# *Insight for Cab Investment firm*

## Business problem:

XYZ is a private firm in US. Due to remarkable growth in the Cab Industry in last few years and multiple key players in the market, it is planning for an investment in Cab industry and as per their Go-to-Market(G2M) strategy they want to understand the market before taking final decision.

## Properties of the data provided (data intake report):

After merging 4 csv files, the final dataset contains 3,59,392 rows and 24 columns containing information of 2 cab services from 19 cities.

## Steps taken in order to create an applicable data set:

1. Merged Cab and City data on 'City' column.
2. Merged Customer and Transaction data on 'Transaction_ID'.
3. Finally merged the above two data on 'Transaction_ID'.

## Steps taken perform analysis:

1. Convert 'Date of Travel' column into pandas datetime column and set it as the index
2. Created new columns to better analyze the trend.
3. EDA
4. Hypothesis Testing

## Type of analysis performed:

1. Univariate Analysis
2. Bivariate Analysis
3. Time series Analysis

## Assumptions made:

1. Outliers are present in "Price Charged" feature but due to unavailability of trip duration details, we are not treating this as outlier.

2. Profit of rides are calculated keeping other factors constant and only "Price Charged" and "Cost of Trip" features used to calculate profit.
3. Users feature of city dataset is treated as number of cab users in the city.

# Data Collection

## Import Libraries & set default style

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib

sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (15, 9)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

## Read csv files

```
cab = pd.read_csv("Cab_Data.csv")
city = pd.read_csv("City.csv")
customer = pd.read_csv("Customer_ID.csv")
transaction = pd.read_csv("Transaction_ID.csv")
```

## Merge into one dataframe

```
df_1 = pd.merge(cab, city, on="City")
df_2 = pd.merge(customer, transaction, on="Customer ID")
df = pd.merge(df_1, df_2, on="Transaction ID")
```

## Create new columns

```
df['Date of Travel'] = pd.to_datetime(df['Date of Travel'])
df['Day'] = df['Date of Travel'].dt.day
df['Weekday'] = df['Date of Travel'].dt.weekday
df['Month'] = df['Date of Travel'].dt.month
df['Year'] = df['Date of Travel'].dt.year
df['Profit'] = df['Price Charged'] - df['Cost of Trip']
```

```
df['Profit Percentage per Trip'] = ((df['Profit'] / df['Cost of Trip'])*100).round(2)
df['Profit per KM'] = ((df['Profit'] / df['KM Travelled']))

df['Population'] = df['Population'].str.replace(',', '').astype(float)
df['Users'] = df['Users'].str.replace(',', '').astype(float)
df['Users Density'] = df['Users'] / df['Population']
```

```
df.sort_values(by='Date of Travel', inplace=True)
df.set_index('Date of Travel', inplace=True)
```

# Data Exploration

```
pd.set_option("display.max_columns", 25)
df
```

```
df.shape
```

```
(359392, 21)
```

of

Final dataset contains 3,59,392 rows & 21 columns

2016- 10004899 Yellow LOS 25.53 402.89 327.8052 1595037.0

## Get some statistical values of each Numerical colums

```
df.describe()
```

| | Transaction ID | KM Travelled | Price Charged | Cost of Trip | Population | |
|---|---|---|---|---|---|---|
| **count** | 3.593920e+05 | 359392.000000 | 359392.000000 | 359392.000000 | 3.593920e+05 | 359392 |
| **mean** | 1.022076e+07 | 22.567254 | 423.443311 | 286.190113 | 3.132198e+06 | 158365 |
| **std** | 1.268058e+05 | 12.233526 | 274.378911 | 157.993661 | 3.315194e+06 | 100850 |
| **min** | 1.000001e+07 | 1.900000 | 15.600000 | 19.000000 | 2.489680e+05 | 3643 |
| **25%** | 1.011081e+07 | 12.000000 | 206.437500 | 151.200000 | 6.712380e+05 | 80021 |
| **50%** | 1.022104e+07 | 22.440000 | 386.360000 | 282.480000 | 1.595037e+06 | 144132 |
| **75%** | 1.033094e+07 | 32.960000 | 583.660000 | 413.683200 | 8.405837e+06 | 302149 |
| **max** | 1.044011e+07 | 48.000000 | 2048.030000 | 691.200000 | 8.405837e+06 | 302149 |

12-31 10438259 Cab DALLAS TX 34.00 635.45 428.4000 942908.0

Since there is no null value and also we can see that the minimum and maximum km travelled, price and cost are all valid values so no need to drop any rows from the dataset

## Get type, null-value count

```
df.info();
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 359392 entries, 2016-01-02 to 2018-12-31
Data columns (total 21 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   Transaction ID                359392 non-null  int64
 1   Company                       359392 non-null  object
 2   City                          359392 non-null  object
 3   KM Travelled                  359392 non-null  float64
 4   Price Charged                 359392 non-null  float64
```

```
 5   Cost of Trip                359392 non-null  float64
 6   Population                  359392 non-null  float64
 7   Users                       359392 non-null  float64
 8   Customer ID                 359392 non-null  int64
 9   Gender                      359392 non-null  object
 10  Age                         359392 non-null  int64
 11  Income (USD/Month)          359392 non-null  int64
 12  Payment_Mode                359392 non-null  object
 13  Day                         359392 non-null  int64
 14  Weekday                     359392 non-null  int64
 15  Month                       359392 non-null  int64
 16  Year                        359392 non-null  int64
 17  Profit                      359392 non-null  float64
 18  Profit Percentage per Trip  359392 non-null  float64
 19  Profit per KM               359392 non-null  float64
 20  Users Density               359392 non-null  float64
dtypes: float64(9), int64(8), object(4)
memory usage: 60.3+ MB
```

There are no missing values

## Check Duplicate Rows if any

```
duplicate = df[df.duplicated()]
duplicate
```

| | Transaction ID | Company | City | KM Travelled | Price Charged | Cost of Trip | Population | Users | Cust |
|---|---|---|---|---|---|---|---|---|---|
| **Date of Travel** | | | | | | | | | |

There are no duplicate rows!

## Find unique values of each column

```
df.nunique()
```

```
Transaction ID              359392
Company                          2
City                            19
KM Travelled                   874
Price Charged                99176
Cost of Trip                 16291
Population                      19
Users                           19
Customer ID                  46148
Gender                           2
```

```
Age                             48
Income (USD/Month)           22725
Payment_Mode                     2
Day                             31
Weekday                          7
Month                           12
Year                             3
Profit                      303907
Profit Percentage per Trip   21939
Profit per KM               356133
Users Density                   19
dtype: int64
```

There are 2 cab service provider in 19 different cities

## City with highest no. of running cabs

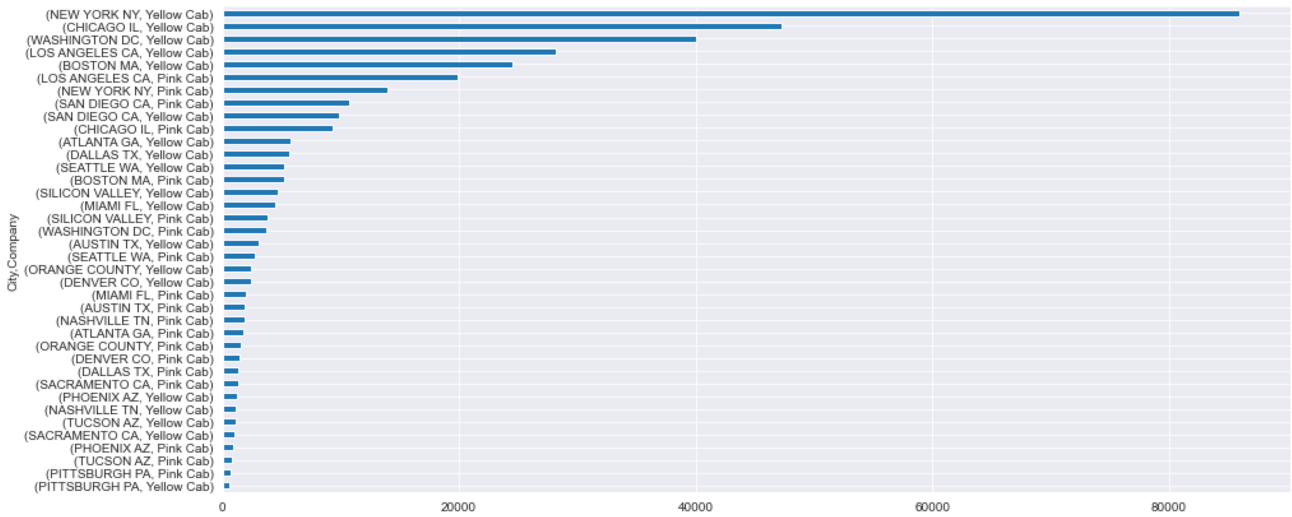
```
df['City'].value_counts()
```

```
NEW YORK NY       99885
CHICAGO IL        56625
LOS ANGELES CA    48033
WASHINGTON DC     43737
BOSTON MA         29692
SAN DIEGO CA      20488
SILICON VALLEY     8519
SEATTLE WA         7997
ATLANTA GA         7557
DALLAS TX          7017
MIAMI FL           6454
AUSTIN TX          4896
ORANGE COUNTY      3982
DENVER CO          3825
NASHVILLE TN       3010
SACRAMENTO CA      2367
PHOENIX AZ         2064
TUCSON AZ          1931
PITTSBURGH PA      1313
Name: City, dtype: int64
```

New York City count in the dataset is the highest which may imply more no. of cabs are running in this city. This may be due to high population also.

## Demand of the 2 cab service providers in each city

```
plt.figure(figsize=(15, 7))
df.groupby('City').Company.value_counts().sort_values(ascending=True).plot(kind='barh');
```




## Yellow Cabs are dominating in most of the cities

```
city_grp = df.groupby('City')
city_grp['Company'].value_counts().unstack()
```

| Company | Pink Cab | Yellow Cab |
|---|---|---|
| **City** | | |
| **ATLANTA GA** | 1762 | 5795 |
| **AUSTIN TX** | 1868 | 3028 |

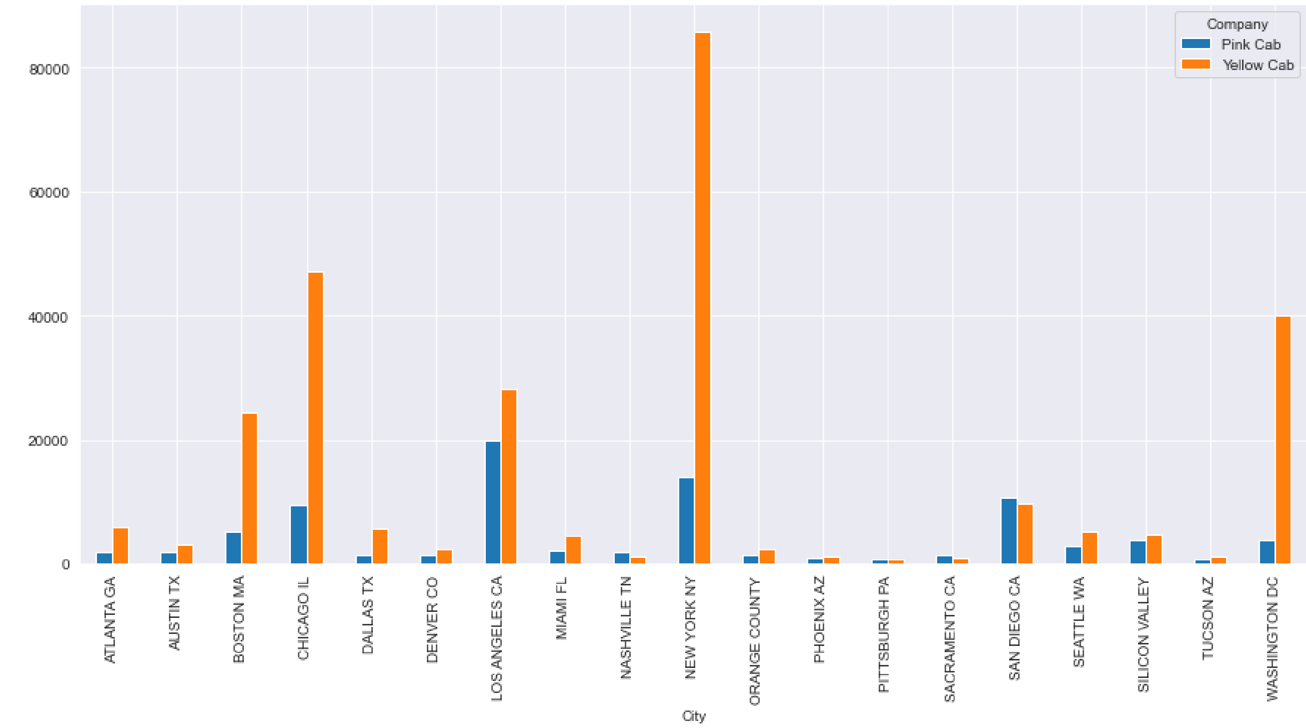
People prefer Yellow Cabs over Pink Cabs in every city except these 4:

1. Nashville
2. Pittsburgh
3. Sacramento
4. San Diego

| | | |
|---|---|---|
| **NASHVILLE TN** | 1841 | 1169 |

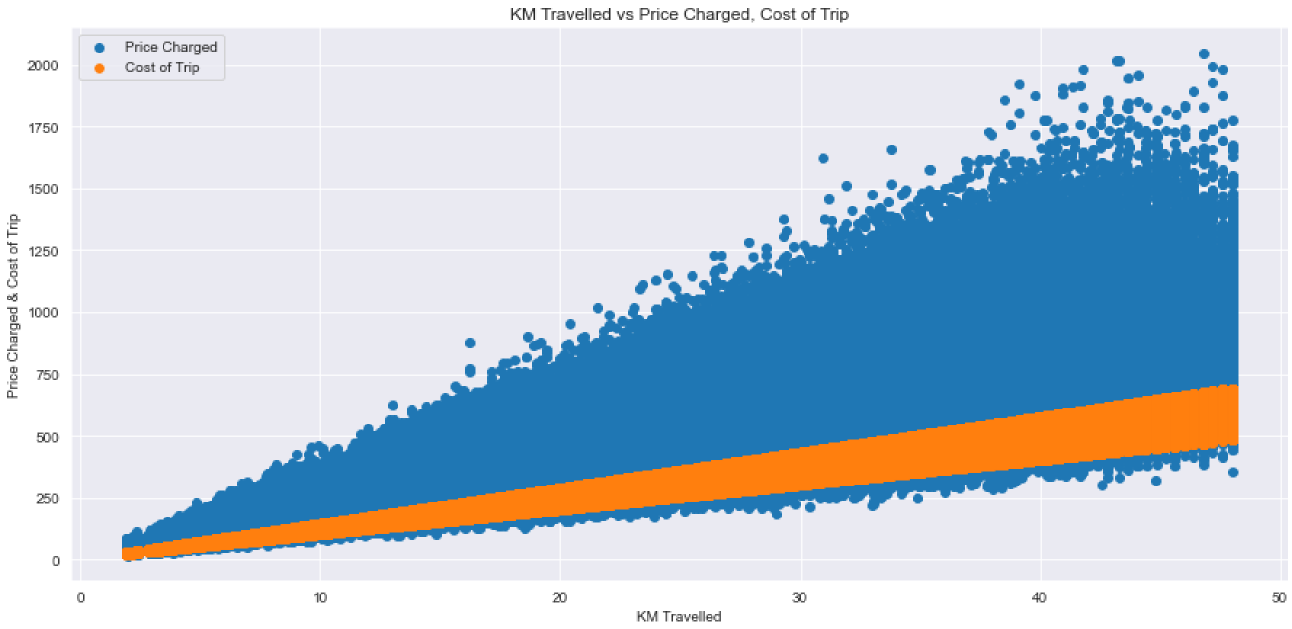
## Visual Comparison:

```
city_grp['Company'].value_counts().unstack().plot(kind='bar', figsize=(15, 7));
```




```
fig, ax = plt.subplots(figsize=(15,7))
ax.scatter(x = df['KM Travelled'], y = df['Price Charged']);
ax.scatter(x = df['KM Travelled'], y = df['Cost of Trip']);
```

```
plt.xlabel("KM Travelled")
plt.ylabel("Price Charged & Cost of Trip")
plt.title("KM Travelled vs Price Charged, Cost of Trip")
ax.legend(['Price Charged', 'Cost of Trip'])
plt.show()
```




As the distance increases, both cost and price increases linearly but the difference becomes more pronounced

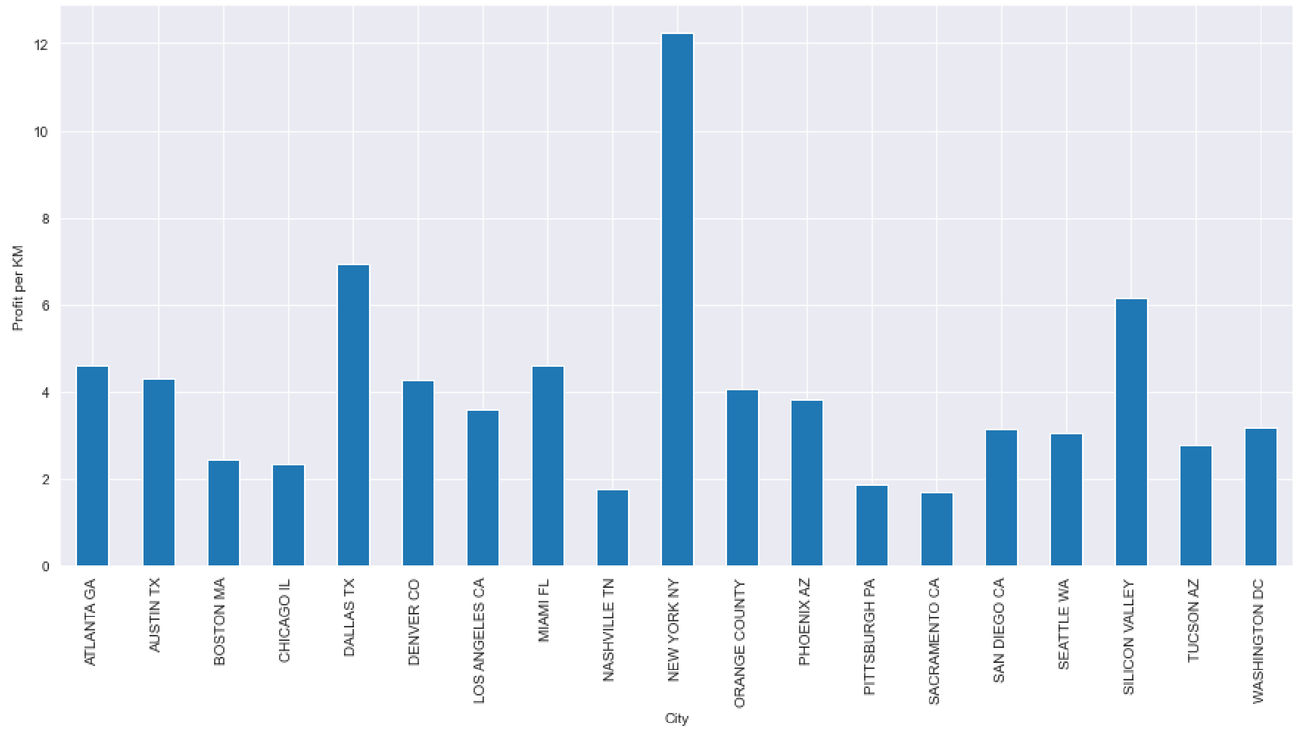
## Profit per KM City wise

```
df.groupby('City')['Profit per KM'].median()
```

```
City
ATLANTA GA          4.591498
AUSTIN TX           4.296468
BOSTON MA           2.448953
CHICAGO IL          2.329217
DALLAS TX           6.936991
DENVER CO           4.262593
LOS ANGELES CA      3.570160
MIAMI FL            4.599710
NASHVILLE TN        1.769476
NEW YORK NY        12.268408
ORANGE COUNTY       4.051526
PHOENIX AZ          3.815931
PITTSBURGH PA       1.863913
SACRAMENTO CA       1.696495
```

```
SAN DIEGO CA        3.131335
SEATTLE WA          3.052507
SILICON VALLEY      6.169811
TUCSON AZ           2.770540
WASHINGTON DC       3.171498
Name: Profit per KM, dtype: float64
```

```
df.groupby('City')['Profit per KM'].median().plot(kind='bar', figsize=(15,7), ylabel='Prof
```




New York City has the highest Profit per KM while Sacramenyo has the lowest Profit per KM

## Overall profit analysis over 3 years

```
month_year_group = df.groupby(['Month', 'Year'])
(month_year_group[['KM Travelled', 'Profit', 'Profit Percentage per Trip', 'Profit per KM'
```

| | KM Travelled | | | Profit | | | Profit Percentage per Trip | | | Profit |
|---|---|---|---|---|---|---|---|---|---|---|
| Year | 2016 | 2017 | 2018 | 2016 | 2017 | 2018 | 2016 | 2017 | 2018 | 2016 |
| Month | | | | | | | | | | |
| 1 | 22.68 | 22.310 | 22.47 | 101.3955 | 102.5436 | 78.1024 | 42.165 | 43.920 | 32.780 | 5.28482 |
| 2 | 22.23 | 22.800 | 22.42 | 105.1704 | 98.8360 | 83.3908 | 44.110 | 41.760 | 35.100 | 5.50875 |
| 3 | 22.66 | 22.310 | 22.40 | 99.0996 | 100.4620 | 80.1816 | 42.190 | 43.920 | 33.610 | 5.26304 |
| 4 | 22.47 | 22.200 | 22.77 | 94.0260 | 84.4970 | 75.0600 | 40.620 | 39.725 | 31.440 | 5.03833 |
| 5 | 22.14 | 22.000 | 22.44 | 97.5768 | 106.3420 | 83.5194 | 41.825 | 46.110 | 36.135 | 5.28040 |
| 6 | 22.54 | 22.680 | 22.04 | 101.4520 | 92.0100 | 71.1640 | 42.320 | 39.860 | 30.220 | 5.32666 |
| 7 | 22.88 | 22.420 | 22.47 | 75.8940 | 75.5400 | 56.7342 | 32.635 | 33.380 | 24.590 | 4.03581 |
| 8 | 22.44 | 22.420 | 22.04 | 66.4040 | 79.0450 | 56.9450 | 29.500 | 34.605 | 25.060 | 3.62477 |

On comparison, we see that there is slight decrement in the profit margin for the year 2018.

[ ] ↳ *1 cell hidden*

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 22.60 | 22.610 | 22.54 | 91.5096 | 95.3624 | 73.1400 | 39.300 | 43.640 | 31.170 | 4.85063 |

There is a dip in profit per KM each year during July and August which implies there is some seasonality.

[ ] ↳ *1 cell hidden*

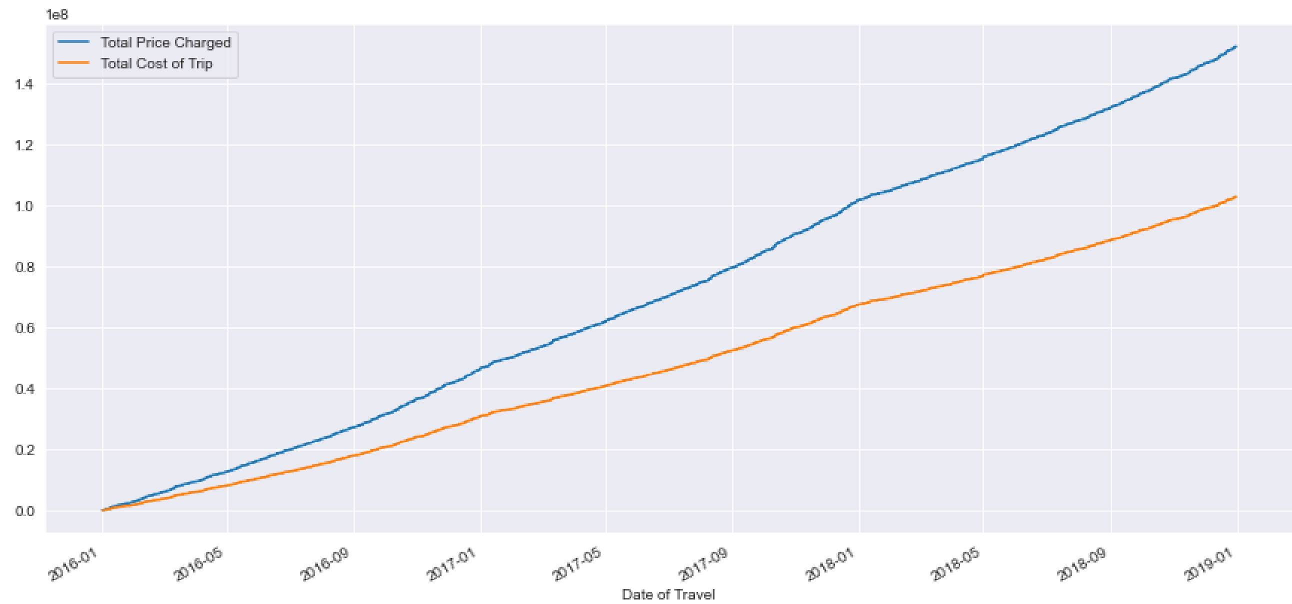
Avg distance travelled is 22.5 KM. Later we will prove it using null hypothesis.

## Weekly Analysis:

[ ] ↳ *2 cells hidden*

# Cummulative Profit Vs Cost over the years b/w 2 cab service providers

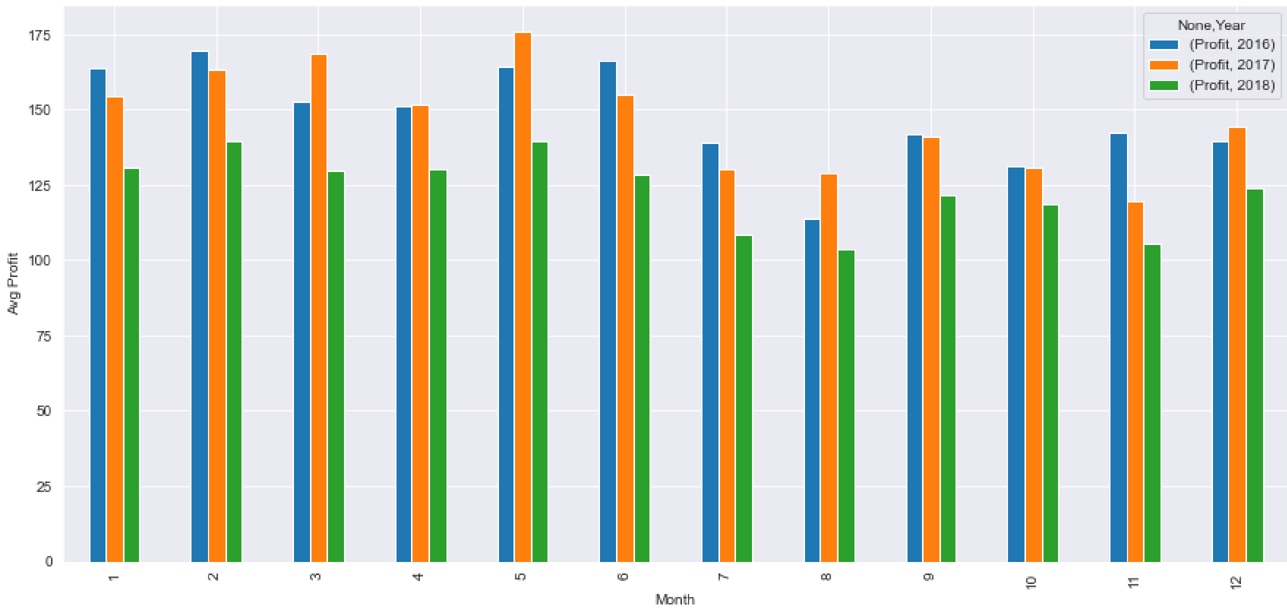
```
df['Total Price Charged'] = df['Price Charged'].cumsum()
df['Total Cost of Trip'] = df['Cost of Trip'].cumsum()

plt.figure(figsize=(15, 7))
df['Total Price Charged'].plot();
df['Total Cost of Trip'].plot();
plt.legend();
```




- Above graph shows the power of compounding effect.

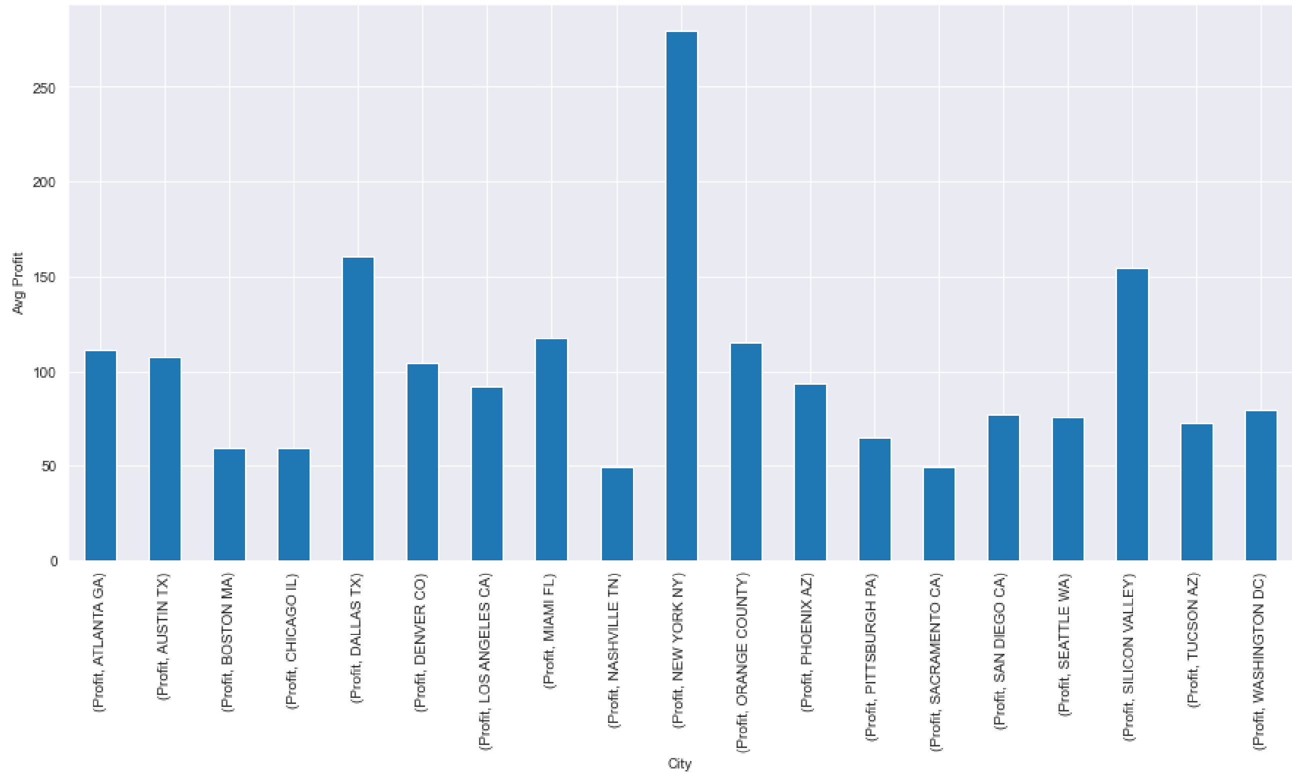
  To maximize the profit, XYZ should invest for a long term

```
((month_year_group[['Profit']].mean()).unstack()).plot(kind='bar', figsize=(15, 7), ylabel
```




```
((city_grp[['Profit']].mean()).unstack()).plot(kind='bar', figsize=(15, 7), ylabel='City'
```

```
((city_grp[['Profit']].mean()).unstack()).plot(kind='bar', figsize=(15, 7), xlabel='City',
```

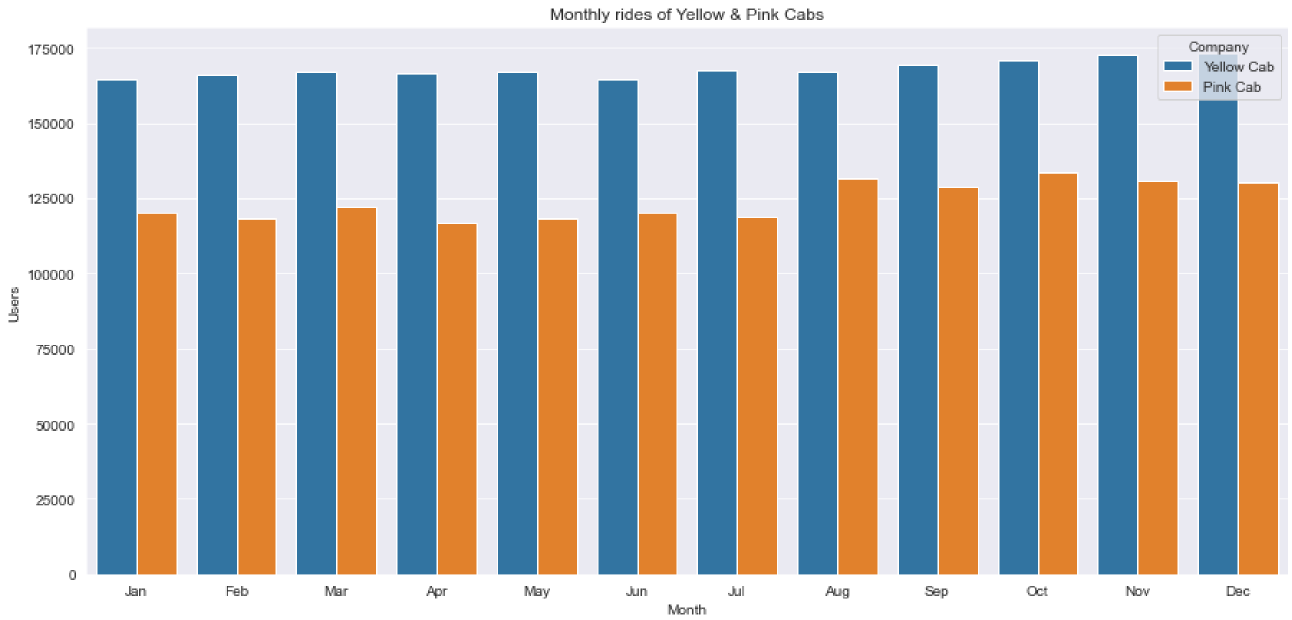



## Top 5 cities with highest avg profit (in descending order):

1. New York
2. Dallas
3. Silicon Valley
4. Miami
5. Orange County

```
month = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec
```

```
plt.figure(figsize=(15,7))
g = sns.barplot('Month', 'Users', data=df, hue='Company', ci=None);
g.set_xticklabels(labels=month, rotation=0)
g.set_title('Monthly rides of Yellow & Pink Cabs')
plt.show()
```

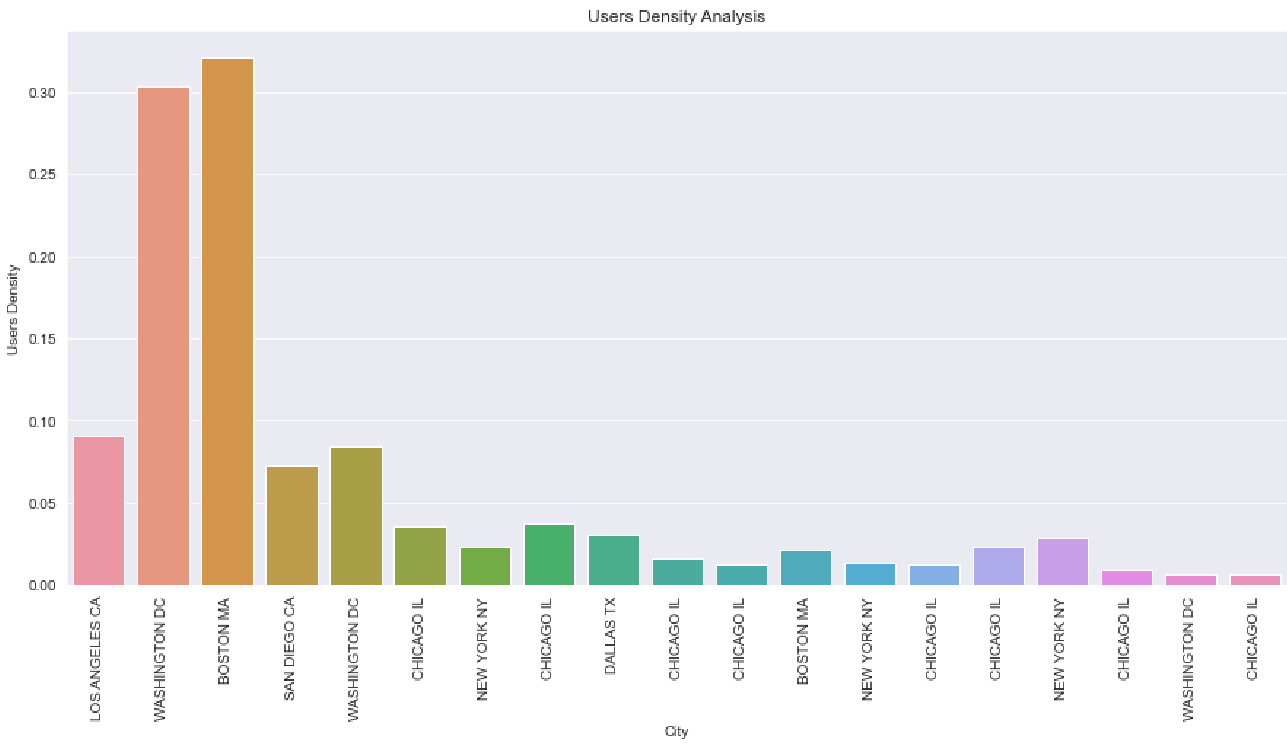



## Yellow cab has more users each month over the years

```
plt.figure(figsize=(15,7))
g = sns.barplot('City', 'Profit', data=df, hue='Company', ci=None, dodge=0);
g.set_xticklabels(labels=df['City'], rotation=90)
g.set_title('Overall Profit of Yellow, Pink Cabs in each City')
plt.show()
```

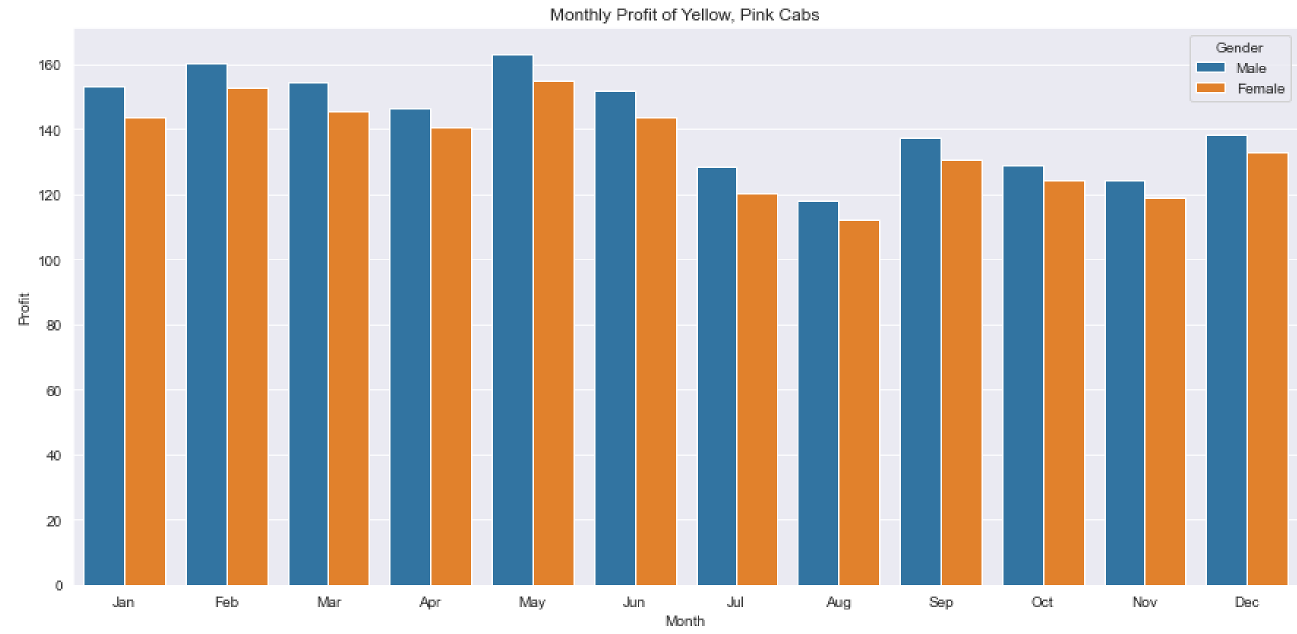

- Except Chicago, Yellow Cab has more profit margin in each city.

```
plt.figure(figsize=(15,7))
g = sns.barplot('City', 'Users Density', data=df, ci=None, dodge=1);
g.set_xticklabels(labels=df['City'], rotation=90)
g.set_title('Users Density Analysis')
plt.show()
```

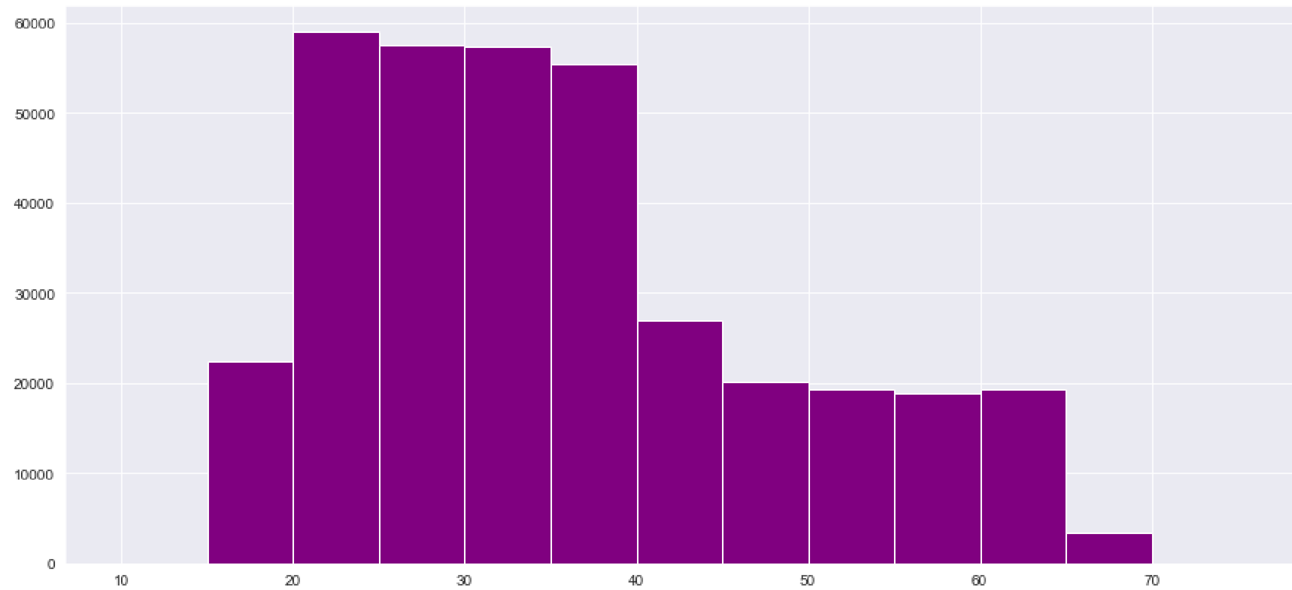



- Around 30% of the population in Washington DC and Boston use cab services whereas for all other cities it's less than 10%

```
plt.figure(figsize=(15,7))
g = sns.barplot('Month', 'Profit', data=df, hue='Gender', ci=None);
g.set_xticklabels(labels=month, rotation=0)
g.set_title('Monthly Profit of Yellow, Pink Cabs')
plt.show()
```

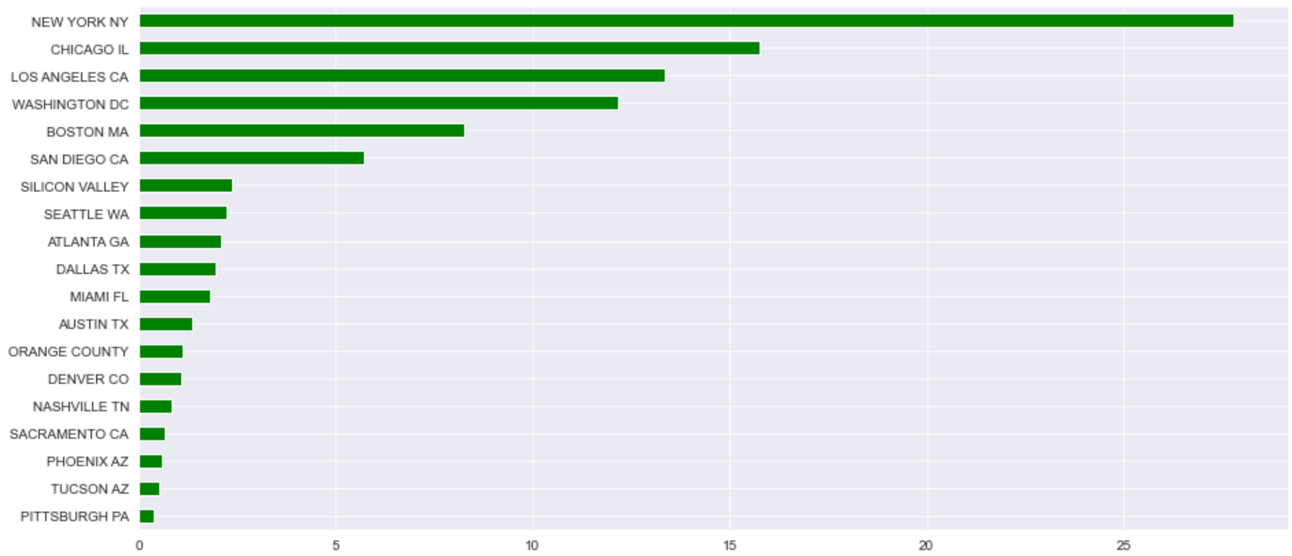



```
plt.figure(figsize=(15,7))
plt.hist(df.Age, bins=np.arange(10, 80, 5), color='purple');
plt.show()
```

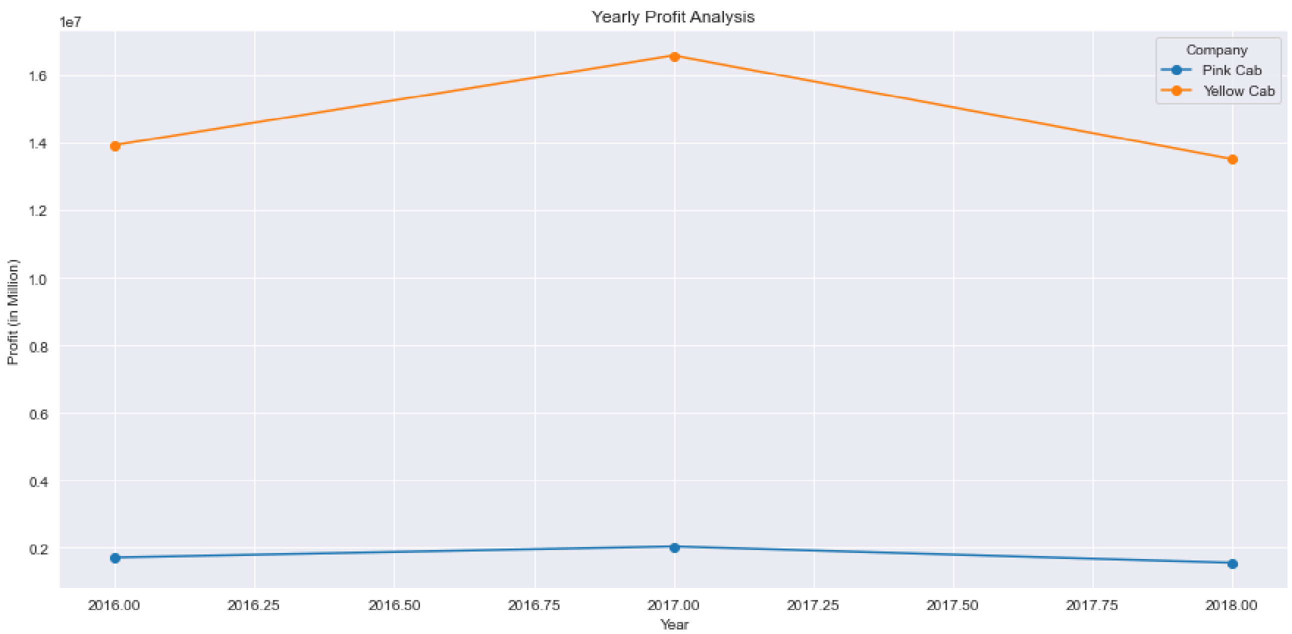



## Most of the users are aged between 20 to 40 years

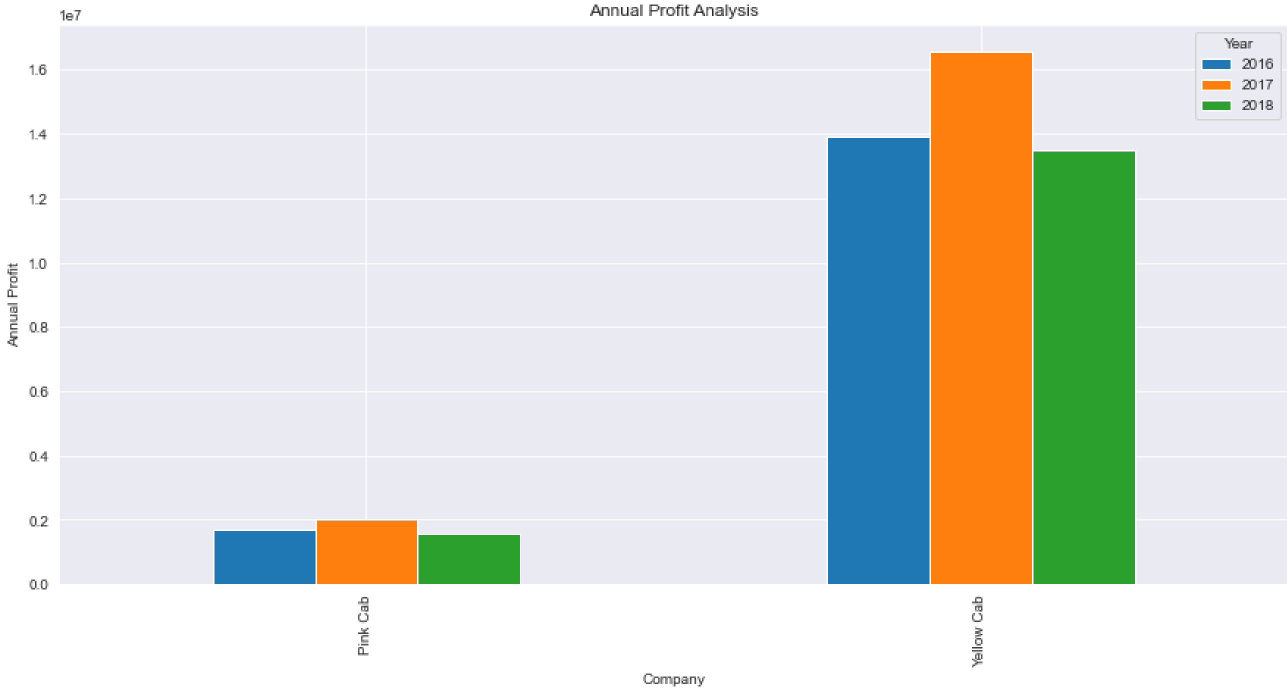
```
plt.figure(figsize=(15,7))
(df.City.value_counts(normalize=True, ascending=True)*100).plot(kind='barh', color='g');
plt.show()
```




```
fig, ax = plt.subplots(figsize=(15, 7))
df.groupby(['Year', 'Company']).sum()['Profit'].unstack().plot(ax=ax, title='Yearly Profit
```




```
(df.pivot_table(index='Company', columns='Year', values='Profit', aggfunc='sum')).plot(kin
```




```
df.pivot_table(index='Company', columns='Year', values='Profit', aggfunc='sum', margins=Tr
```

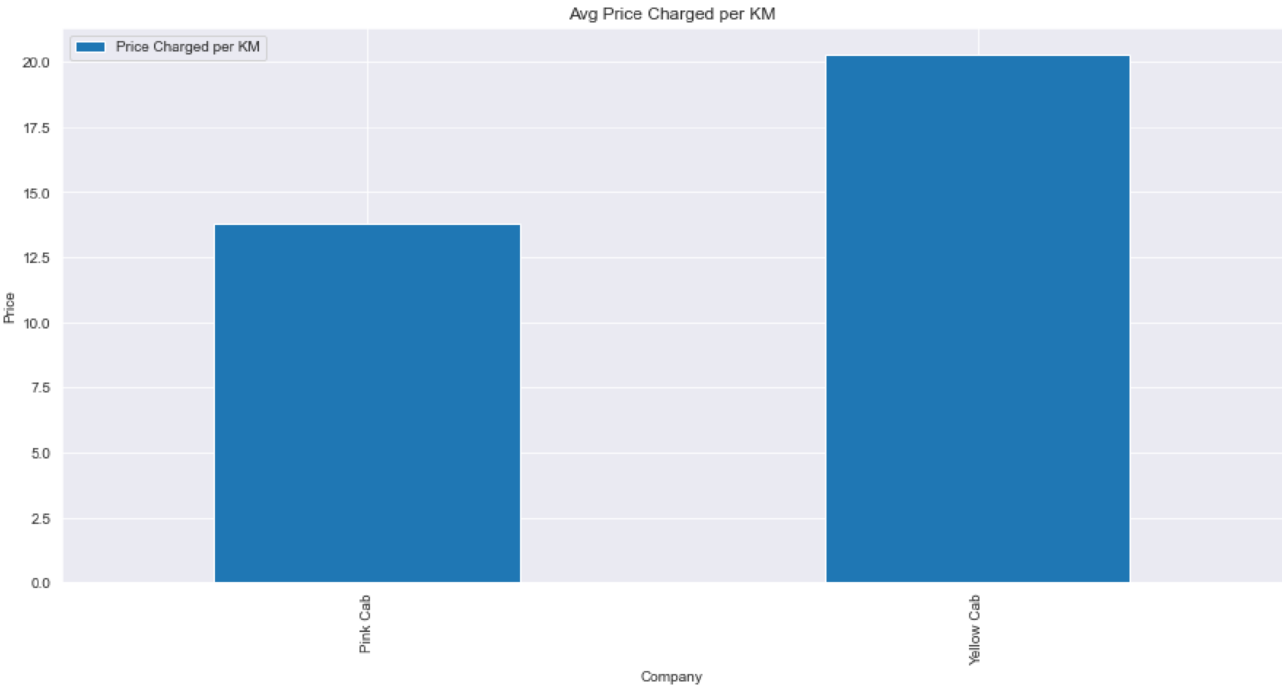
| Year<br>Company | 2016 | 2017 | 2018 | All |
|---|---|---|---|---|
| **Pink Cab** | 1.713511e+06 | 2.033655e+06 | 1.560162e+06 | 5.307328e+06 |
| **Yellow Cab** | 1.392700e+07 | 1.657598e+07 | 1.351740e+07 | 4.402037e+07 |
| **All** | 1.564051e+07 | 1.860963e+07 | 1.507756e+07 | 4.932770e+07 |

```
fig, ax = plt.subplots(figsize=(15, 7))
df.groupby(['Month', 'Company']).sum()['Profit'].unstack().plot(ax=ax, title='Monthly Prof
```


- There is a decrease in Monthly Profit of Yellow Cab during June to August whereas Pink Cab has an increase

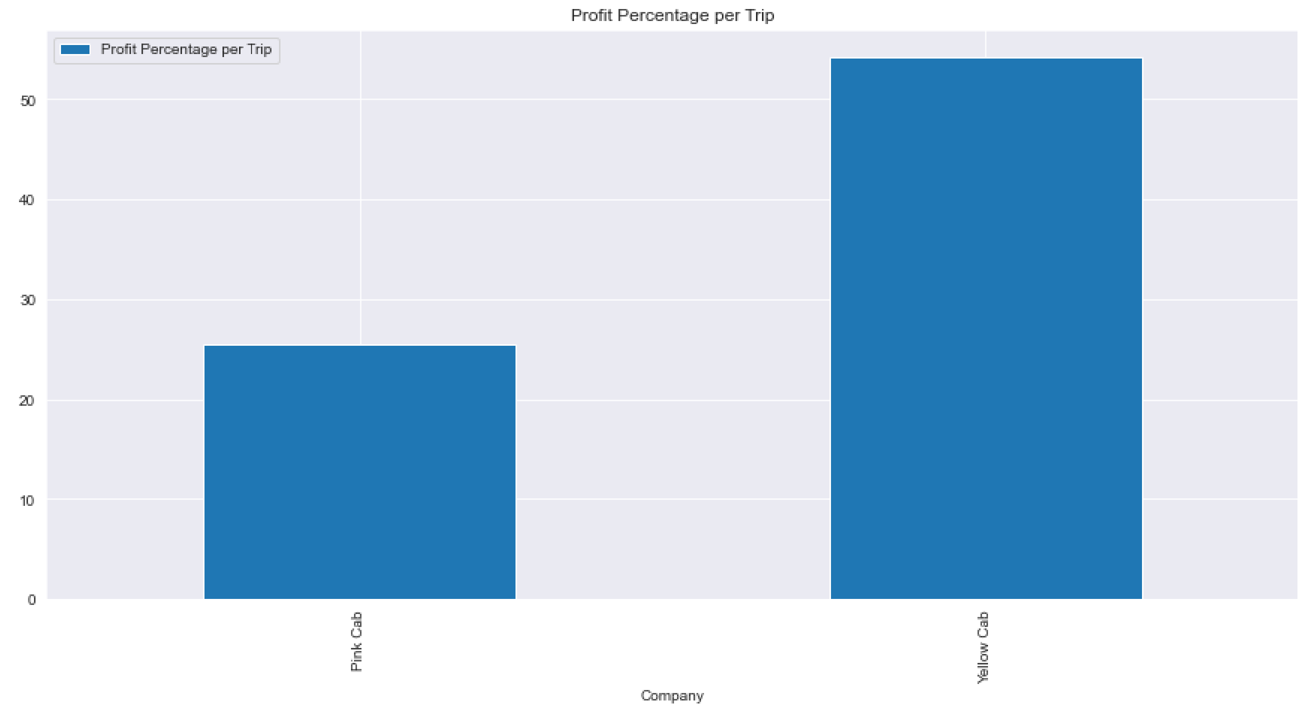
```
df['Price Charged per KM'] = df['Price Charged'] / df['KM Travelled']
```

```
((df[['Price Charged per KM', 'Company']].groupby('Company')).mean()).plot(kind='bar', fig
plt.show()
```




- Avg Price Charged per KM for Yellow Cab is 20.3 USD & for Pink Cab is 13.76 USD

```
df[['Profit Percentage per Trip', 'Company']].groupby('Company').mean().plot(kind='bar', f
plt.show()
```

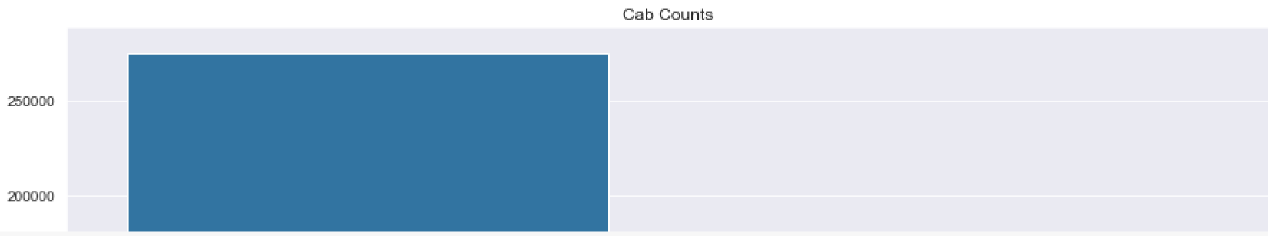



Profit Percentage per Trip for

Pink Cab is 25.559567

Yellow Cab is 54.296631

```
plt.figure(figsize=(15,7))
g=sns.countplot(x='Company', data=df);
g.set_title('Cab Counts')
plt.show()
```



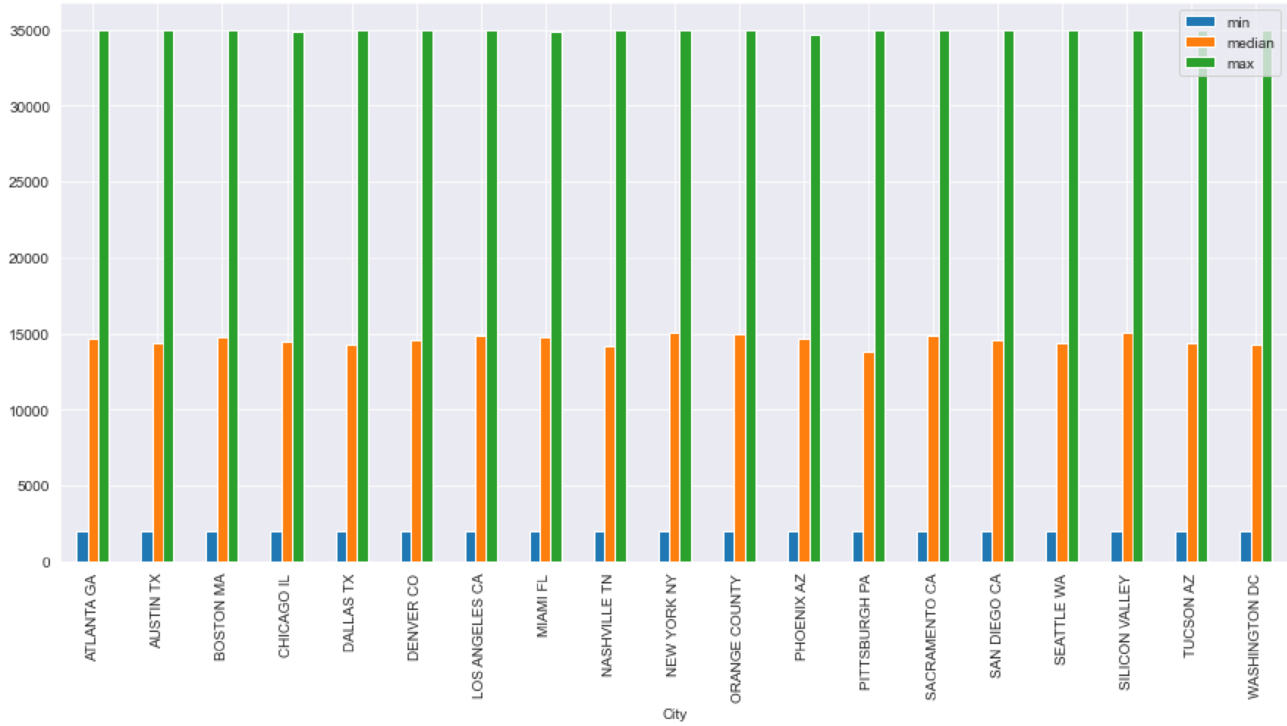

```
df['Company'].value_counts(normalize=True)
```

```
Yellow Cab    0.764294
Pink Cab      0.235706
Name: Company, dtype: float64
```

```
city_grp['Income (USD/Month)'].agg(['median', 'mean', 'min', 'max'])
```

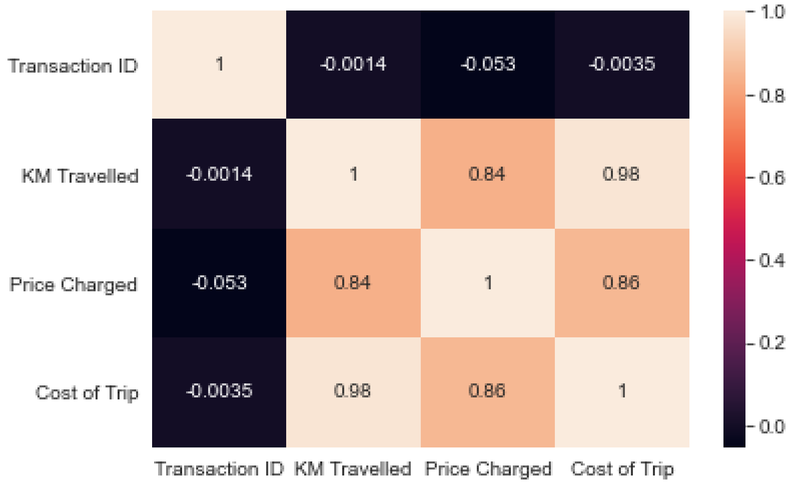
| City | median | mean | min | max |
|---|---|---|---|---|
| **ATLANTA GA** | 14655 | 14933.150986 | 2029 | 34972 |
| **AUSTIN TX** | 14374 | 14696.495711 | 2027 | 34921 |
| **BOSTON MA** | 14743 | 15128.563317 | 2019 | 34985 |
| **CHICAGO IL** | 14527 | 15101.718269 | 2007 | 34901 |
| **DALLAS TX** | 14242 | 14846.508194 | 2007 | 34996 |
| **DENVER CO** | 14580 | 14975.655163 | 2022 | 35000 |
| **LOS ANGELES CA** | 14889 | 15064.550455 | 2007 | 34984 |
| **MIAMI FL** | 14759 | 14984.887202 | 2013 | 34862 |
| **NASHVILLE TN** | 14195 | 14734.359801 | 2002 | 34960 |
| **NEW YORK NY** | 15024 | 15184.765801 | 2012 | 34989 |
| **ORANGE COUNTY** | 14963 | 15188.944500 | 2030 | 34979 |
| **PHOENIX AZ** | 14646 | 15012.038275 | 2011 | 34681 |
| **PITTSBURGH PA** | 13833 | 14410.332064 | 2010 | 34984 |
| **SACRAMENTO CA** | 14829 | 15268.225180 | 2001 | 34995 |
| **SAN DIEGO CA** | 14612 | 15049.874854 | 2016 | 34936 |
| **SEATTLE WA** | 14358 | 14840.748281 | 2000 | 34967 |
| **SILICON VALLEY** | 15107 | 15248.547717 | 2000 | 34977 |
| **TUCSON AZ** | 14422 | 14942.952356 | 2012 | 34928 |
| **WASHINGTON DC** | 14268 | 14727.430162 | 2003 | 34996 |

```
(city_grp['Income (USD/Month)'].agg(['min', 'median', 'max'])).plot(kind='bar', figsize=(1
```

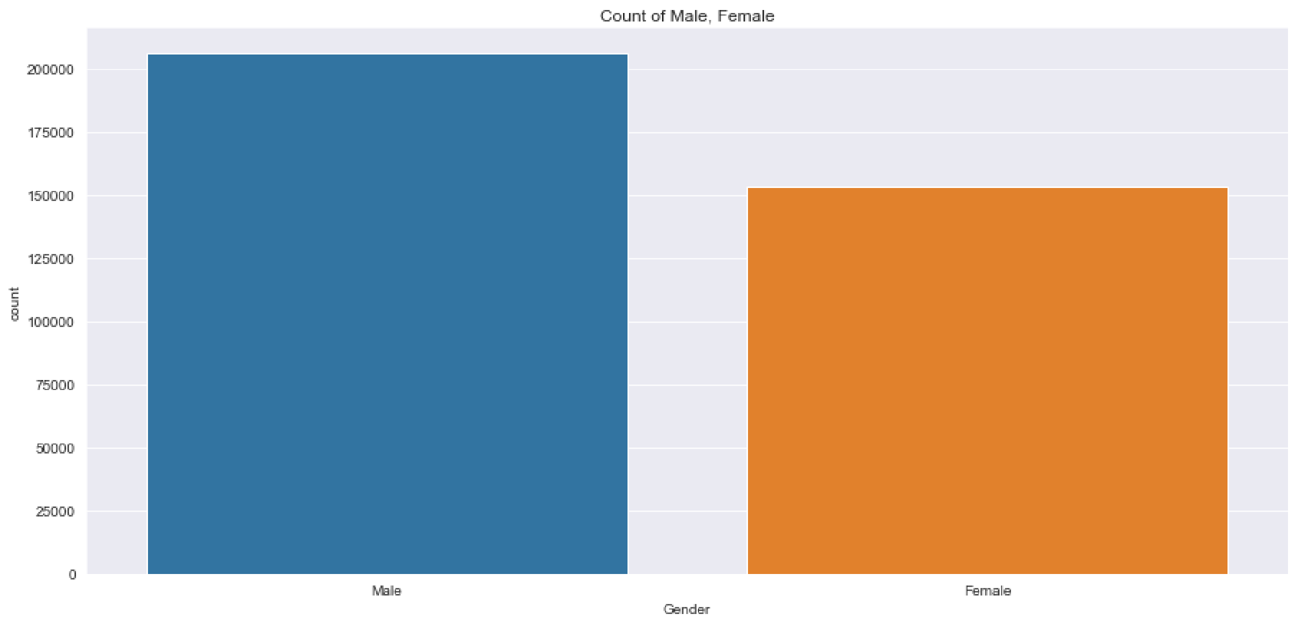



## There is equal range of incomes for each city

```
sns.heatmap(cab.corr(), color='b', annot=True);
```

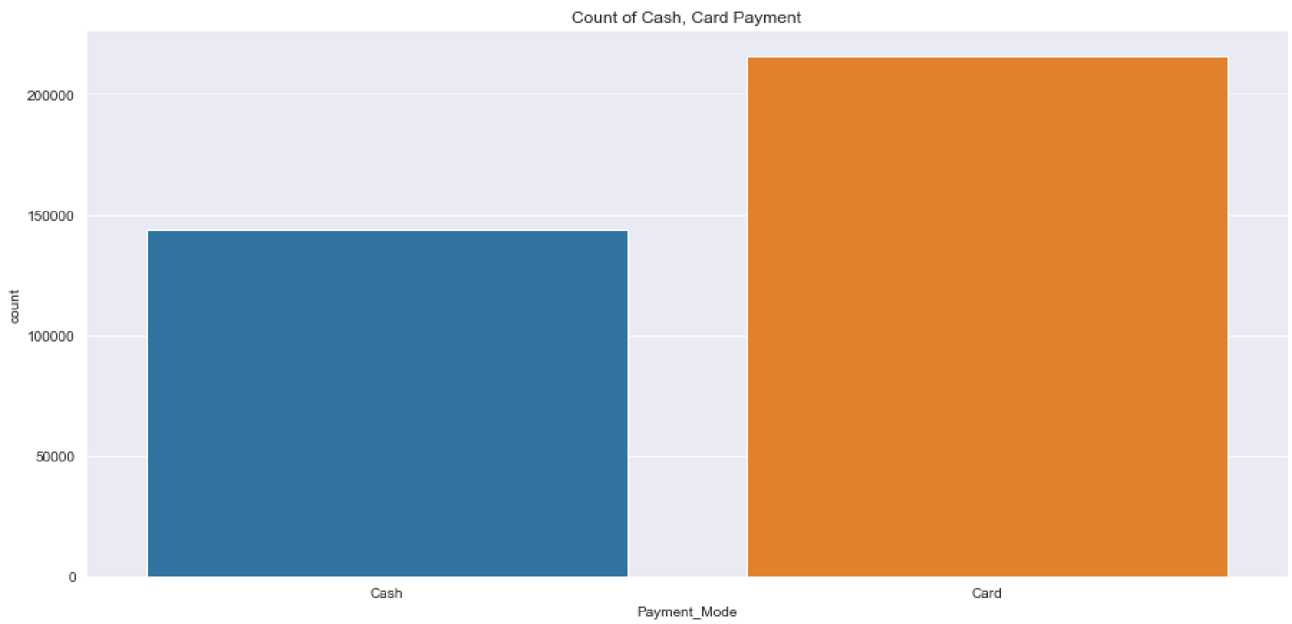



```
plt.figure(figsize=(15,7))
g=sns.countplot(x = 'Gender', data = df);
g.set_title('Count of Male, Female')
plt.show()
```




## More no. of Male users

```
plt.figure(figsize=(15,7))
g=sns.countplot(df['Payment_Mode']);
g.set_title('Count of Cash, Card Payment')
plt.show()
```

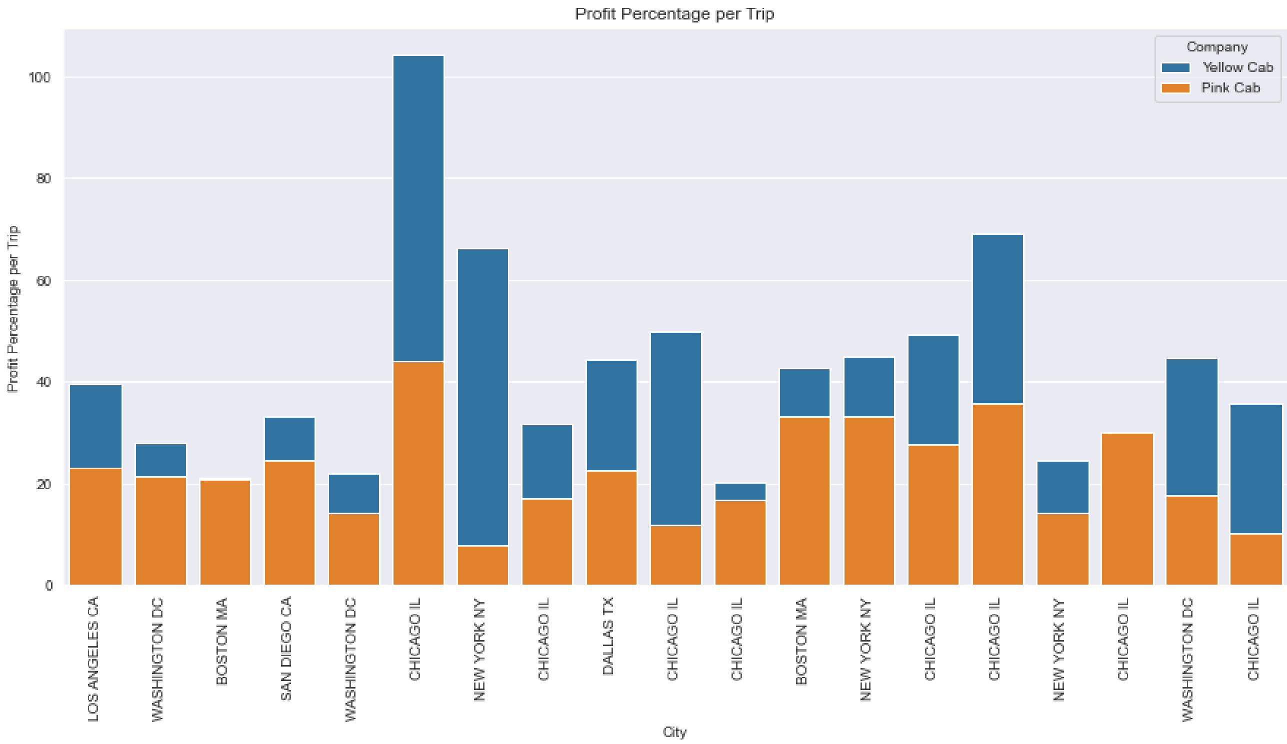


## No. of Card payment is more than no. of cash payment

```
df['Payment_Mode'].value_counts()
```

```
Card     215504
Cash     143888
Name: Payment_Mode, dtype: int64
```

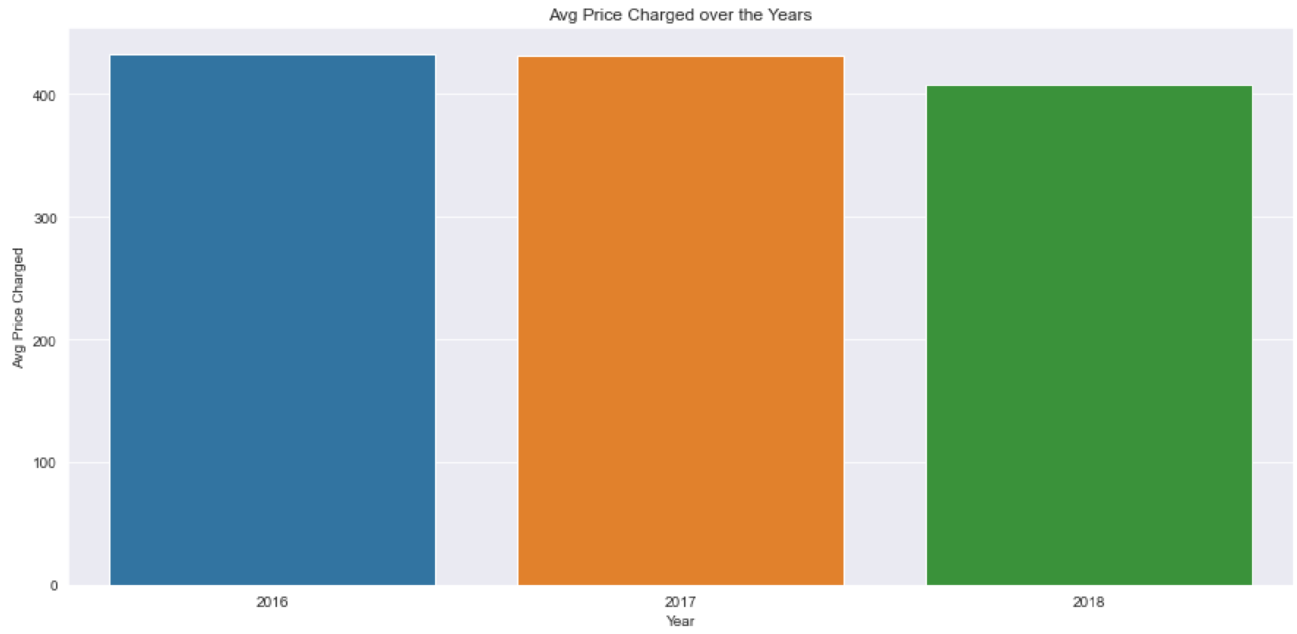
```
plt.figure(figsize=(15,7))
g = sns.barplot(x='City', y='Profit Percentage per Trip', hue='Company', data=df, dodge=0,
g.set_title('Profit Percentage per Trip')
g.set_xticklabels(labels=df['City'], rotation=90);
plt.show()
```




## Except Chicago and Boston Yellow Cab makes more profit per trip in every city

```
yearly_price = df.groupby(['Year'])['Price Charged'].mean().reset_index().rename(columns =
```

```
plt.figure(figsize=(15,7))
sns.barplot(x = 'Year', y = 'Avg Price Charged', data = yearly_price).set_title("Avg Price
plt.show()
```




- The avg price charged for the year 2018 is comparitively less.

```
yearly_cost = df.groupby(['Year'])['Cost of Trip'].mean().reset_index().rename(columns = {
```

```
plt.figure(figsize=(15,7))
sns.barplot(x = 'Year', y = 'Avg Cost of Trip', data = yearly_cost).set_title("Avg Cost of
plt.show()
```

The avg cost of trip over the years remains constant


# T Test

A t-test is a type of inferential statistic which is used to determine if there is a significant difference between the means of two groups which may be related in certain features

T-test has 2 types :

1. One sampled t-test
2. Two-sampled t-test.

```
import scipy.stats as stat
```

```
from scipy.stats import ttest_1samp
```

```
from scipy.stats import ttest_ind
```

```
df.describe()
```

| | Transaction ID | KM Travelled | Price Charged | Cost of Trip | Population | |
|---|---|---|---|---|---|---|
| **count** | 3.593920e+05 | 359392.000000 | 359392.000000 | 359392.000000 | 3.593920e+05 | 359392 |
| **mean** | 1.022076e+07 | 22.567254 | 423.443311 | 286.190113 | 3.132198e+06 | 158365 |
| **std** | 1.268058e+05 | 12.233526 | 274.378911 | 157.993661 | 3.315194e+06 | 100850 |
| **min** | 1.000001e+07 | 1.900000 | 15.600000 | 19.000000 | 2.489680e+05 | 3643 |
| **25%** | 1.011081e+07 | 12.000000 | 206.437500 | 151.200000 | 6.712380e+05 | 80021 |
| **50%** | 1.022104e+07 | 22.440000 | 386.360000 | 282.480000 | 1.595037e+06 | 144132 |
| **75%** | 1.033094e+07 | 32.960000 | 583.660000 | 413.683200 | 8.405837e+06 | 302149 |
| **max** | 1.044011e+07 | 48.000000 | 2048.030000 | 691.200000 | 8.405837e+06 | 302149 |

## Hypothesis testing 01:

H0 = Mean Cost of Trip is 286

H1 = Mean Cost of Trip is not 286

```
sample_size = int((10/100)*359392) # Considering 10% values as sample data
cost_sample = np.random.choice(df['Cost of Trip'], sample_size)
```

```
sample_size
```

```
35939
```

```
ttest, p_value = ttest_1samp(cost_sample, 286)
```

```
p_value
```

```
0.05510960874795206
```

```
if p_value < 0.05:    # alpha value is 0.05 or 5%
    print("We are rejecting null hypothesis (H0): \nMean Cost of Trip is not 286")
else:
    print("We are accepting null hypothesis (H0): \nMean Cost of Trip is 286")
```

```
We are accepting null hypothesis (H0):
Mean Cost of Trip is 286
```

## Hypothesis testing 02:

### H0 = Mean Price Charged is 423

### H1 = Mean Price Charged is not 423

```
price_sample = np.random.choice(df['Price Charged'], sample_size)
```

```
ttest, p_value = ttest_1samp(price_sample, 423)
p_value
```

```
0.8721769582107021
```

```
if p_value < 0.05:    # alpha value is 0.05 or 5%
    print("We are rejecting null hypothesis (H0): \nMean Price Charged is not 423")
else:
    print("We are accepting null hypothesis (H0): \nMean Price Charged is 423")
```

```
We are accepting null hypothesis (H0):
Mean Price Charged is 423
```

## Hypothesis testing 03:

### H0 = Mean Age is 35

### H1 = Mean Age is not 35

```
age_sample = np.random.choice(df.Age, sample_size)
```

```
ttest, p_value = ttest_1samp(age_sample, 35)
p_value
```

```
5.6915316899835056e-09
```

```
if p_value < 0.05:    # alpha value is 0.05 or 5%
    print("We are rejecting null hypothesis (H0): \nMean Age is not 35")
else:
    print("We are accepting null hypothesis (H0): \nMean Age is 35")
```

```
We are rejecting null hypothesis (H0):
Mean Age is not 35
```

## Hypothesis testing 04:

## H0 = Mean KM Travelled is 22.5 KM

## H1 = Mean KM Travelled is not 22.5 KM

```
km_sample = np.random.choice(df['KM Travelled'], sample_size)
```

```
ttest, p_value = ttest_1samp(km_sample, 22.5)
```

```
p_value
```

```
0.34875967961536747
```

```
if p_value < 0.05:    # alpha value is 0.05 or 5%
    print("We are rejecting null hypothesis (H0): \nMean KM Travelled is not 22.5 KM")
else:
    print("We are accepting null hypothesis (H0): \nMean KM Travelled is 22.5 KM")
```

```
We are accepting null hypothesis (H0):
Mean KM Travelled is 22.5 KM
```

## Hypothesis testing 05:

## H0 = Mean Profit Percentage per Trip is 47.5%

## H1 = Mean Profit Percentage per Trip is not 47.5%

```
pppt_sample = np.random.choice(df['Profit Percentage per Trip'], sample_size)
```

```
ttest, p_value = ttest_1samp(pppt_sample, 47.5)
```

```
p_value
```

```
0.816652764164575
```

```
if p_value < 0.05:    # alpha value is 0.05 or 5%
    print("We are rejecting null hypothesis (H0): \nMean Profit Percentage per Trip is not
else:
    print("We are accepting null hypothesis (H0): \nMean Profit Percentage per Trip is 47.
```

```
We are accepting null hypothesis (H0):
Mean Profit Percentage per Trip is 47.5%
```

## Hypothesis testing (Two-sample T-test) 06:

### H0 = Mean Price charged by Pink, Yellow Cabs are equal

### H1 = Mean Price charged by Pink, Yellow Cabs are not equal

```
df['Price Charged per KM'].groupby(df['Company']).mean()
```

```
Company
Pink Cab      13.768510
Yellow Cab    20.306073
Name: Price Charged per KM, dtype: float64
```

```
price_per_km_sample_yellow = np.random.choice(df[df['Company'] == 'Yellow Cab']['Price Cha
price_per_km_sample_pink = np.random.choice(df[df['Company'] == 'Pink Cab']['Price Charged
```

```
ttest, p_value = ttest_ind(price_per_km_sample_yellow, price_per_km_sample_pink, equal_var
p_value
```

```
0.0
```

```
if p_value < 0.05:    # alpha value is 0.05 or 5%
    print("We are rejecting null hypothesis (H0): \nMean Price charged by Pink, Yellow Cab
else:
    print("We are accepting null hypothesis (H0): \nMean Price charged by Pink, Yellow Cab
```

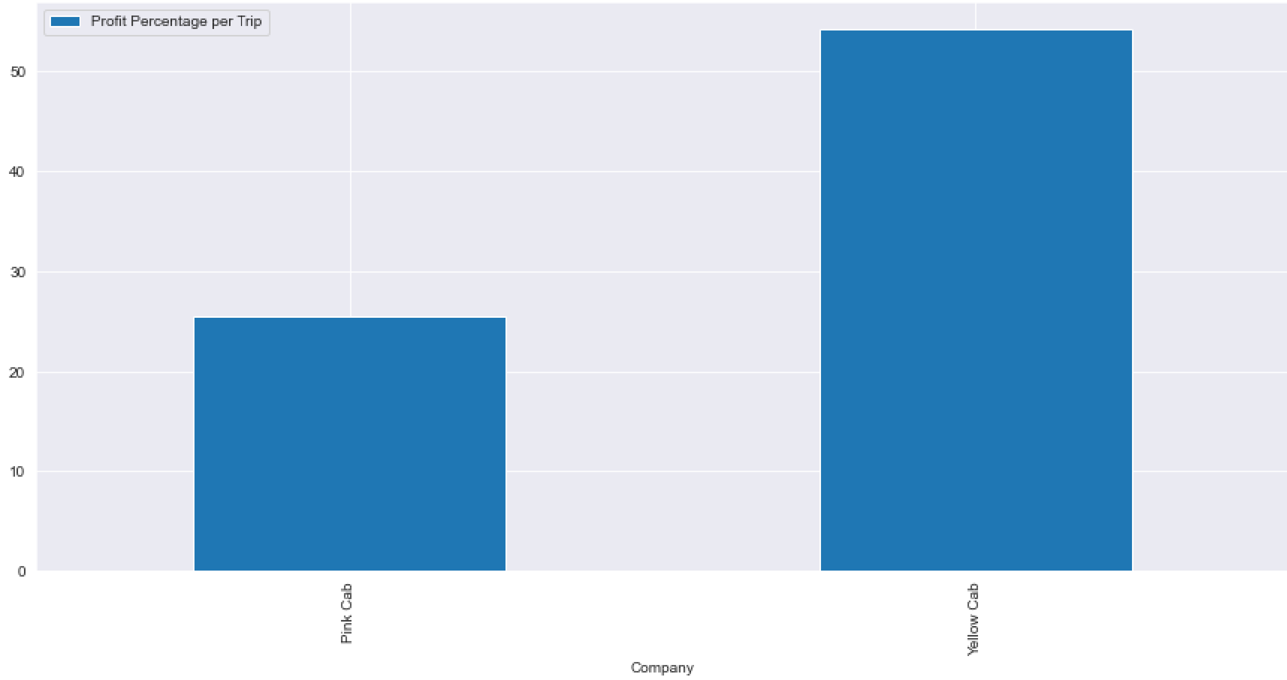
```
We are rejecting null hypothesis (H0):
Mean Price charged by Pink, Yellow Cabs are not equal
```

## Hypothesis testing (Two-sample T-test) 07:

### H0 = Profit Percentage per Trip is equal for both cab service providers

### H1 = Profit Percentage per Trip is not equal for both cab service providers

```
df[['Profit Percentage per Trip', 'Company']].groupby('Company').mean().plot(kind='bar', f
plt.show()
```




```
df['Profit Percentage per Trip'].groupby(df['Company']).mean()
```

```
Company
Pink Cab      25.559567
Yellow Cab    54.296631
Name: Profit Percentage per Trip, dtype: float64
```

```
pppt_sample_yellow = np.random.choice(df[df['Company'] == 'Yellow Cab']['Profit Percentage
pppt_sample_pink = np.random.choice(df[df['Company'] == 'Pink Cab']['Profit Percentage per
```

```
ttest, p_value = ttest_ind(price_per_km_sample_yellow, price_per_km_sample_pink, equal_var
p_value
```

```
0.0
```

```
if p_value < 0.05:    # alpha value is 0.05 or 5%
    print("We are rejecting null hypothesis (H0): \nProfit Percentage per Trip is not equa
else:
    print("We are accepting null hypothesis (H0): \nProfit Percentage per Trip is equal fo
```

```
We are rejecting null hypothesis (H0):
Profit Percentage per Trip is not equal for both cab service providers
```

# Conclusion

- No duplicate data was found
- People prefer Yellow Cabs over Pink Cabs in every city except these 4:

1. Nashville
2. Pittsburgh
3. Sacramento
4. San Diego

- New York City has the highest Profit per KM while Sacramenyo has the lowest Profit per KM
- Avg distance travelled is 22.5 KM
- Over the weekends: distance travelled increases slightly => Profit increases
- Top 5 cities with highest avg profit (in descending order):

1. New York
2. Dallas
3. Silicon Valley
4. Miami
5. Orange County

- Except Chicago, Yellow Cab has more profit margin in each city.
- Around 30% of the population in Washington DC and Boston use cab services whereas for all other cities it's less than 10%
- Most of the users are aged between 20 to 40 years
- Avg Price Charged per KM for Yellow Cab is 20.3 USD & for Pink Cab is 13.76 USD
- Profit Percentage per Trip for
- Pink Cab is 25.559567
- Yellow Cab is 54.296631
- Mean Profit Percentage per Trip is 47.5%