In [372…
```
import numpy as np # linear algebra
import matplotlib.pyplot as plt
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
from scipy import stats
import plotly.express as px
import plotly.graph_objs as go
import plotly
import plotly.graph_objects as go
#import datetime
from datetime import datetime
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [373…
```
df_caby = pd.read_excel('/Users/srilathasirigala/Documents/Intern/CabData/DataSets/Cabydata.xlsx')
df_citi = pd.read_csv('/Users/srilathasirigala/Documents/Intern/CabData/DataSets/City.csv')
df_cust = pd.read_csv('/Users/srilathasirigala/Documents/Intern/CabData/DataSets/Customer_ID.csv')
df_tra = pd.read_csv('/Users/srilathasirigala/Documents/Intern/CabData/DataSets/Transaction_ID.csv')
```

In [374…
```
df_caby.head()
```

Out[374]:

	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	2016-01-09	Pink Cab	ATLANTA GA	28.62	358.52	334.854
2	10000013	2016-01-10	Pink Cab	ATLANTA GA	9.04	125.20	97.632
3	10000014	2016-01-11	Pink Cab	ATLANTA GA	33.17	377.40	351.602
4	10000015	2016-01-12	Pink Cab	ATLANTA GA	8.73	114.62	97.776

In [375…
```
df_caby.info()
df_citi.info()
df_cust.info()
df_tra.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 346700 entries, 0 to 346699
Data columns (total 7 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Transaction ID  346700 non-null  int64
 1   Date of Travel  346700 non-null  datetime64[ns]
 2   Company         346700 non-null  object
 3   City            346700 non-null  object
 4   KM Travelled    346700 non-null  float64
 5   Price Charged   346700 non-null  float64
 6   Cost of Trip    346700 non-null  float64
dtypes: datetime64[ns](1), float64(3), int64(1), object(2)
memory usage: 18.5+ MB
<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
<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
<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 [376…
```
df_caby['Date of Travel'] = pd.to_datetime(df_caby['Date of Travel'])
df_caby = df_caby.rename(columns ={'Date of Travel': 'Date'})
df_caby.info
```

Out[376]:
```
<bound method DataFrame.info of         Transaction ID       Date     Company          City  KM Travelled  \
0              10000011 2016-01-08    Pink Cab     ATLANTA GA         30.45
1              10000012 2016-01-09    Pink Cab     ATLANTA GA         28.62
2              10000013 2016-01-10    Pink Cab     ATLANTA GA          9.04
3              10000014 2016-01-11    Pink Cab     ATLANTA GA         33.17
4              10000015 2016-01-12    Pink Cab     ATLANTA GA          8.73
...                 ...        ...         ...            ...          ...
346695         10439960 2018-12-31  Yellow Cab  WASHINGTON DC         33.93
346696         10439984 2018-12-31  Yellow Cab  WASHINGTON DC         40.00
346697         10440028 2018-12-31  Yellow Cab  WASHINGTON DC         26.22
346698         10440034 2018-12-31  Yellow Cab  WASHINGTON DC         34.68
346699         10440093 2018-12-31  Yellow Cab  WASHINGTON DC          4.32

        Price Charged  Cost of Trip
0              370.95      313.6350
1              358.52      334.8540
2              125.20       97.6320
3              377.40      351.6020
4              114.62       97.7760
...               ...           ...
346695         474.47      411.2316
346696         641.78      484.8000
346697         405.25      327.2256
346698         505.38      470.2608
346699          60.41       55.4688

[346700 rows x 7 columns]>
```

In [377…
```
JoinedData = df_caby.merge(df_tra, on= 'Transaction ID').merge(df_cust, on ='Customer ID').merge(df_citi, on =
```

In [378…
```
MergeData=JoinedData.dropna()
```

In [379…
```
MergeData.describe()
```

Out[379]:

	Transaction ID	KM Travelled	Price Charged	Cost of Trip	Customer ID	Age	Income (USD/Month)
count	3.467000e+05	346700.000000	346700.000000	346700.000000	346700.000000	346700.000000	346700.000000
mean	1.022850e+07	22.563486	421.803841	286.066880	19137.463412	35.330358	15047.438699
std	1.223676e+05	12.232157	272.799700	157.944536	20980.836512	12.594697	7967.149392
min	1.000001e+07	1.900000	15.600000	19.000000	1.000000	18.000000	2000.000000
25%	1.012201e+07	12.000000	205.890000	151.200000	2689.000000	25.000000	8429.750000
50%	1.022897e+07	22.440000	385.240000	282.240000	7451.000000	33.000000	14680.000000
75%	1.033479e+07	32.960000	581.680000	413.586000	35844.000000	42.000000	21035.000000
max	1.044011e+07	48.000000	2048.030000	691.200000	60000.000000	65.000000	35000.000000

In [380…
```
MergeData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 346700 entries, 0 to 346699
Data columns (total 14 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   Transaction ID      346700 non-null  int64
 1   Date                346700 non-null  datetime64[ns]
 2   Company             346700 non-null  object
 3   City                346700 non-null  object
 4   KM Travelled        346700 non-null  float64
 5   Price Charged       346700 non-null  float64
 6   Cost of Trip        346700 non-null  float64
 7   Customer ID         346700 non-null  int64
 8   Payment_Mode        346700 non-null  object
 9   Gender              346700 non-null  object
 10  Age                 346700 non-null  int64
 11  Income (USD/Month)  346700 non-null  int64
 12  Population          346700 non-null  object
 13  Users               346700 non-null  object
dtypes: datetime64[ns](1), float64(3), int64(4), object(6)
memory usage: 39.7+ MB
```

In [381…
```
MergeData.isnull().sum()
```

Out[381]:
```
Transaction ID        0
Date                  0
Company               0
City                  0
KM Travelled          0
Price Charged         0
Cost of Trip          0
Customer ID           0
Payment_Mode          0
Gender                0
Age                   0
Income (USD/Month)    0
Population            0
Users                 0
dtype: int64
```

In [382…
```
for column in MergeData.columns:
    if ' ' in column:
        MergeData = MergeData.rename(columns={column:column.replace(' ','_')})

for column in ["Population","Users"] :

        MergeData[column] = MergeData[column].str.replace(',','')

MergeData.head(10)
```

Out[382]:

	Transaction_ID	Date	Company	City	KM_Travelled	Price_Charged	Cost_of_Trip	Customer_ID	Payment_Mode	Gender	Age	Inco
0	10000011	2016-01-08	Pink Cab	ATLANTA GA	30.45	370.95	313.6350	29290	Card	Male	28	
1	10351127	2018-07-21	Yellow Cab	ATLANTA GA	26.19	598.70	317.4228	29290	Cash	Male	28	
2	10412921	2018-11-23	Yellow Cab	ATLANTA GA	42.55	792.05	597.4020	29290	Card	Male	28	
3	10000012	2016-01-09	Pink Cab	ATLANTA GA	28.62	358.52	334.8540	27703	Card	Male	27	
4	10320494	2018-04-21	Yellow Cab	ATLANTA GA	36.38	721.10	467.1192	27703	Card	Male	27	
5	10324737	2018-05-04	Yellow Cab	ATLANTA GA	6.18	138.40	87.5088	27703	Cash	Male	27	
6	10395626	2018-10-27	Pink Cab	ATLANTA GA	13.39	167.03	141.9340	27703	Card	Male	27	
7	10000013	2016-01-10	Pink Cab	ATLANTA GA	9.04	125.20	97.6320	28712	Cash	Male	53	
8	10079404	2016-09-21	Yellow Cab	ATLANTA GA	39.60	704.30	494.2080	28712	Card	Male	53	
9	10186994	2017-06-23	Yellow Cab	ATLANTA GA	18.19	365.63	246.6564	28712	Card	Male	53	

In [383…
```
for column in ["Company", "City" , "Payment_Mode" , "Gender" ] :

        MergeData[column] = MergeData[column].astype('category')


for column in ["Population", "Users" ] :

        MergeData[column] = MergeData[column].astype('int64')

print("\nFeature's datatypes\n\n{}".format(MergeData.dtypes))
```

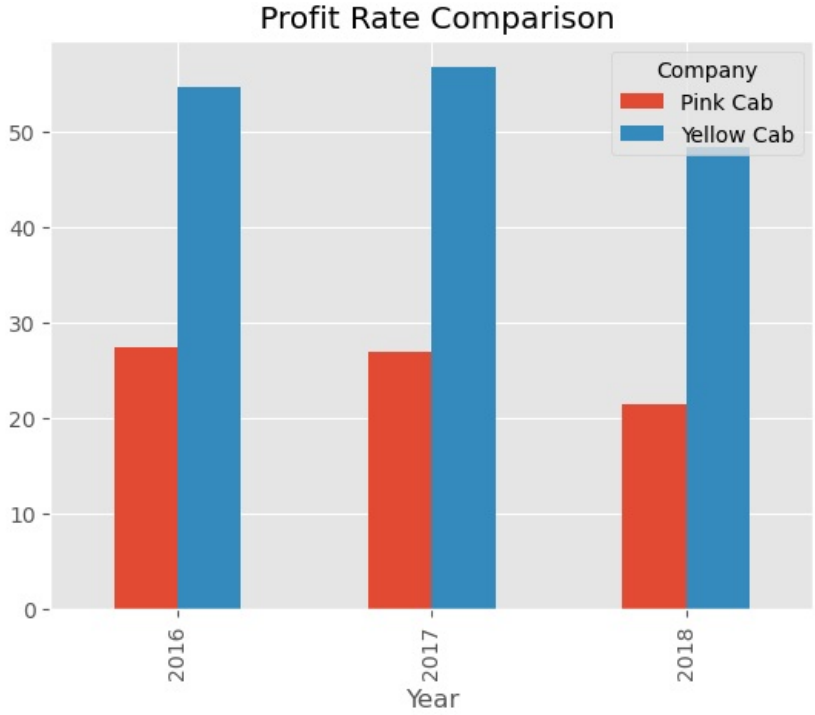
```
Feature's datatypes

Transaction_ID                int64
Date                 datetime64[ns]
Company                    category
City                       category
KM_Travelled                float64
Price_Charged               float64
Cost_of_Trip                float64
Customer_ID                   int64
Payment_Mode               category
Gender                     category
Age                           int64
Income_(USD/Month)            int64
Population                    int64
Users                         int64
dtype: object
```

In [384…
```
##converting the date format into standard date format
MergeData['Year'] = MergeData['Date'].dt.year
MergeData['Month'] = MergeData['Date'].dt.month
MergeData['Profit'] = MergeData['Price_Charged'] - MergeData['Cost_of_Trip']
MergeData['Profit_Rate'] = ((MergeData['Price_Charged'] - MergeData['Cost_of_Trip'])/MergeData['Cost_of_Trip'])
```

In [385…
```
from matplotlib import style
```

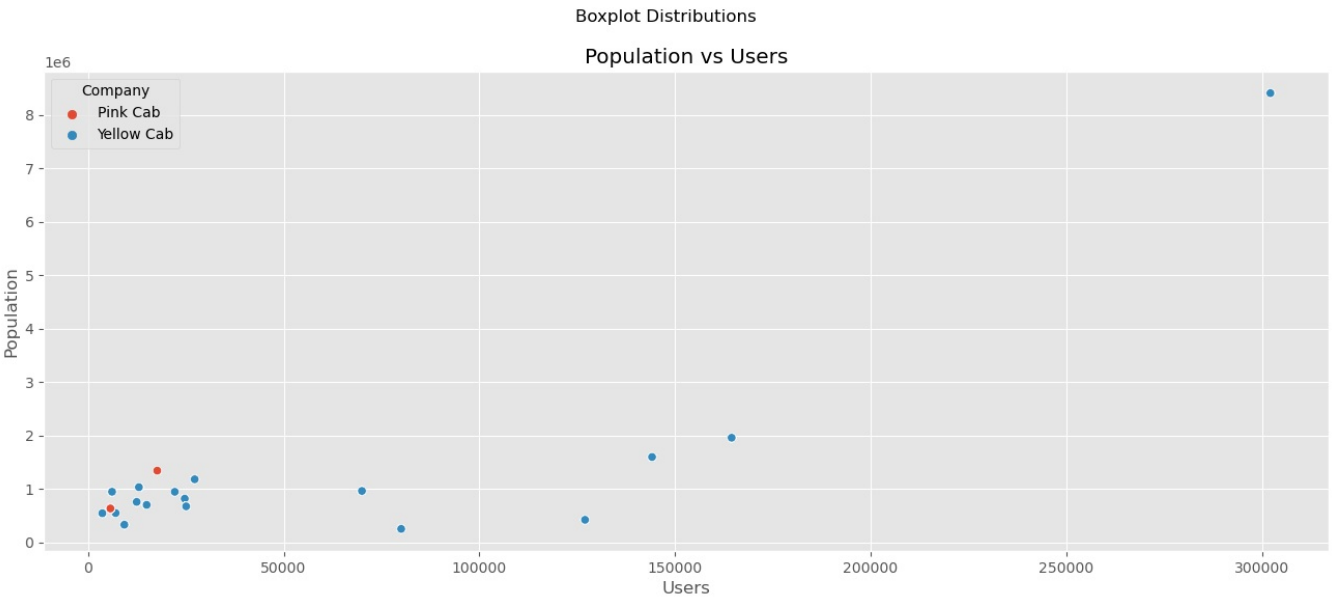
In [386…
```
##Profit Rate Comparison

ProfitRate = MergeData.groupby(['Year','Company']).Profit.sum().unstack()/ MergeData.groupby(['Year','Company']
print("Comparison of two Profit Rates of the company")
print(ProfitRate)
ax = ProfitRate.plot(kind='bar',stacked = False, title = ' Profit Rate Comparison')
plt.show()
```

```
Comparison of two Profit Rates of the company
Company   Pink Cab  Yellow Cab
Year
2016     27.447867   54.776774
2017     26.953767   56.772194
2018     21.429671   48.383983
```



In [387…
```
#Compnay Vs Users vs Populations
fig,axes = plt.subplots(figsize=(16, 6), sharey=True)
fig.suptitle('Boxplot Distributions')
sns.scatterplot(data=MergeData, x='Users', y='Population' , hue="Company"  ).set_title("Population vs Users")
```

Out[387]:
```
Text(0.5, 1.0, 'Population vs Users')
```

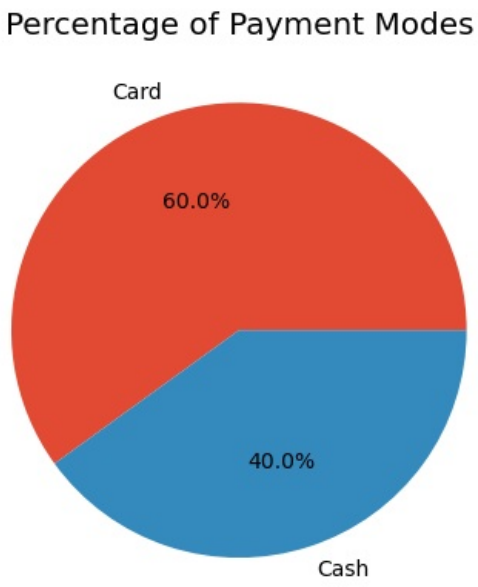


In [388…
```
#Pink and Yellow Cab Firm Users Distribution over City

PinkCabC = MergeData[MergeData["Company"] == "Pink Cab"].groupby("City").count()
YellowCabC = MergeData[MergeData["Company"] == "Yellow Cab"].groupby("City").count()

fig = go.Figure()
fig.add_trace(go.Bar(
                    x=PinkCabC.index,
```

```
                    y=PinkCabC['Users'],
                    name='Pink Cab',
                    marker_color='Pink'
                    ))
fig.add_trace(go.Bar(
                    x=YellowCabC.index,
                    y=YellowCabC['Users'],
                    name='Yellow Cab',
                    marker_color='Yellow'
                    ))
fig.update_layout(
    yaxis_title="Users",
    title="Pink & Yellow Cab Firm Users Distribution Over City" )
```

In [389…

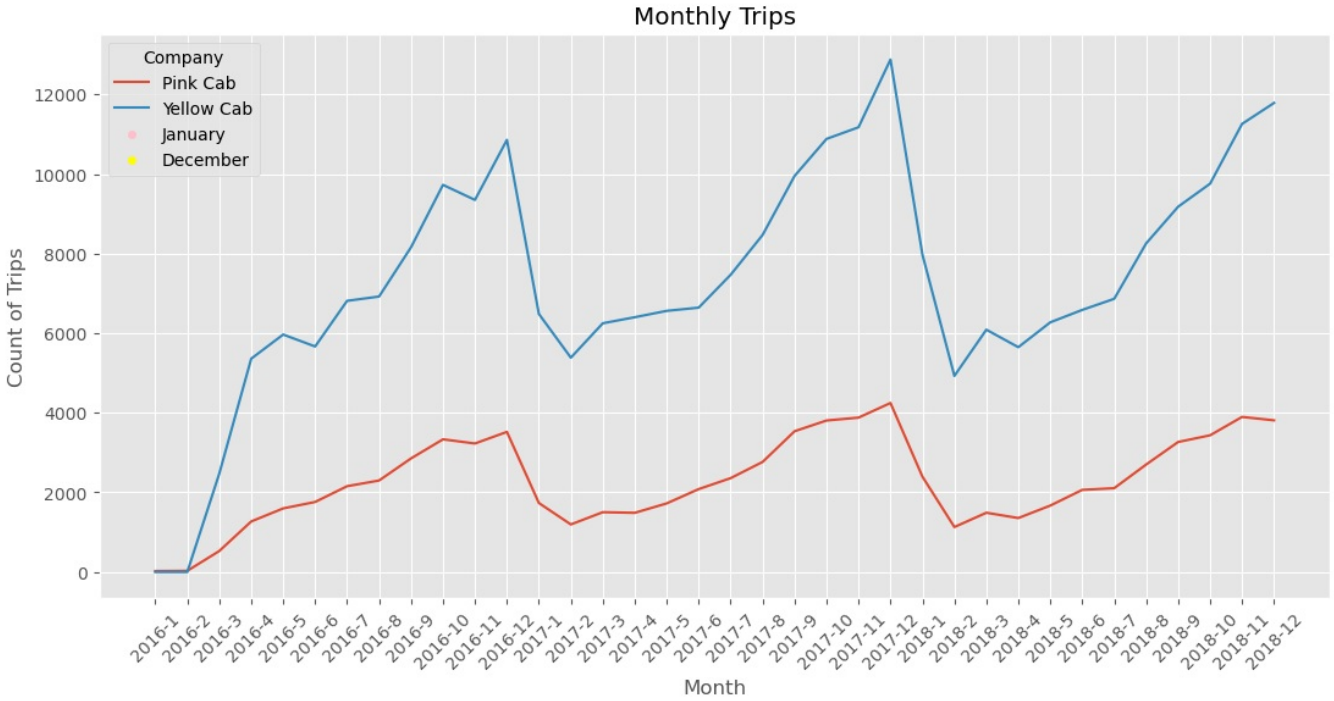
```
#Pie chart to show the percentage of different payment modes used by customer

# Group the data by payment mode and count the number of transactions
transactions_by_payment_mode = MergeData.groupby('Payment_Mode')['Transaction_ID'].count()

# Create a pie chart
plt.pie(transactions_by_payment_mode.values, labels=transactions_by_payment_mode.index, autopct='%1.1f%%')
plt.title('Percentage of Payment Modes')
plt.show()
```



In [390…

```
monthstats = MergeData.groupby(['Year', 'Month', 'Company']).size().reset_index().\
                                                    rename(columns = {0:'count'})
monthstats
monthstats['monthly'] = monthstats['Year'].astype('str') + "-" + monthstats['Month'].astype('str')
```

```
monthstats
plt.figure(figsize = (13,6))
ax = sns.lineplot(x = 'monthly', y = 'count', data = monthstats, hue = 'Company')
for Month, name, color in zip([1,12], ['January', 'December'], ['pink', 'Yellow']):
    monthstats.query(f"Month == '{Month}'")[['monthly', 'count']].\
    plot.scatter(x = 'monthly', y = 'count', ax = ax, label = f'{name}', color = color);

plt.xticks(rotation = 45)
plt.title('Monthly Trips');
plt.xlabel('Month');
plt.ylabel('Count of Trips');
```

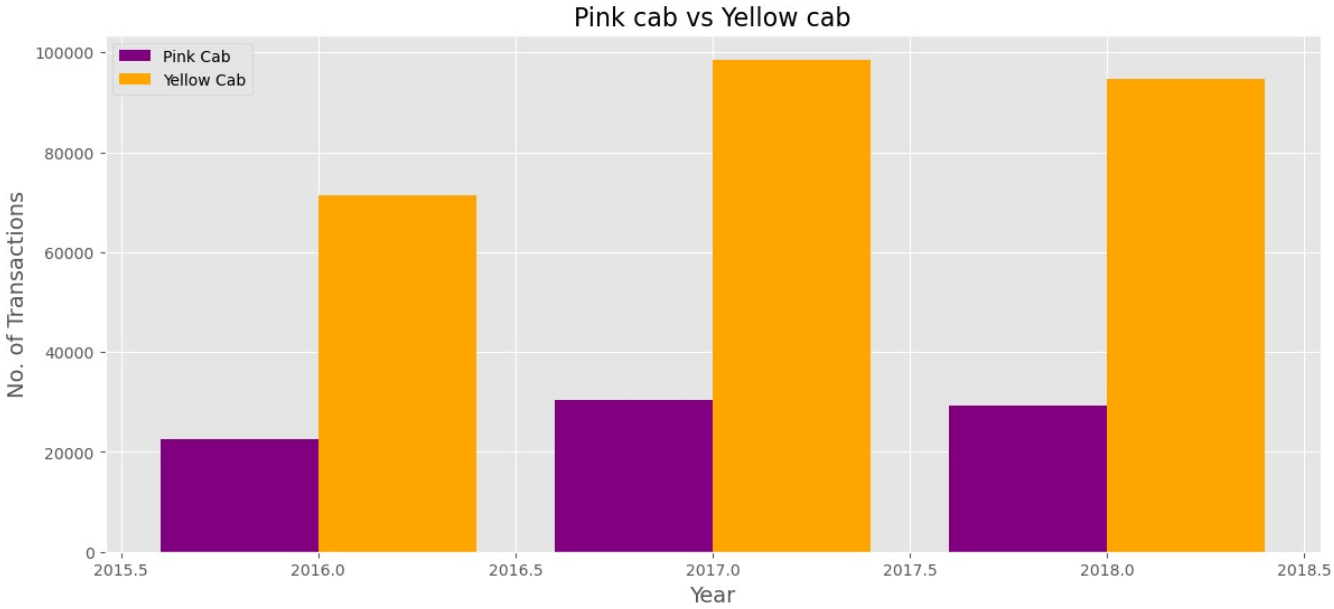


In [391…

```
#Number Of Transactions

fig1 =MergeData[MergeData.Company=='Pink Cab'].groupby('Year').Transaction_ID.count()
fig2 = MergeData[MergeData.Company=='Yellow Cab'].groupby('Year').Transaction_ID.count()
plt.figure(figsize=(14,6))
ax = plt.subplot(111)
ax.bar(fig1.index-0.2,fig1.values, width=0.4, color='purple', align='center',label='Pink Cab')
ax.bar(fig2.index+0.2, fig2.values, width=0.4, color='Orange', align='center',label='Yellow Cab')
plt.title("Pink cab vs Yellow cab",fontsize = 16)
plt.ylabel('No. of Transactions',fontsize = 14)
plt.xlabel('Year',fontsize = 14)
plt.legend()
plt.show
```

Out[391]:

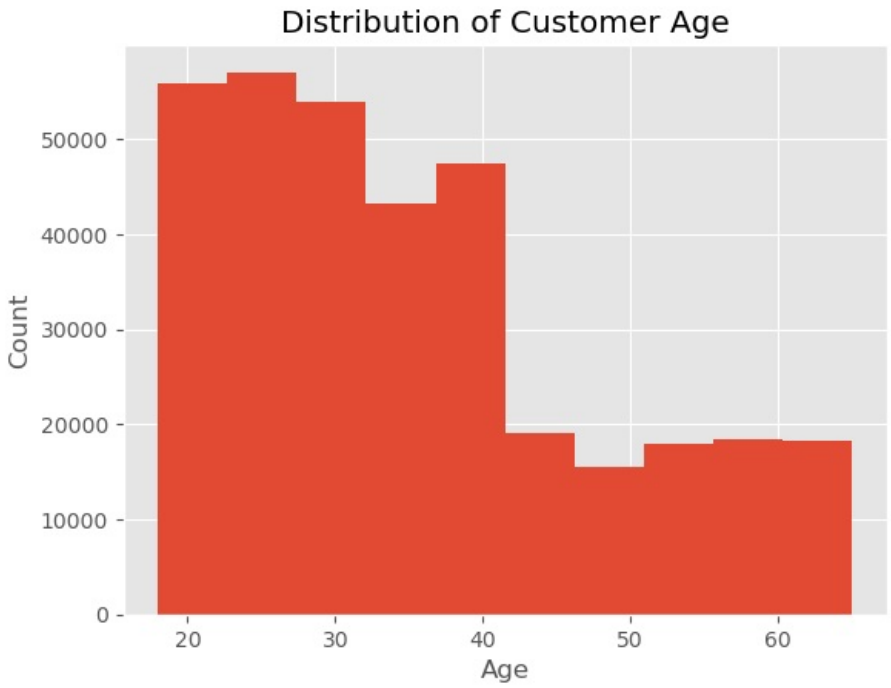
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



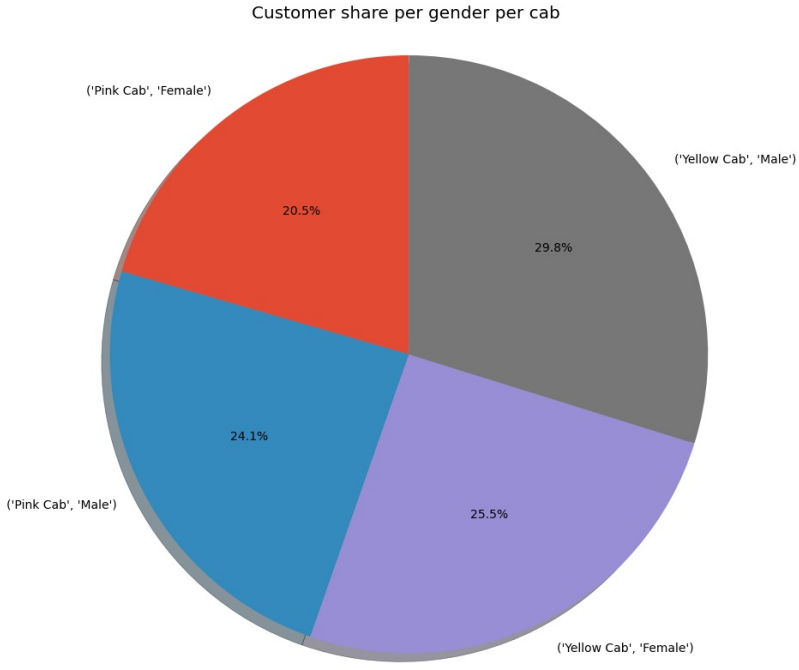
In [392…

```
# Create a histogram
#4. Histogram to show the distribution of age among customers:


plt.hist(MergeData['Age'], bins=10)
```

```
plt.title('Distribution of Customer Age')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



In [395…
```
gender_cab=MergeData.groupby(['Company','Gender']).Customer_ID.nunique()

labs = gender_cab.index
vals = gender_cab.values
figp, axp = plt.subplots(figsize=(20,10))
axp.pie(vals , labels=labs, autopct='%1.1f%%',shadow=True, startangle=90,)
axp.axis('equal')

plt.title('Customer share per gender per cab')
plt.show()
```



In [ ]:
```
#MArket Share by company

pd.crosstab(index = MergeData.Year, columns = MergeData.Company, normalize = 'index').\
plot(kind = 'bar', stacked = True, rot = 0, title = 'Company Market Share', color = ['tab:pink', 'gold'],
    figsize = (10, 5), ylabel = 'Proportion').\
legend(loc = 'lower center', ncol = 2, bbox_to_anchor = (0.5, -0.3));
```

In [ ]:
```
```

In [ ]:
```
gender_cab=data.groupby(['Company','Gender']).Customer_ID.nunique()
gender_cab
Company     Gender
Pink Cab    Female    14819
```

```
            Male      17511
Yellow Cab  Female    18394
            Male      21502
Name: Customer_ID, dtype: int64
labs = gender_cab.index
vals = gender_cab.values
figp, axp = plt.subplots(figsize=(20,10))
axp.pie(vals , labels=labs, autopct='%1.1f%%',shadow=True, startangle=90,)
axp.axis('equal')

plt.title('Customer share per gender per cab')
plt.show()
```

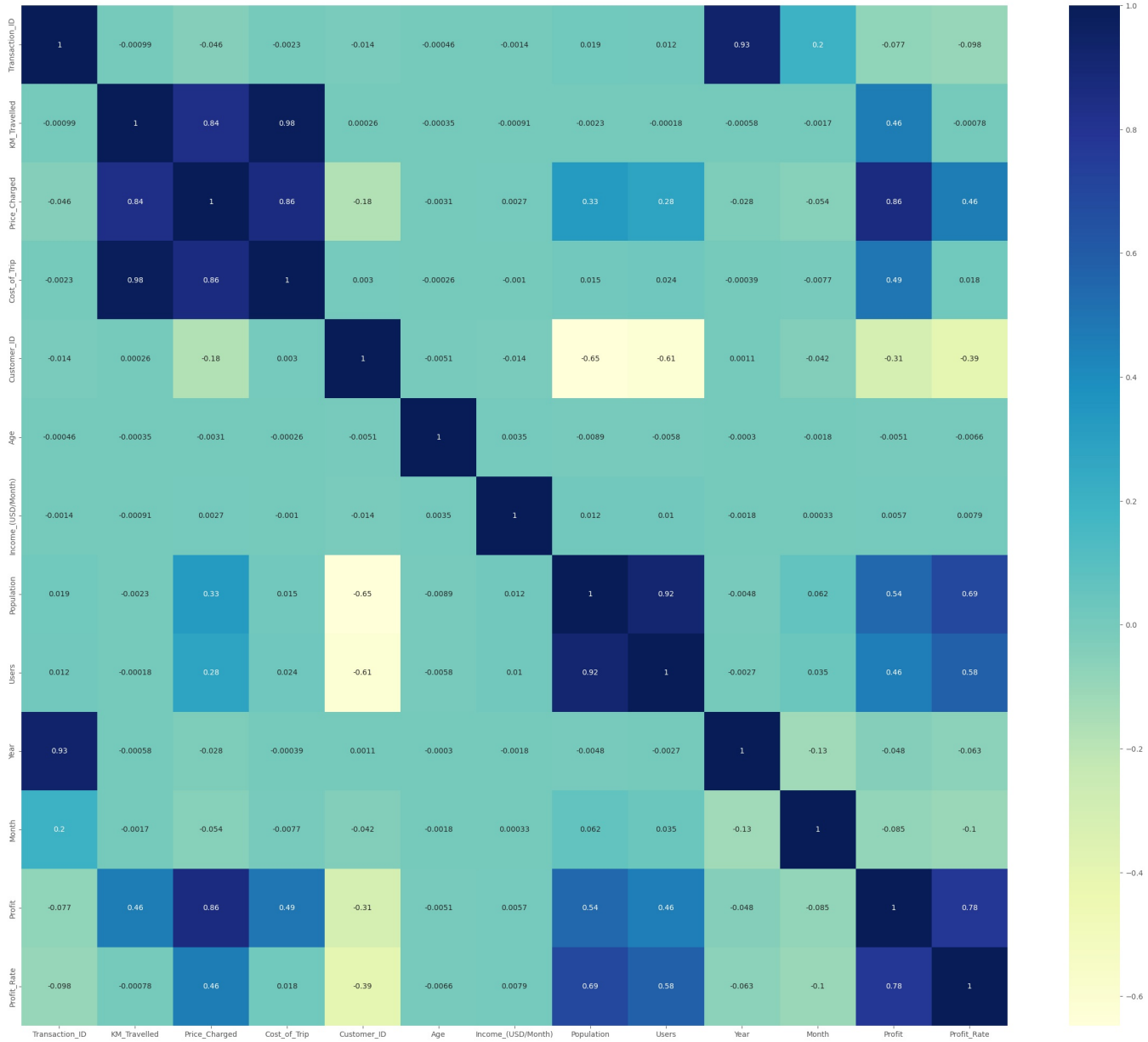
In [ ]:
```
# Group data by company and get fare amount

company_fares =MergeData.groupby('Company')['Price_Charged'].apply(list)

# Plot box plot
fig, ax = plt.subplots()
ax.boxplot(company_fares)
ax.set_xticklabels(company_fares.index)
ax.set_xlabel('Company')
ax.set_ylabel('Price_Charged')
plt.show()
```

In [ ]:
```
plt.figure(figsize=(20,8))



plt.subplot(1,2,1)
plt.title('Age vs Company')
sns.boxplot(x=MergeData.Company, y=MergeData.Age, palette=("cubehelix"))

plt.show()
```

In [ ]:
```
corr_matrix = MergeData.corr()

# print the correlation coefficients
print(corr_matrix)
```

In [396…
```
#Positive correlation coefficients indicate that the variables tend to increase or decrease together, while neg
```

In [397…
```
plt.figure(figsize = (30, 25))
sns.heatmap(corr_matrix, annot = True, cmap="YlGnBu")
plt.show()
```



In [ ]:
```
#Highly correlated variables to price_charged are - KM_Travelled,costof trip,profit
```

In [398…
```
#
#MergeData['Day of Week'] = pd.to_datetime(MergeData['Date']).dt.dayofweek

#We can then select the predictor variables and target variable:

#X = MergeData[['KM_Travelled', 'Price_Charged', 'Day of Week']]
#y = MergeData['Cost_of_Trip']

#We can split the data into training and testing sets:

#from sklearn.model_selection import train_test_split

#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


#Finally, we can fit a linear regression model and make predictions on the test set:

#from sklearn.linear_model import LinearRegression
#from sklearn.metrics import mean_squared_error

#lr = LinearRegression()
#lr.fit(X_train, y_train)

#y_pred = lr.predict(X_test)
#mse = mean_squared_error(y_test, y_pred)
```

In [408…
```
MergeData.isnull().sum() # Check for missing values
MergeData.describe() # Check for outliers
#Perform EDA to understand the relationships between the variables:
sns.pairplot(MergeData) # Scatter plots to understand relationships
sns.heatmap(MergeData.corr(), cmap='coolwarm', annot=True) # Correlation heatmap
#Select the variables that are most strongly correlated with the target variable (fare_amount), and create a ne

selected_cols = ['City', 'KM_Travelled ','Cost_of_Trip ','Gender', 'Age', 'Users','Price_Charged']
new_data = MergeData[selected_cols]
```

```
#Split the data into training and testing sets:

from sklearn.model_selection import train_test_split

X = new_data.drop('Price_Charged', axis=1)
y = new_data['Price_Charged']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
#Fit a linear regression model on the training data:

from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_train, y_train)
#Evaluate the model's performance on the testing data:

y_pred = lr.predict(X_test)

from sklearn.metrics import mean_squared_error, r2_score

print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
print('R-squared Score:', r2_score(y_test, y_pred))
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/var/folders/g1/734tgsrx39q4gpdsgyvfnp840000gn/T/ipykernel_19184/2642483227.py in <module>
      7
      8 selected_cols = ['City', 'KM_Travelled ', 'Cost_of_Trip ', 'Gender', 'Age', 'Users','Price_Charged']
----> 9 new_data = MergeData[selected_cols]
     10 #Split the data into training and testing sets:
     11

~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/frame.py in __getitem__(self, key)
   3509             if is_iterator(key):
   3510                 key = list(key)
-> 3511             indexer = self.columns._get_indexer_strict(key, "columns")[1]
   3512
   3513         # take() does not accept boolean indexers

~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/indexes/base.py in _get_indexer_strict(self, key, axis_
name)
   5794             keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
   5795
-> 5796         self._raise_if_missing(keyarr, indexer, axis_name)
   5797
   5798         keyarr = self.take(indexer)

~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/indexes/base.py in _raise_if_missing(self, key, indexer
, axis_name)
   5857
   5858             not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
-> 5859             raise KeyError(f"{not_found} not in index")
   5860
   5861     @overload

KeyError: "['KM_Travelled ', 'Cost_of_Trip '] not in index"
```


Transaction_ID
KM_Travelled
Price_Charged
Cost_of_Trip
Customer_ID
Age
Income_(USD/Month)
Population
Users
Year
Month
Profit
Profit_Rate
Day of Week

In [ ]: