amount':'sum'}).\
plot(kind='line',title='Monthly trend of Tax amount from 2018 to 2019',figsize=(8,3)))

# Reading the top 5 cities from the dataset based on amount.
col_list = ['Oklahoma City','Tulsa','Norman','Broken Arrow','Lawton']
yy = new_ytd_gas['City_Name'].isin(col_list)
x = new_ytd_gas.loc[yy]
x['Year_Month'] = x['Distribution Date'].dt.to_period('M')
```

```
# Plotting the top five cities taxes paid to the state in each month.
fig, ax = plt.subplots(figsize=(18, 5))
sns.barplot(x='Year_Month',y='Amount',data=x,hue='City_Name',ci=None, ax=ax)\
        .set_title('Top 5 Cities based on the Amount contributed in the Gas Excise Tax')
```

Statistical Analysis

```
final_dataset.head(3)
```

Creating vehicle registration and tax amount column paid by each county

```
# Preparing dataset to get proper numbers of vehicle registered in County using 'Sorting'
sorted_vehregs =
final_dataset[['County','number_registered']].sort_values(['County','number_registered'],ascendin
g=[True,False])
sorted_vehregs.head(3)

# Filtered the first vehicle number registered by each county.
filt_regis_each_county = sorted_vehregs.groupby('County').agg({'number_registered':'first'})
filt_regis_each_county.head(3)

# Preparing dataset to get proper numbers of tax amount paid by each County using 'Sorting' &
'Drop_duplicates'
sorted_amount =
final_dataset[['County','DistributionDate','Amount']].sort_values(['County','DistributionDate','A
mount']\
                                                    ,ascending=[True,True,False])\
        .drop_duplicates(subset=['County','DistributionDate'])
sorted_amount.head(3)

# Filtered the sum of the amount by each county and their respective distribution date.
filt_amount_each_distdate =
sorted_amount.groupby(['County','DistributionDate']).agg({'Amount':'first'}).reset_index()
amount_each_county = filt_amount_each_distdate.groupby('County').agg({'Amount':'sum'})
amount_each_county.head(3)

# Merged the above two dataframe 'amount_each_county' & 'filt_regis_each_county' using
'JOIN' method.
```

```
joined_vehregs_amount_by_county =
filt_regis_each_county.join(amount_each_county,how='inner')
joined_vehregs_amount_by_county.head(3)
```

Creating total population and income column by each county

```
# Preparing dataset to get proper numbers of population in County using 'Sorting' &
'Drop_duplicates'
sorted_totalpop =
final_dataset[['TractId','County','TotalPop']].sort_values(['TractId','County','TotalPop'],\
                                                    ascending=[True,True,False])\
                        .drop_duplicates(subset=['TractId','County'])
sorted_totalpop.head(3)

# Filtered the sum of total population by each county.
filt_totalpop = sorted_totalpop.groupby('County').agg({'TotalPop':'sum'})
filt_totalpop.head(3)

# Preparing dataset to get proper numbers of Income by each County using 'Sorting' &
'Drop_duplicates'
sorted_income =
final_dataset[['TractId','County','Income']].sort_values(['TractId','County','Income'],\
                                                    ascending=[True,True,False])\
                        .drop_duplicates(subset=['TractId','County'])
sorted_income.head(3)

# Filtered the sum of the income by each county.
filt_income = sorted_income.groupby('County').agg({'Income':'sum'})
filt_income.head(3)

# Merged the above two dataframe 'filt_totalpop' & 'filt_income' using 'JOIN' method.
join_income_pop = filt_totalpop.join(filt_income,how='inner')
join_income_pop.head(3)

# Merged the two dataframe 'joined_vehregs_amount_by_county' & 'join_income_pop' using
'JOIN' method.
joined_final_df = joined_vehregs_amount_by_county.join(join_income_pop,how='inner')
joined_final_df.head(3)
```

Scatterplot

- **Number of Vehicle Registered vs Amount.**
- **Total Population vs Income.**
- **Total Population vs Vehicle Registered.**
- **Total Population vs Amount.**

```
# All variable relationships can be seen in pairplot.
sns.pairplot(joined_final_df.reset_index(),height=2.5)
```

Number of Vehicle Registered vs Amount.

```
sns.scatterplot(x='number_registered',y='Amount',data=joined_final_df)\
                                        .set_title('Vehicle Registered by Amount in each County')

a = joined_vehregs_amount_by_county.sort_values('number_registered',ascending=False)
b = joined_vehregs_amount_by_county.sort_values('Amount',ascending=False)
a2 = a.head(5).style.set_table_attributes("style='display:inline'").set_caption("Sorted by number
registered")
b2 = b.head(5).style.set_table_attributes("style='display:inline'").set_caption("Sorted by
Amount")
display_html(a2._repr_html_()+" "+b2._repr_html_(), raw=True)
```

Total Population vs Income.

```
final_dataset.head(3)

sns.scatterplot(x='TotalPop',y='Income',data=joined_final_df).set_title('Total Population vs
Income')

a = joined_final_df[['TotalPop','Income']].sort_values('TotalPop',ascending=False)
b = joined_final_df[['TotalPop','Income']].sort_values('Income',ascending=False)
a2 = a.head(5).style.set_table_attributes("style='display:inline'").set_caption("Sorted by Total
Population")
b2 = b.head(5).style.set_table_attributes("style='display:inline'").set_caption("Sorted by
Income")
display_html(a2._repr_html_()+" "+b2._repr_html_(), raw=True)
```

Total Population vs Vehicle Registered.

```
sns.scatterplot(x='TotalPop',y='number_registered',data=joined_final_df)\
```

```
    .set_title('Total Population vs Vehicle Registration by each County')

a = joined_final_df[['TotalPop','number_registered']].sort_values('TotalPop',ascending=False)
b =
joined_final_df[['TotalPop','number_registered']].sort_values('number_registered',ascending=Fal
se)
a2 = a.head(5).style.set_table_attributes("style='display:inline'").set_caption("Sorted by Total
Population")
b2 = b.head(5).style.set_table_attributes("style='display:inline'").set_caption("Sorted by Number
Registered")
display_html(a2._repr_html_()+" "+b2._repr_html_(), raw=True)
```

Total Population vs Amount.

```
sns.scatterplot(x='TotalPop',y='Amount',data=joined_final_df)\
    .set_title('Total Population vs Amount by each County')

a = joined_final_df[['TotalPop','Amount']].sort_values('TotalPop',ascending=False)
b = joined_final_df[['TotalPop','Amount']].sort_values('Amount',ascending=False)
a2 = a.head(5).style.set_table_attributes("style='display:inline'").set_caption("Sorted by Total
Population")
b2 = b.head(5).style.set_table_attributes("style='display:inline'").set_caption("Sorted by
Amount")
display_html(a2._repr_html_()+" "+b2._repr_html_(), raw=True)

axx1 = joined_final_df[['number_registered','Amount']]
axx2 = joined_final_df[['TotalPop','Income']]
axx3 = joined_final_df[['TotalPop','number_registered']]
axx4 = joined_final_df[['TotalPop','Amount']]

axx1.plot(x='number_registered',y='Amount',kind='scatter')
```

Linear Regression Model

- How much each person contributing in income?

```
joined_final_df.describe().T.round(2)

model = smf.ols('Income ~ TotalPop',data=join_income_pop)

result = model.fit()
```

```
result.summary()

result.params
```

Conclusion:

Income = -100959.74 + 15.11 * Total Population

- Each person is adding $ 15 in income.

- Thus, 99.0 % of variability in Income is explained by Total Population

```
# Preparing dataset to get proper numbers of professional and Driver in County using 'Sorting' & 'Drop_duplicates'
sorted_prof_drive =
final_dataset[['County','Professional','Drive']].sort_values(['County','Professional','Drive']\
                                                    ,ascending=[True,False,False])\
                        .drop_duplicates(subset=['County','Professional','Drive'])
sorted_prof_drive.head(3)

# Filtered the first value of the Drive by each county.
filt_drive =
sorted_prof_drive[['County','Drive']].sort_values(['County','Drive'],ascending=[True,False])
group_drive = filt_drive.groupby('County').agg({'Drive':'first'})
group_drive.head(3)

# Filtered the first value of the Professional by each county.
filt_prof =
sorted_prof_drive[['County','Professional']].sort_values(['County','Professional'],ascending=[True,False])
group_prof = filt_prof.groupby('County').agg({'Professional':'first'})
group_prof.head(3)

# Merged the two dataframe 'group_drive' & 'group_prof' using 'JOIN' method.
final_drive_prof = group_drive.join(group_prof,how='inner')
final_drive_prof.head(3)
```

Scatterplot

- Number of driver vs Professional

```
sns.scatterplot(x='Drive',y='Professional',data=final_drive_prof).set_title('Driving percentage vs Professional')
```

```
a = final_drive_prof.sort_values('Drive',ascending=False)
b = final_drive_prof.sort_values('Professional',ascending=False)
a2 = a.head(5).style.set_table_attributes("style='display:inline'").set_caption("Sorted by Drive")
b2 = b.head(5).style.set_table_attributes("style='display:inline'").set_caption("Sorted by
Professional")
display_html(a2._repr_html_()+" "+b2._repr_html_(), raw=True)
```

Model

- How many drivers are contributing to professionals?

```
model2 = smf.ols('Professional ~ Drive',data=final_drive_prof)

result2 = model2.fit()

result2.summary()

result2.params
```

Conclusion:

- We can say that to start seeking professional, we need 90 minimum driver then it will contribute towards professionals.

- Thus, 28.9% of variability in Professional is explained by Drive.

Target Cities within top 5 Counties

- Counties ---- 'Oklahoma','Tulsa','Cleveland','Comanche','Canadian'

```
# Sorting five counties from all the available counties in our dataset.
target_county = ['Oklahoma','Tulsa','Cleveland','Comanche','Canadian']
target = final_dataset[final_dataset['County'].isin(target_county)]
target.head(3)

# Grouping rows to get the exact values of cities.
a =
target.groupby(['TractId','County','CityName','ZipCode']).agg({'FamilyWork':'first','Drive':'first'}
).reset_index()
a.head()

# Grouping again to get the family work vs drive plot in each county
e = a.groupby(['County','CityName']).agg({'FamilyWork':'sum','Drive':'first'}).reset_index()
```

```
e.head()

# Plotting relational plot to compare with top 5 counties.
sns.relplot(y='FamilyWork',x='Drive',data=e,hue='County',kind='scatter',col='County',col_wrap=
3,height=3)

# Sorting family work more than 1% to see how many cities inside each county.
f = e[e['FamilyWork'] > 1.0].sort_values('FamilyWork',ascending=False)

# Grouping counties based on count of cities to plot which county has maximum number of
cities.
g = f.groupby('County').agg({'CityName':'count'}).plot(kind='bar',\
                                        title='Number of Cities containing more than 1% Family Work')
g.set_ylabel('Number of Cities')

# Grouping counties, cities and zip code to get the family work, professionals and drive.
top_cities =
target.groupby(['TractId','County','CityName','ZipCode']).agg({'TotalPop':'sum','Income':'sum','Pr
ofessional':'first','Drive':'first',\
                                    'FamilyWork':'first'}).reset_index().\

sort_values(['Income','Professional','Drive','FamilyWork'],ascending=[False,False,False,False])

# Manipulating family work column to visualize in the plot.
top_cities['FamilyWork'] = top_cities['FamilyWork']*40

# Grouping cities to show how many professionals, drive and family work.
f = top_cities.groupby('CityName').agg({'Professional':'first','Drive':'first','FamilyWork':'first'})
f['FamilyWork'].plot(color='green',kind='bar',figsize=(16,4),legend = True,title='Number of
cities with "familyWork", "professional" & "Drive"')\
                            .tick_params(axis='x',rotation=45)
f['Professional'].plot(kind='line',figsize=(16,4),legend=True,color='red').tick_params(axis='x',rot
ation=45)
f['Drive'].plot(kind='line',figsize=(16,4),legend=True).tick_params(axis='x',rotation=45)
```

References

[1] KyleAustinYoung. (2019, September 6). Cheapest Food Delivery Service? Here's a pricing comparison! *Food Delivery Guru.*

Retrieved from https://fooddeliveryguru.com/cheapest-food-delivery-service/.

[2] "Motor Fuel Forms." *Oklahoma Tax Commission - Motor Fuel*, Oklahoma Tax Commission.

Retrieved from www.ok.gov/tax/Forms_&_Publications/Forms/Motor_Fuel/#.

[3] "Motor Vehicle." *Oklahoma Tax Commission - Motor Vehicle*, Oklahoma Tax Commission.

Retrieved from www.ok.gov/tax/Individuals/Motor_Vehicle/.

[4] Neutrino, Muon. "US Census Demographic Data." *Kaggle*, Kaggle, 3 Mar. 2019.

Retrieved from www.kaggle.com/muonneutrino/us-census-demographic-data.

[5] Ofer, Dan. "US Zip Codes to County State to FIPS Crosswalk." *Kaggle*, Kaggle, 18 Mar. 2018.

Retrieved from www.kaggle.com/danofer/zipcodes-county-fips-crosswalk.

[6] "Oklahoma ZIP Code List." *ZIP Code List for Oklahoma*

Retrieved from www.zipcodestogo.com/Oklahoma/.

[7] Fueldup, Home.

Retrieved from https://www.fueldup.com/.