G2M insight for Cab Investment firm

In [1]:
```
# Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime, date, timedelta
```

There are four dataset : Transaction_ID.csv, Customer_ID.csv, Cab_Data.csv, City.csv

In [2]:
```
# Loading the datasets
transaction = pd.read_csv("Transaction_ID.csv")
customer = pd.read_csv("Customer_ID.csv")
cab = pd.read_csv("Cab_Data.csv")
city = pd.read_csv("City.csv")
```

Analysing the datasets

Transaction:

In [3]:
```
transaction.head(5)
```

Out[3]:

	Transaction ID	Customer ID	Payment_Mode
0	10000011	29290	Card
1	10000012	27703	Card
2	10000013	28712	Cash
3	10000014	28020	Cash
4	10000015	27182	Card

In [4]:
```
transaction.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440098 entries, 0 to 440097
Data columns (total 3 columns):
 #   Column          Non-Null Count   Dtype 
---  ------          --------------   ----- 
 0   Transaction ID  440098 non-null  int64 
 1   Customer ID     440098 non-null  int64 
 2   Payment_Mode    440098 non-null  object
dtypes: int64(2), object(1)
memory usage: 10.1+ MB
```

In [5]:
```
transaction.shape
```

Out[5]:
```
(440098, 3)
```

Customer:

In [6]:
```
customer.head(5)
```

Out[6]:

	Customer ID	Gender	Age	Income (USD/Month)
0	29290	Male	28	10813
1	27703	Male	27	9237
2	28712	Male	53	11242
3	28020	Male	23	23327
4	27182	Male	33	8536

In [7]:
```
customer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49171 entries, 0 to 49170
Data columns (total 4 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Customer ID          49171 non-null  int64 
 1   Gender               49171 non-null  object
 2   Age                  49171 non-null  int64 
 3   Income (USD/Month)   49171 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 1.5+ MB
```

In [8]:
```
customer.shape
```

Out[8]:
```
(49171, 4)
```

In [9]:
```
customer.describe()
```

Out[9]:

	Customer ID	Age	Income (USD/Month)
count	49171.000000	49171.000000	49171.000000
mean	28398.252283	35.363121	15015.631856
std	17714.137333	12.599066	8002.208253
min	1.000000	18.000000	2000.000000
25%	12654.500000	25.000000	8289.500000
50%	27631.000000	33.000000	14656.000000
75%	43284.500000	42.000000	21035.000000
max	60000.000000	65.000000	35000.000000

Cab:

In [10]:
```
cab.head(5)
```

Out[10]:

	Transaction ID	Date of Travel	Company	City	KM Travelled	Price Charged	Cost of Trip
0	10000011	42377	Pink Cab	ATLANTA GA	30.45	370.95	313.635
1	10000012	42375	Pink Cab	ATLANTA GA	28.62	358.52	334.854
2	10000013	42371	Pink Cab	ATLANTA GA	9.04	125.20	97.632
3	10000014	42376	Pink Cab	ATLANTA GA	33.17	377.40	351.602
4	10000015	42372	Pink Cab	ATLANTA GA	8.73	114.62	97.776

In [11]:
```
cab["Date of Travel"] = cab["Date of Travel"].apply(lambda x: date(1899, 12,30) + time
cab.head(5)
```

Out[11]:

	Transaction ID	Date of Travel	Company	City	KM Travelled	Price Charged	Cost of Trip
0	10000011	2016-01-08	Pink Cab	ATLANTA GA	30.45	370.95	313.635
1	10000012	2016-01-06	Pink Cab	ATLANTA GA	28.62	358.52	334.854
2	10000013	2016-01-02	Pink Cab	ATLANTA GA	9.04	125.20	97.632
3	10000014	2016-01-07	Pink Cab	ATLANTA GA	33.17	377.40	351.602
4	10000015	2016-01-03	Pink Cab	ATLANTA GA	8.73	114.62	97.776

In [12]:
```
cab.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 359392 entries, 0 to 359391
Data columns (total 7 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   Transaction ID  359392 non-null  int64  
 1   Date of Travel  359392 non-null  object 
 2   Company         359392 non-null  object 
 3   City            359392 non-null  object 
 4   KM Travelled    359392 non-null  float64
 5   Price Charged   359392 non-null  float64
 6   Cost of Trip    359392 non-null  float64
dtypes: float64(3), int64(1), object(3)
memory usage: 19.2+ MB
```

In [13]:
```
cab.shape
```

Out[13]:
```
(359392, 7)
```

In [14]:
```
cab.describe()
```

Out[14]:

	Transaction ID	KM Travelled	Price Charged	Cost of Trip
count	3.593920e+05	359392.000000	359392.000000	359392.000000
mean	1.022076e+07	22.567254	423.443311	286.190113
std	1.268058e+05	12.233526	274.378911	157.993661
min	1.000001e+07	1.900000	15.600000	19.000000
25%	1.011081e+07	12.000000	206.437500	151.200000
50%	1.022104e+07	22.440000	386.360000	282.480000
75%	1.033094e+07	32.960000	583.660000	413.683200
max	1.044011e+07	48.000000	2048.030000	691.200000

City

In [15]:
```
city.head(5)
```

Out[15]:

	City	Population	Users
0	NEW YORK NY	8,405,837	302,149
1	CHICAGO IL	1,955,130	164,468
2	LOS ANGELES CA	1,595,037	144,132
3	MIAMI FL	1,339,155	17,675
4	SILICON VALLEY	1,177,609	27,247

In [16]:
```
city.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   City        20 non-null     object
 1   Population  20 non-null     object
 2   Users       20 non-null     object
dtypes: object(3)
memory usage: 608.0+ bytes
```

In [17]:
```
city.shape
```

Out[17]:
```
(20, 3)
```

In [18]:
```
city.describe()
```

Out[18]:

	City	Population	Users
count	20	20	20
unique	20	20	20
top	NEW YORK NY	8,405,837	302,149
freq	1	1	1

Removing ',' in the count of population and users to correct the data type

In [19]:
```
city['Population'] = city['Population'].str.replace(',', '').astype(int)
```

In [20]:
```
city['Users'] = city['Users'].str.replace(',', '').astype(int)
```

In [21]:
```
city.head(5)
```

Out[21]:

	City	Population	Users
0	NEW YORK NY	8405837	302149
1	CHICAGO IL	1955130	164468
2	LOS ANGELES CA	1595037	144132
3	MIAMI FL	1339155	17675
4	SILICON VALLEY	1177609	27247

Merging Transaction_ID and Customer_ID and resultant with with Cab_Data

In [22]:
```
customer_transaction = customer.merge(transaction, on = 'Customer ID')
customer_transaction.columns
```

Out[22]:
```
Index(['Customer ID', 'Gender', 'Age', 'Income (USD/Month)', 'Transaction ID',
       'Payment_Mode'],
      dtype='object')
```

In [23]:
```
customer_transaction.head()
```

Out[23]:

	Customer ID	Gender	Age	Income (USD/Month)	Transaction ID	Payment_Mode
0	29290	Male	28	10813	10000011	Card
1	29290	Male	28	10813	10351127	Cash
2	29290	Male	28	10813	10412921	Card
3	27703	Male	27	9237	10000012	Card
4	27703	Male	27	9237	10320494	Card

In [24]:
```
cab_with_customer_transaction = cab.merge(customer_transaction, on = ['Transaction ID'
cab_with_customer_transaction.columns
```

Out[24]:
```
Index(['Transaction ID', 'Date of Travel', 'Company', 'City', 'KM Travelled',
       'Price Charged', 'Cost of Trip', 'Customer ID', 'Gender', 'Age',
       'Income (USD/Month)', 'Payment_Mode'],
      dtype='object')
```

In [25]:
```
cab_with_customer_transaction.head(5)
```

Out[25]:

	Transaction ID	Date of Travel	Company	City	KM Travelled	Price Charged	Cost of Trip	Customer ID	Gender	Age	(USD
0	10000011	2016-01-08	Pink Cab	ATLANTA GA	30.45	370.95	313.635	29290	Male	28	
1	10000012	2016-01-06	Pink Cab	ATLANTA GA	28.62	358.52	334.854	27703	Male	27	
2	10000013	2016-01-02	Pink Cab	ATLANTA GA	9.04	125.20	97.632	28712	Male	53	
3	10000014	2016-01-07	Pink Cab	ATLANTA GA	33.17	377.40	351.602	28020	Male	23	
4	10000015	2016-01-03	Pink Cab	ATLANTA GA	8.73	114.62	97.776	27182	Male	33	

Now merging with city data to get master data set

In [26]:
```
data = cab_with_customer_transaction.merge(city, on = 'City')
data.columns
```

Out[26]:
```
Index(['Transaction ID', 'Date of Travel', 'Company', 'City', 'KM Travelled',
       'Price Charged', 'Cost of Trip', 'Customer ID', 'Gender', 'Age',
       'Income (USD/Month)', 'Payment_Mode', 'Population', 'Users'],
      dtype='object')
```

In [27]:
```
data.head(5)
```

Out[27]:

	Transaction ID	Date of Travel	Company	City	KM Travelled	Price Charged	Cost of Trip	Customer ID	Gender	Age	(USD
0	10000011	2016-01-08	Pink Cab	ATLANTA GA	30.45	370.95	313.635	29290	Male	28	
1	10000012	2016-01-06	Pink Cab	ATLANTA GA	28.62	358.52	334.854	27703	Male	27	
2	10000013	2016-01-02	Pink Cab	ATLANTA GA	9.04	125.20	97.632	28712	Male	53	
3	10000014	2016-01-07	Pink Cab	ATLANTA GA	33.17	377.40	351.602	28020	Male	23	
4	10000015	2016-01-03	Pink Cab	ATLANTA GA	8.73	114.62	97.776	27182	Male	33	

In [28]:
```
data.shape
```

Out[28]:
```
(359392, 14)
```

In [29]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 359392 entries, 0 to 359391
Data columns (total 14 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   Transaction ID       359392 non-null  int64  
 1   Date of Travel       359392 non-null  object 
 2   Company              359392 non-null  object 
 3   City                 359392 non-null  object 
 4   KM Travelled         359392 non-null  float64
 5   Price Charged        359392 non-null  float64
 6   Cost of Trip         359392 non-null  float64
 7   Customer ID          359392 non-null  int64  
 8   Gender               359392 non-null  object 
 9   Age                  359392 non-null  int64  
 10  Income (USD/Month)   359392 non-null  int64  
 11  Payment_Mode         359392 non-null  object 
 12  Population           359392 non-null  int64  
 13  Users                359392 non-null  int64  
dtypes: float64(3), int64(6), object(5)
memory usage: 41.1+ MB
```

In [30]:
```
data.describe()
```

Out[30]:

	Transaction ID	KM Travelled	Price Charged	Cost of Trip	Customer ID	Age
count	3.593920e+05	359392.000000	359392.000000	359392.000000	359392.000000	359392.000000
mean	1.022076e+07	22.567254	423.443311	286.190113	19191.652115	35.336705
std	1.268058e+05	12.233526	274.378911	157.993661	21012.412463	12.594234
min	1.000001e+07	1.900000	15.600000	19.000000	1.000000	18.000000
25%	1.011081e+07	12.000000	206.437500	151.200000	2705.000000	25.000000
50%	1.022104e+07	22.440000	386.360000	282.480000	7459.000000	33.000000
75%	1.033094e+07	32.960000	583.660000	413.683200	36078.000000	42.000000
max	1.044011e+07	48.000000	2048.030000	691.200000	60000.000000	65.000000

Checking for null and duplicate values

In [31]:
```
data.duplicated().sum()
```

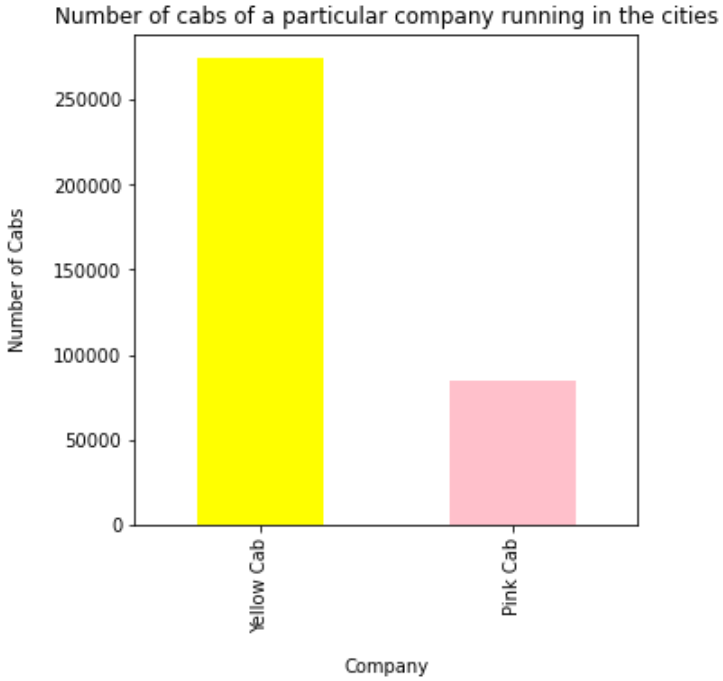
Out[31]:
```
0
```

In [32]:
```
data.isna().sum()
```

Out[32]:
```
Transaction ID        0
Date of Travel        0
Company               0
City                  0
KM Travelled          0
Price Charged         0
Cost of Trip          0
Customer ID           0
Gender                0
Age                   0
Income (USD/Month)    0
Payment_Mode          0
Population            0
Users                 0
dtype: int64
```

Analysing our data:

In [57]:
```
data['Company'].value_counts().plot(kind='bar', figsize=(5, 5), color=['yellow', 'pink
plt.xlabel("Company", labelpad=14)
plt.ylabel("Number of Cabs", labelpad=14)
plt.title("Number of cabs of a particular company running in the cities");
```



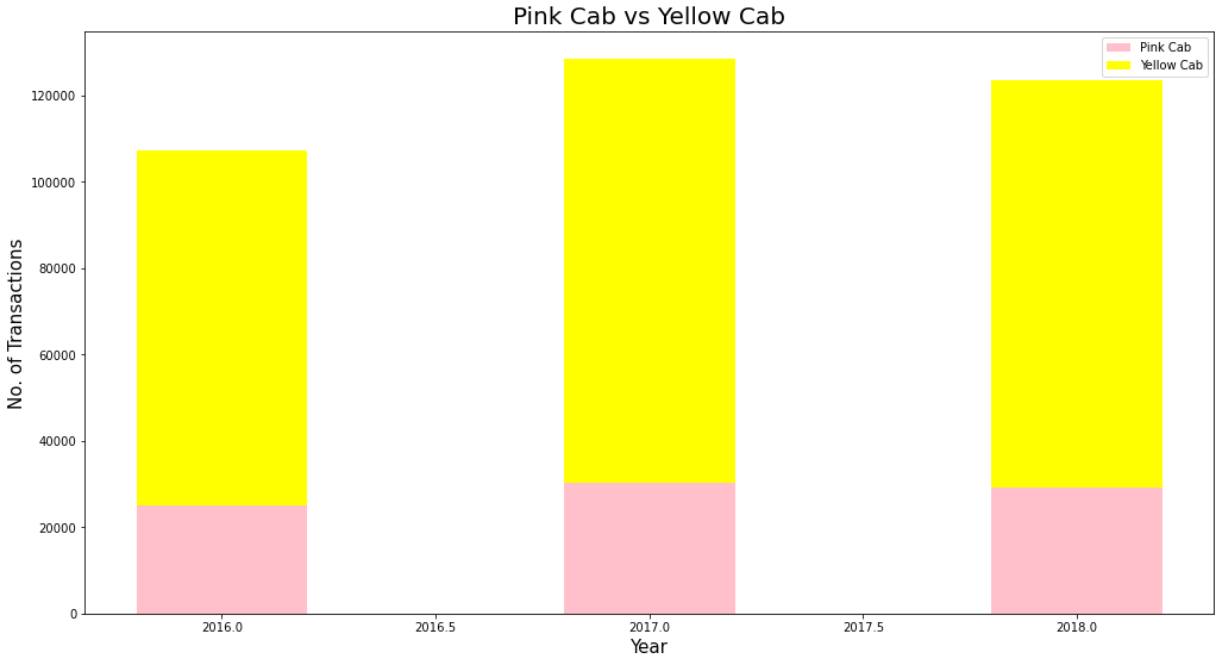
By looking at the above plot we cal see that Yellow cab has more cabs thank Pink cabs company

In [34]:
```
# Convert 'Date of Travel' to datetime format
data['Date of Travel'] = pd.to_datetime(data['Date of Travel'])

pink_transaction = data[data['Company'] == 'Pink Cab'].groupby(data['Date of Travel'].
yellow_transaction = data[data['Company'] == 'Yellow Cab'].groupby(data['Date of Trave

labels = pink_transaction.index.unique()

plt.figure(figsize=(17, 9))
ax1 = plt.subplot(111)
ax1.bar(labels, pink_transaction.values, width=0.4, color='pink', align='center', labe
ax1.bar(labels, yellow_transaction.values, width=0.4, color='yellow', align='center',
plt.title("Pink Cab vs Yellow Cab", fontsize=20)
plt.ylabel('No. of Transactions', fontsize=15)
plt.xlabel('Year', fontsize=15)
plt.legend()
plt.show()
```



Yellow cab has more transactions than pink, as is event from the higher number of cabs

In [35]:
```
# Calculating profit
data['Profit'] = data['Price Charged'] - data['Cost of Trip']
```

In [48]:
```
data['Profit']
```

Out[48]:
```
0         57.3150
1         23.6660
2         27.5680
3         25.7980
4         16.8440
            ...
359387     5.8800
359388     6.9020
359389    87.4200
359390    32.1420
359391    13.9608
Name: Profit, Length: 359392, dtype: float64
```

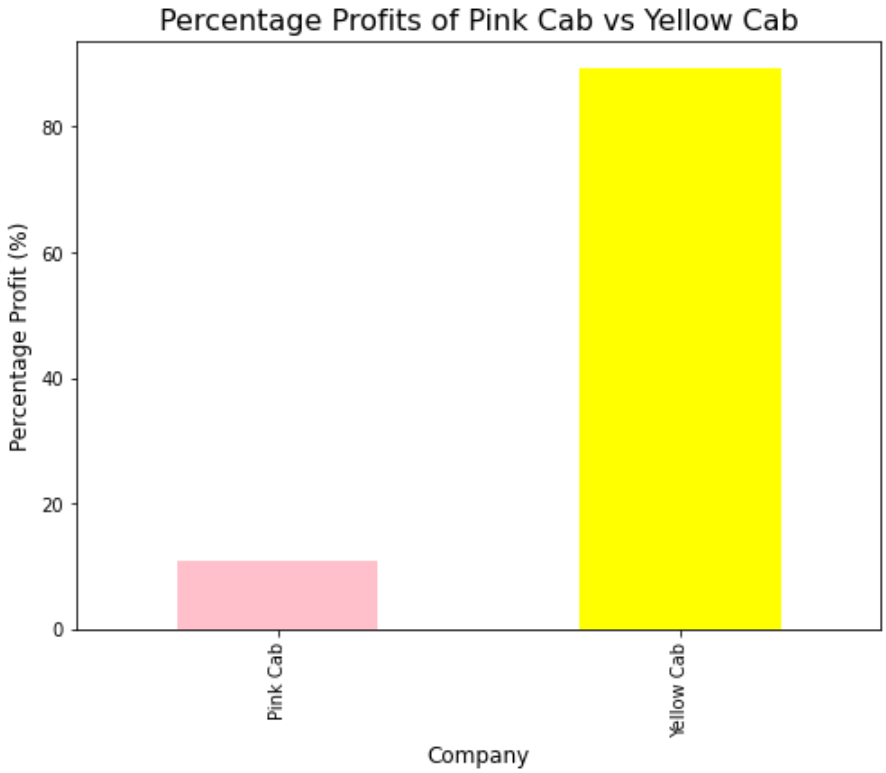
In [58]:
```
profit_by_company = data.groupby('Company')['Profit'].sum()

total_profit = profit_by_company.sum()

# Calculate percentage profits
percentage_profits = (profit_by_company / total_profit) * 100
print(percentage_profits)

plt.figure(figsize=(8, 6))
percentage_profits.plot(kind='bar', color=['pink', 'yellow'])
plt.title("Percentage Profits of Pink Cab vs Yellow Cab", fontsize=16)
plt.ylabel('Percentage Profit (%)', fontsize=12)
plt.xlabel('Company', fontsize=12)
plt.show()
```

```
Company
Pink Cab      10.759326
Yellow Cab    89.240674
Name: Profit, dtype: float64
```

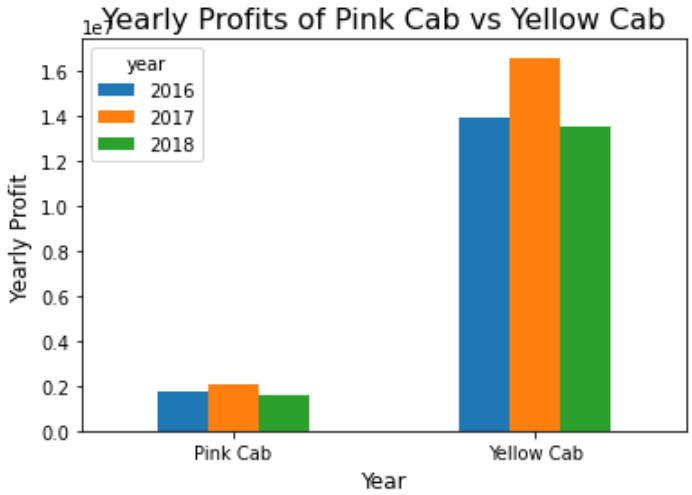


In [60]:
```
# Calculate yearly profit for each company
yearly_profit_by_company = data.groupby(['Company', data['Date of Travel'].dt.year])['

# Unstack the data to have 'Company' as columns and years as index
yearly_profit_by_company = yearly_profit_by_company.unstack()

plt.figure(figsize=(10, 6))
yearly_profit_by_company.plot(kind='bar')
plt.title("Yearly Profits of Pink Cab vs Yellow Cab", fontsize=16)
plt.ylabel('Yearly Profit', fontsize=12)
plt.xlabel('Year', fontsize=12)
plt.xticks(rotation=0)
plt.legend(title='year')
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```

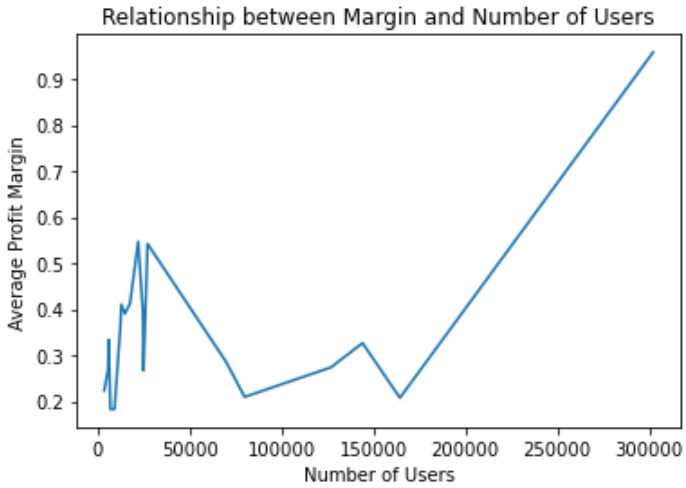


In [64]:
```
# Calculate profit margin for each transaction
data['Profit Margin'] = (data['Price Charged'] - data['Cost of Trip']) / data['Cost of

# Group by number of users and calculate the average profit margin
average_margin_by_users = data.groupby('Users')['Profit Margin'].mean()

# Plot the relationship between average margin and number of users
plt.plot(average_margin_by_users.index, average_margin_by_users.values)
plt.xlabel('Number of Users')
```

```
plt.ylabel('Average Profit Margin')
plt.title('Relationship between Margin and Number of Users')
plt.show()
```



Clearly the profit of yellow cabs is much higher than pink cabs across years

In [ ]:

In [ ]: