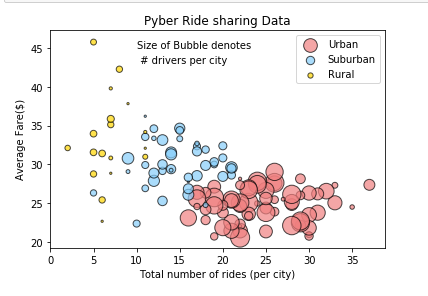
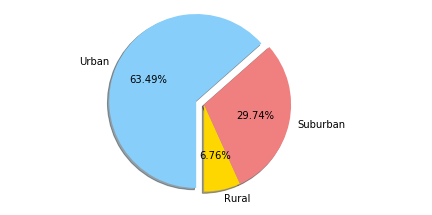
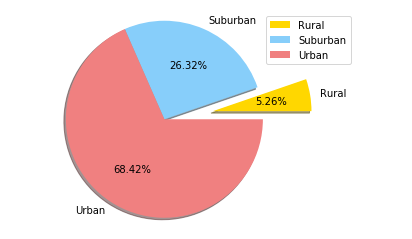
Pyber solutions

**PYBER Analysis**

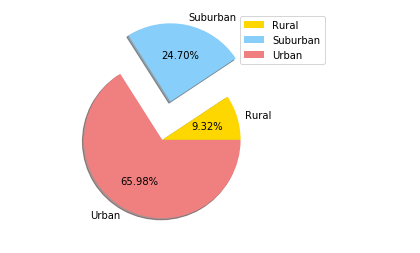
1. There is a correlation between number of rides and the average fare per city and total number of drivers per city.
2. **Urban cities** in general have lower average fares as seen below. Of total fares highest at 63.49% in the Pie chart below.
3. **Rural cities** had the higher average fare per city the number of drivers are lesser and also the total number of rides are less as compared to Urban cities, the rural cities had the smallest segment of the total fares by city pie chart at 6.68%.
4. Urban cities had the largest percentage of total rides.
5. Urban cities had the largest percentage of drivers and that’s why the average fares are lower due to competition . The Urban cities have more drivers and more riders



1. # % of total fares by City Type
2. #Total rides by city type.



1. #Total drivers by city type.



**Python Program**

# Dependencies

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

#Import city and and ride data csv files.

#City data

#color scheme dictionary and preview

color\_scheme = {'Gold':'#FFD700', 'Light Sky Blue':'#87CEFA', 'Light Coral':'#F08080'}

colors = ["Gold", "lightcoral", "lightskyblue"]

# sns.palplot(sns.color\_palette(color\_scheme.values()))

citydata = "generated\_data/city\_data.csv"

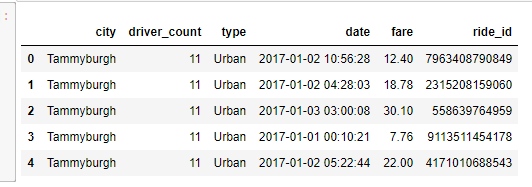
rides = "generated\_data/ride\_data.csv"

city\_data = pd.read\_csv(citydata)

ride\_data = pd.read\_csv(rides)

merged = pd.merge(city\_data, ride\_data, how='left', on="city")

merged.head()



#finding max number of rides by city for validation

#max = merged["city"].value\_counts().max()

#print(max)

# creating 3 data frames for plottting purpose

urbandf = merged[merged["type"]=="Urban"]

surbandf = merged[merged["type"]=="Suburban"]

Ruraldf = merged[merged["type"]=="Rural"]

#average fare per city

UrbanAvfare = urbandf.groupby(['city']).mean()['fare']

SUbrbanAvfare = surbandf.groupby(['city']).mean()['fare']

RurrbanAvfare = Ruraldf.groupby(['city']).mean()['fare']

#print(SUbrbanAvfare) # Total Drivers per city

totdriversU = urbandf.groupby("city")["driver\_count"].mean()

totdriversS = surbandf.groupby("city")["driver\_count"].mean()

totdriversR = Ruraldf.groupby("city")["driver\_count"].mean()

#print(totdriversR)

# Total Drivers per city

totdriversU = urbandf.groupby("city")["driver\_count"].mean()

totdriversS = surbandf.groupby("city")["driver\_count"].mean()

totdriversR = Ruraldf.groupby("city")["driver\_count"].mean()

#print(totdriversR)



# Total Drivers per city

totdriversU = urbandf.groupby("city")["driver\_count"].mean()

totdriversS = surbandf.groupby("city")["driver\_count"].mean()

totdriversR = Ruraldf.groupby("city")["driver\_count"].mean()

#print(totdriversR)

# create a scatter plot amplifying the marker to show better

plt.scatter(totridesU, UrbanAvfare , s= totdriversU\*5 , c='#F08080', alpha=.7, edgecolor="black", marker="o", label="Urban")

plt.scatter(totridesS, SUbrbanAvfare , s= totdriversS\*5 , c='#87CEFA', alpha=.7, edgecolor="black", marker="o", label="Suburban")

plt.scatter(totridesR, RurrbanAvfare , s= totdriversR\*5 , c='#FFD700', alpha=.7, edgecolor="black", marker="o", label="Rural")

plt.title("Pyber Ride sharing Data")

plt.xlabel("Total number of rides (per city)")

plt.ylabel("Average Fare($)")

plt.text(10, 45,"Size of Bubble denotes ", fontsize=10)

plt.text(10, 43," # drivers per city", fontsize=10)

plt.legend()

sns.set

plt.show()

#df["Clo name"]= Array name

# % of total fares by City Type

Tot\_fare\_pie = merged.groupby(["type"])["fare"].sum()

print(Tot\_fare\_pie)

Tot\_fare\_pie.index

# creating Pie

explode = (0,0, 0.1)

colors = ["Gold", "lightcoral", "lightskyblue"]

#sizes = [Tot\_fare\_pie]

# Tells matplotlib that we want a pie chart with equal axes

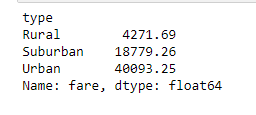
plt.axis("equal")

plt.legend(loc = "upper right")

plt.pie(Tot\_fare\_pie, labels = Tot\_fare\_pie.index, autopct = "%1.2f%%", colors = colors, explode = explode, shadow = True, startangle = 270)

# Prints our pie chart to the screen

plt.show()



#Total rides by city type.

Totrides\_pie = merged.groupby(["type"])["ride\_id"].count()

Totrides\_pie

#Use the City Pie index for wedge labels.

Totrides\_pie.index

#Set the colors of each section of the pie chart.

colors = ["gold", "lightskyblue", "lightcoral"]

#Only explode the first section of the pie chart.

explode = (0.5,0,0)

#Create the pie chart.

plt.pie(Totrides\_pie, labels = Totrides\_pie.index, autopct = "%1.2f%%", colors = colors, explode = explode, shadow = True)

#Tell Matplotlib that we wnat the pie chart to have equal axes.

plt.axis("equal")

#Add the legend.

plt.legend(loc = "upper right")

#Show the chart.

plt.show()

#Total drivers by city type.

Totdrivers\_pie = merged.groupby(["type"])["driver\_count"].mean()

Totdrivers\_pie

#Use the City Pie index for wedge labels.

Totdrivers\_pie.index

#Set the colors of each section of the pie chart.

colors = ["gold", "lightskyblue", "lightcoral"]

#Only explode the first section of the pie chart.

explode = (0,0.5,0)

#Create the pie chart.

plt.pie(Totdrivers\_pie, labels = Totdrivers\_pie.index, autopct = "%1.2f%%", colors = colors, explode = explode, shadow = True)

#Tell Matplotlib that we wnat the pie chart to have equal axes.

plt.axis("equal")

#Add the legend.

plt.legend(loc = "upper right")

#Show the chart.

plt.show()