Project Overview

This analysis aims to provide actionable insights to XYZ for their cab industry investment decision by evaluating the performance of two cab companies using EDA.

```
import pandas as pd

# Upload and load datasets
cab_data = pd.read_csv('/content/Cab_Data.csv')
city_data = pd.read_csv('/content/City.csv')
customer_data = pd.read_csv('/content/Customer_ID.csv')
transaction_data = pd.read_csv('/content/Transaction_ID.csv')
```

```
# Display the first few rows of each dataset
print("Cab Data:")
print(cab_data.head())

print("\nCity Data:")
print(city_data.head())

print("\nCustomer Data:")
print(customer_data.head())

print("\nTransaction Data:")
print(transaction_data.head())
```

```
Cab Data:
   Transaction ID  Date of Travel   Company        City  KM Travelled  \
0        10000011           42377  Pink Cab  ATLANTA GA         30.45
1        10000012           42375  Pink Cab  ATLANTA GA         28.62
2        10000013           42371  Pink Cab  ATLANTA GA          9.04
3        10000014           42376  Pink Cab  ATLANTA GA         33.17
4        10000015           42372  Pink Cab  ATLANTA GA          8.73

   Price Charged  Cost of Trip
0         370.95       313.635
1         358.52       334.854
2         125.20        97.632
3         377.40       351.602
4         114.62        97.776

City Data:
             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

Customer Data:
   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

Transaction Data:
   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
```

```
# Check for missing values
print("Missing Values in Cab Data:\n", cab_data.isnull().sum())
print("Missing Values in City Data:\n", city_data.isnull().sum())
print("Missing Values in Customer Data:\n", customer_data.isnull().sum())
print("Missing Values in Transaction Data:\n", transaction_data.isnull().sum())

# Convert 'Date of Travel' to datetime in Cab_Data
cab_data['Date of Travel'] = pd.to_datetime(cab_data['Date of Travel'], origin='1899-12-30', unit='D')
```

```
# Standardize city names
cab_data['City'] = cab_data['City'].str.strip()
city_data['City'] = city_data['City'].str.strip()

# Check for duplicates
print("Duplicates in Cab Data:", cab_data.duplicated().sum())
print("Duplicates in City Data:", city_data.duplicated().sum())
print("Duplicates in Customer Data:", customer_data.duplicated().sum())
print("Duplicates in Transaction Data:", transaction_data.duplicated().sum())
```

```
Missing Values in Cab Data:
 Transaction ID    0
Date of Travel    0
Company           0
City              0
KM Travelled      0
Price Charged     0
Cost of Trip      0
dtype: int64
Missing Values in City Data:
 City          0
Population    0
Users         0
dtype: int64
Missing Values in Customer Data:
 Customer ID           0
Gender                0
Age                   0
Income (USD/Month)    0
dtype: int64
Missing Values in Transaction Data:
 Transaction ID    0
Customer ID       0
Payment_Mode      0
dtype: int64
Duplicates in Cab Data: 0
Duplicates in City Data: 0
Duplicates in Customer Data: 0
Duplicates in Transaction Data: 0
```

```
# Merge Cab_Data with Transaction_ID
merged_data = pd.merge(cab_data, transaction_data, on='Transaction ID')

# Merge with Customer_ID
master_data = pd.merge(merged_data, customer_data, on='Customer ID')

# Merge with City
master_data = pd.merge(master_data, city_data, on='City')

# Display the structure of the final master dataset
print("Master Dataset:")
print(master_data.info())
```

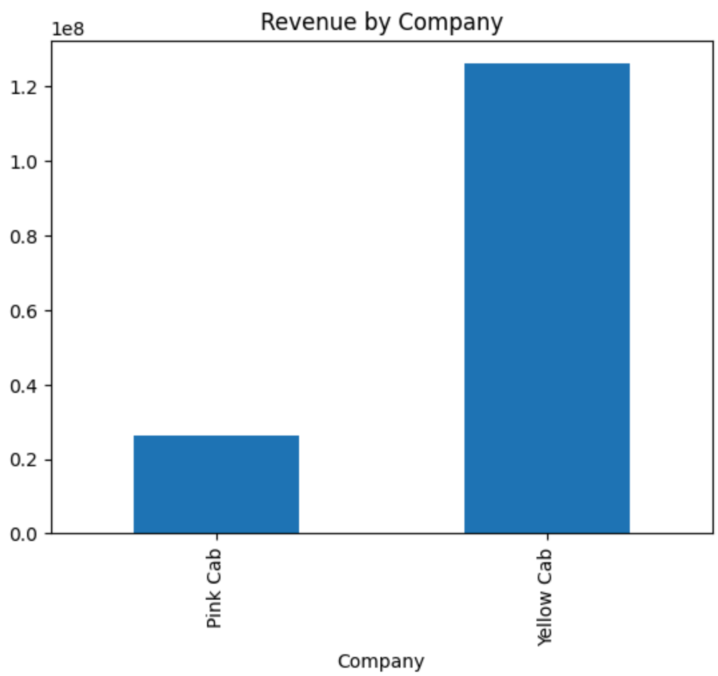
```
Master Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 359392 entries, 0 to 359391
Data columns (total 14 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   Transaction ID      359392 non-null  int64
 1   Date of Travel      359392 non-null  datetime64[ns]
 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   Payment_Mode        359392 non-null  object
 9   Gender              359392 non-null  object
 10  Age                 359392 non-null  int64
 11  Income (USD/Month)  359392 non-null  int64
 12  Population          359392 non-null  object
 13  Users               359392 non-null  object
dtypes: datetime64[ns](1), float64(3), int64(4), object(6)
memory usage: 38.4+ MB
None
```

Revenue and Margin Analysis by Company

Understanding which company generates more revenue and better margins is critical for investment decisions.

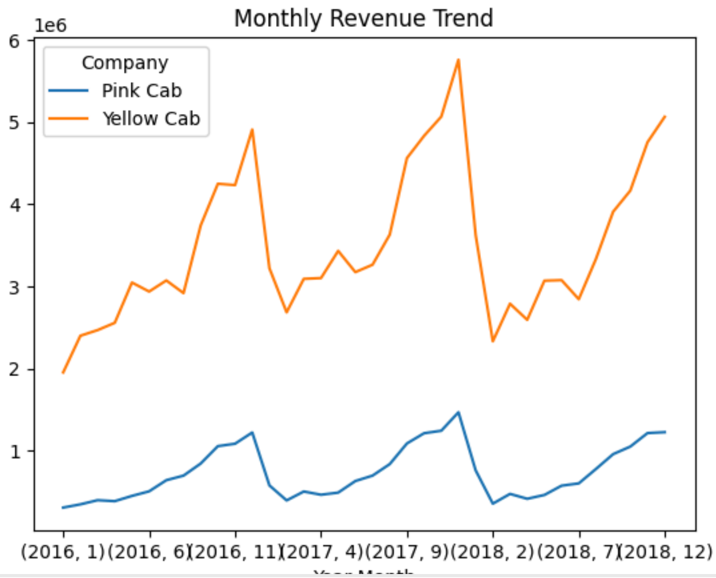
```
# Summary statistics
print(master_data.describe())

# Distribution of revenue by company
master_data.groupby('Company')['Price Charged'].sum().plot(kind='bar', title="Revenue by Company")
```

```
       Transaction ID                 Date of Travel   KM Travelled  \
count    3.593920e+05                         359392  359392.000000
mean     1.022076e+07  2017-08-17 01:37:55.042293760      22.567254
min      1.000001e+07            2016-01-02 00:00:00       1.900000
25%      1.011081e+07            2016-11-23 00:00:00      12.000000
50%      1.022104e+07            2017-09-10 00:00:00      22.440000
75%      1.033094e+07            2018-05-12 00:00:00      32.960000
max      1.044011e+07            2018-12-31 00:00:00      48.000000
std      1.268058e+05                            NaN      12.233526

       Price Charged   Cost of Trip    Customer ID            Age  \
count  359392.000000  359392.000000  359392.000000  359392.000000
mean      423.443311     286.190113   19191.652115      35.336705
min        15.600000      19.000000       1.000000      18.000000
25%       206.437500     151.200000    2705.000000      25.000000
50%       386.360000     282.480000    7459.000000      33.000000
75%       583.660000     413.683200   36078.000000      42.000000
max      2048.030000     691.200000   60000.000000      65.000000
std       274.378911     157.993661   21012.412463      12.594234

       Income (USD/Month)
count       359392.000000
mean         15048.822937
min           2000.000000
25%           8424.000000
50%          14685.000000
75%          21035.000000
max          35000.000000
std           7969.409482
<Axes: title={'center': 'Revenue by Company'}, xlabel='Company'>
```

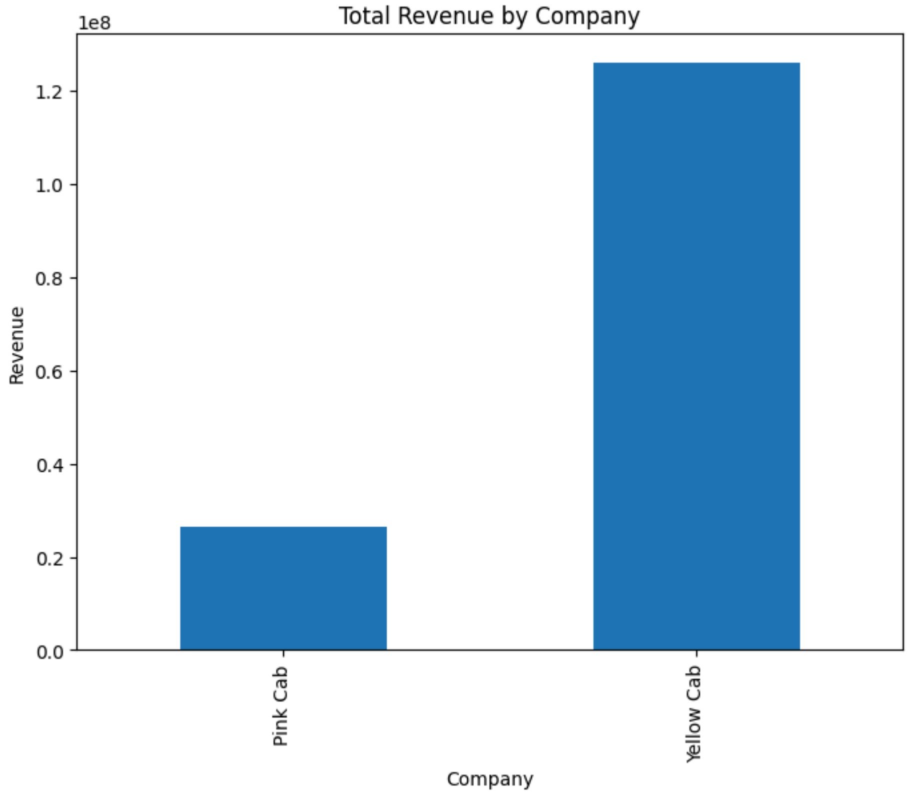


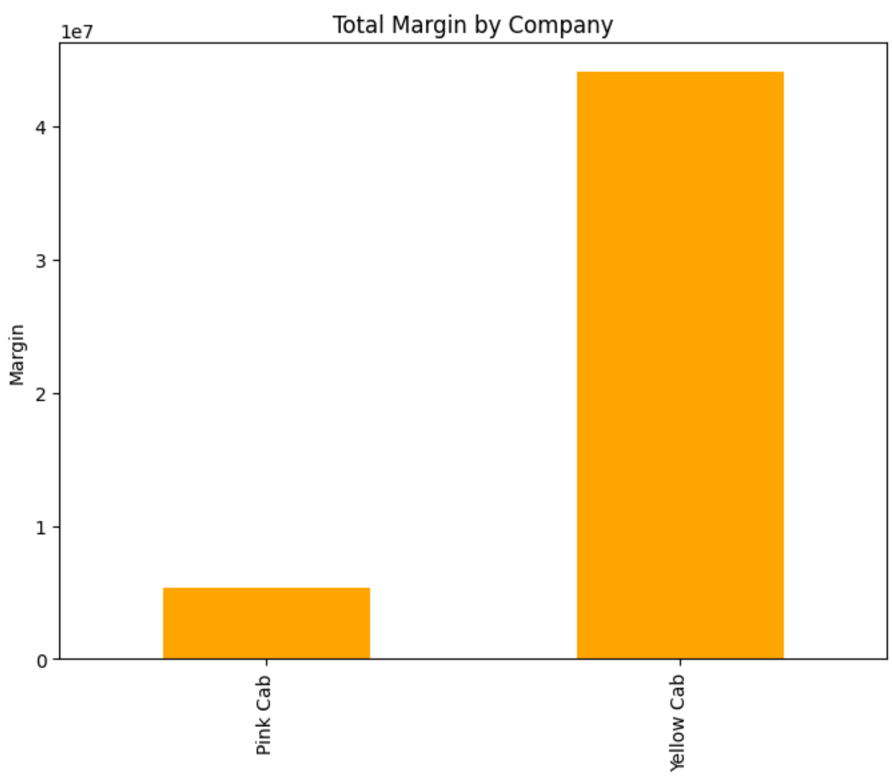

```
import matplotlib.pyplot as plt
import seaborn as sns

# Extract year and month
master_data['Year'] = master_data['Date of Travel'].dt.year
master_data['Month'] = master_data['Date of Travel'].dt.month
```

```
# Monthly revenue trend
monthly_revenue = master_data.groupby(['Company', 'Year', 'Month'])['Price Charged'].sum().unstack(level=0)
monthly_revenue.plot(kind='line', title="Monthly Revenue Trend")
plt.show()
```



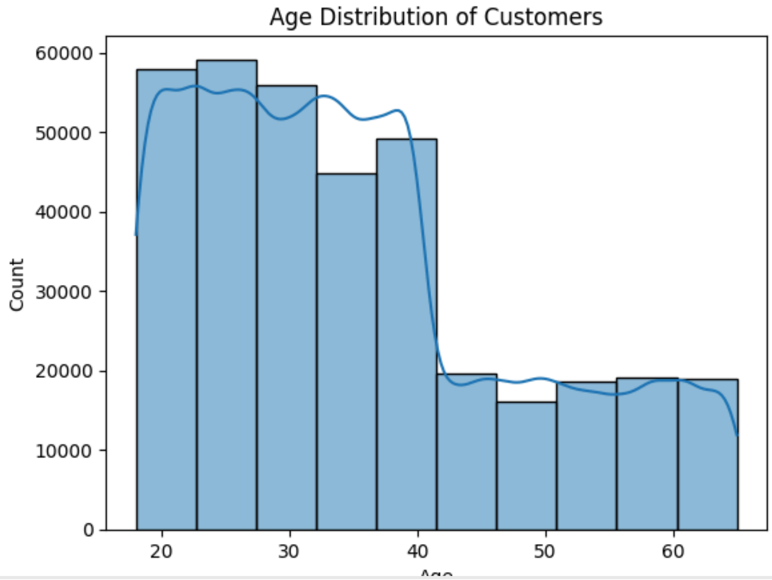

```
# Total revenue by company
revenue_by_company = master_data.groupby('Company')['Price Charged'].sum()
revenue_by_company.plot(kind='bar', title="Total Revenue by Company", figsize=(8, 6))
plt.ylabel("Revenue")
plt.show()

# Total margin by company
master_data['Margin'] = master_data['Price Charged'] - master_data['Cost of Trip']
margin_by_company = master_data.groupby('Company')['Margin'].sum()
margin_by_company.plot(kind='bar', title="Total Margin by Company", color="orange", figsize=(8, 6))
plt.ylabel("Margin")
plt.show()
```






```
# Average income by company
avg_income = master_data.groupby('Company')['Income (USD/Month)'].mean()
print("Average Income by Company:\n", avg_income)

# Age distribution
sns.histplot(master_data['Age'], bins=10, kde=True)
plt.title("Age Distribution of Customers")
plt.show()
```

```
Average Income by Company:
 Company
Pink Cab      15059.047137
Yellow Cab    15045.669817
Name: Income (USD/Month), dtype: float64
```



City-Wise Performance Analysis

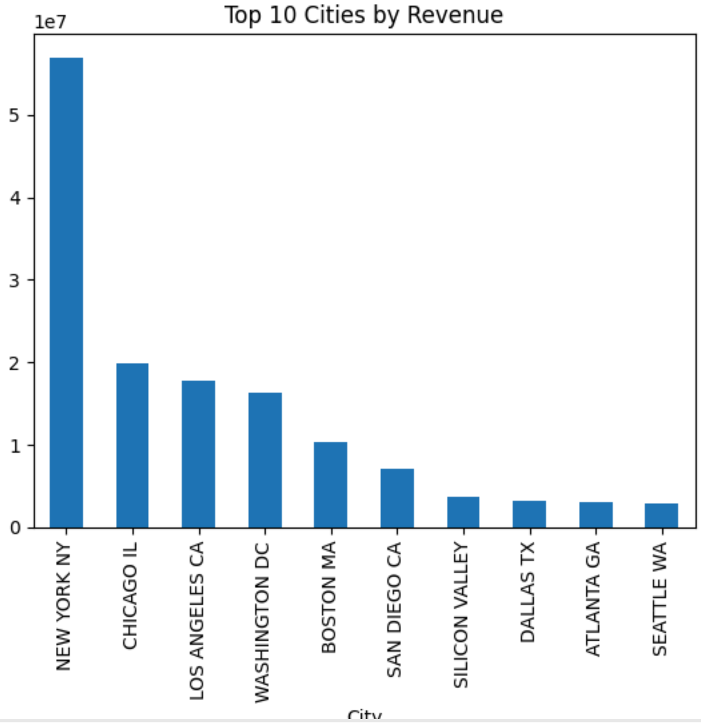
Analyze city-wise revenue and the number of users to identify high-performing regions.

Insights from Seasonality in Revenue

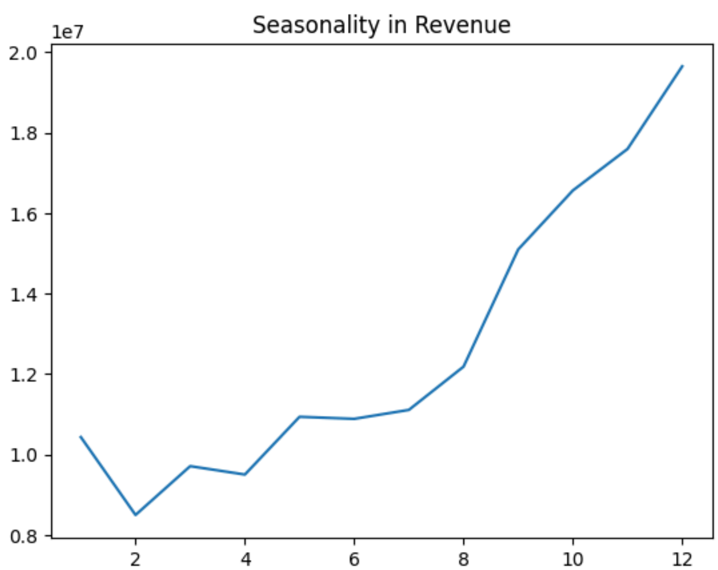
1. **Revenue peaks in December**: This suggests a strong seasonal trend, likely influenced by holidays and winter demand.
2. **Lower revenue in early months (January, February)**: Possible reduced demand post-holiday season.
3. **Recommendation**:
   - Focus marketing efforts during peak months (November-December).
   - Offer promotions or discounts in low-demand months to attract more customers.

```
# City-wise revenue
city_revenue = master_data.groupby('City')['Price Charged'].sum().sort_values(ascending=False)
print("Top Cities by Revenue:\n", city_revenue.head())

# Visualize top cities by revenue
city_revenue.head(10).plot(kind='bar', title="Top 10 Cities by Revenue")
plt.show()
```

```
Top Cities by Revenue:
 City
NEW YORK NY       56954061.67
CHICAGO IL        19841318.52
LOS ANGELES CA    17795624.41
WASHINGTON DC     16366703.83
BOSTON MA         10359755.42
Name: Price Charged, dtype: float64
```



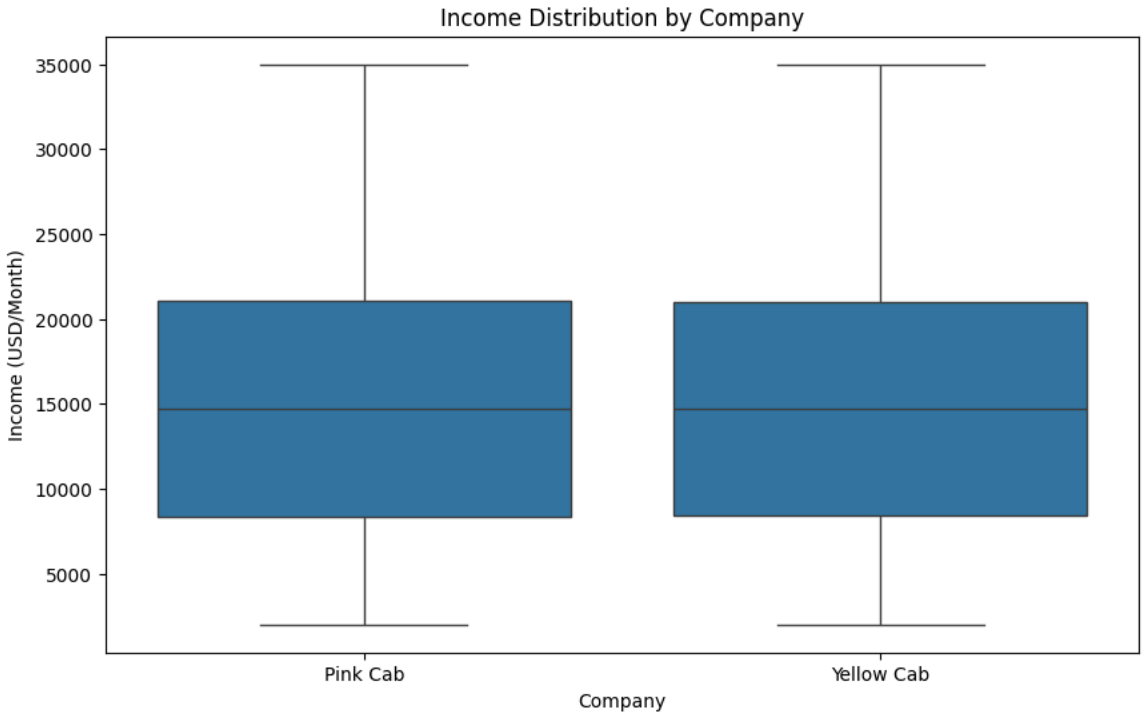

```
# Check for seasonality
seasonality = master_data.groupby(master_data['Date of Travel'].dt.month)['Price Charged'].sum()
seasonality.plot(kind='line', title="Seasonality in Revenue")
plt.show()
```

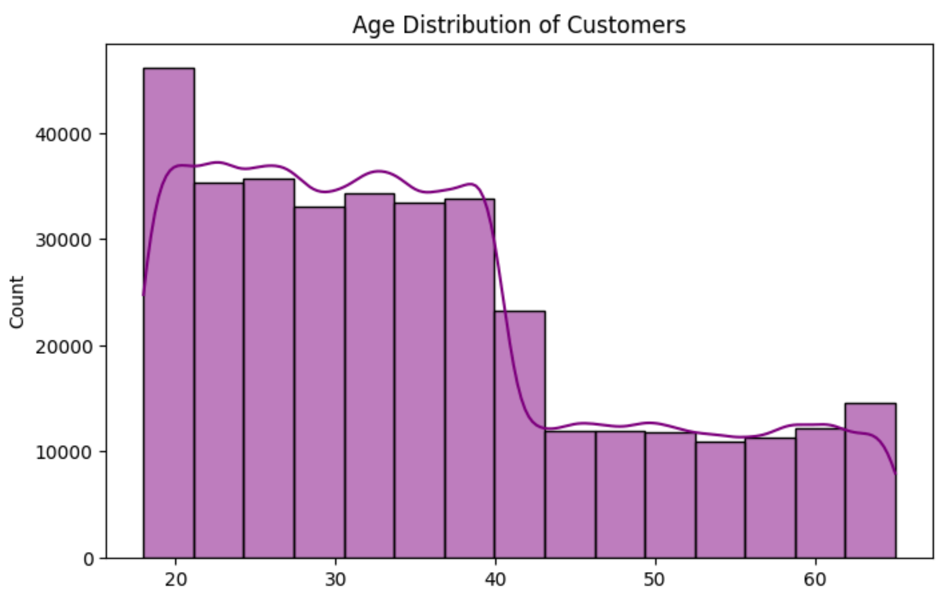


Customer Demographics Analysis

Analyze customer income and age to identify the most valuable customer segments.

```
import seaborn as sns

# Income distribution by company
plt.figure(figsize=(10, 6))
sns.boxplot(x='Company', y='Income (USD/Month)', data=master_data)
plt.title("Income Distribution by Company")
plt.show()

# Age distribution
plt.figure(figsize=(8, 5))
sns.histplot(master_data['Age'], bins=15, kde=True, color='purple')
plt.title("Age Distribution of Customers")
plt.xlabel("Age")
plt.show()
```





```
# Check the structure of master_data
print(master_data.info())

# Display the first few rows to confirm the column names
print(master_data.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 359392 entries, 0 to 359391
Data columns (total 17 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   Transaction ID      359392 non-null  int64
 1   Date of Travel      359392 non-null  datetime64[ns]
 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   Payment_Mode        359392 non-null  object
 9   Gender              359392 non-null  object
 10  Age                 359392 non-null  int64
 11  Income (USD/Month)  359392 non-null  int64
 12  Population          359392 non-null  object
 13  Users               359392 non-null  object
 14  Year                359392 non-null  int32
 15  Month               359392 non-null  int32
 16  Margin              359392 non-null  float64
dtypes: datetime64[ns](1), float64(4), int32(2), int64(4), object(6)
memory usage: 43.9+ MB
None
   Transaction ID Date of Travel   Company        City  KM Travelled  \
0        10000011     2016-01-08  Pink Cab  ATLANTA GA         30.45
1        10000012     2016-01-06  Pink Cab  ATLANTA GA         28.62
2        10000013     2016-01-02  Pink Cab  ATLANTA GA          9.04
3        10000014     2016-01-07  Pink Cab  ATLANTA GA         33.17
4        10000015     2016-01-03  Pink Cab  ATLANTA GA          8.73

   Price Charged  Cost of Trip  Customer ID Payment_Mode Gender  Age  \
0         370.95       313.635        29290         Card   Male   28
1         358.52       334.854        27703         Card   Male   27
2         125.20        97.632        28712         Cash   Male   53
3         377.40       351.602        28020         Cash   Male   23
4         114.62        97.776        27182         Card   Male   33

   Income (USD/Month) Population     Users  Year  Month  Margin
0               10813   814,885    24,701   2016      1  57.315
1                9237   814,885    24,701   2016      1  23.666
2               11242   814,885    24,701   2016      1  27.568
3               23327   814,885    24,701   2016      1  25.798
4                8536   814,885    24,701   2016      1  16.844
```

```
# Inspect the Population and Price Charged columns
print(master_data[['City', 'Population', 'Price Charged']].head())

# Check for non-numeric values in Population
print(master_data['Population'].unique())

# Convert Population to numeric again and handle errors
master_data['Population'] = pd.to_numeric(master_data['Population'], errors='coerce')

# Drop rows where Population or Price Charged is missing
master_data = master_data.dropna(subset=['Population', 'Price Charged'])

# Verify the cleaned data
print(master_data[['City', 'Population', 'Price Charged']].head())
```

```
Empty DataFrame
Columns: [City, Population, Price Charged]
Index: []
[]
Empty DataFrame
Columns: [City, Population, Price Charged]
Index: []
```

```
# Check unique values in Population and Price Charged before cleaning
print("Unique Population Values:")
print(city_data['Population'].unique())
```

```
print("\nUnique Price Charged Values:")
print(cab_data['Price Charged'].unique())

# Display first rows of city_data and cab_data
print("\nCity Data:")
print(city_data.head())

print("\nCab Data:")
print(cab_data.head())
```

```
Unique Population Values:
[' 8,405,837 ' ' 1,955,130 ' ' 1,595,037 ' ' 1,339,155 ' ' 1,177,609 '
 ' 1,030,185 ' ' 959,307 ' ' 943,999 ' ' 942,908 ' ' 814,885 ' ' 754,233 '
 ' 698,371 ' ' 671,238 ' ' 631,442 ' ' 629,591 ' ' 545,776 ' ' 542,085 '
 ' 418,859 ' ' 327,225 ' ' 248,968 ']

Unique Price Charged Values:
[370.95 358.52 125.2  ...  31.49 742.24 620.62]

City Data:
             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

Cab Data:
   Transaction ID Date of Travel   Company        City  KM Travelled  \
0        10000011     2016-01-08  Pink Cab  ATLANTA GA         30.45
1        10000012     2016-01-06  Pink Cab  ATLANTA GA         28.62
2        10000013     2016-01-02  Pink Cab  ATLANTA GA          9.04
3        10000014     2016-01-07  Pink Cab  ATLANTA GA         33.17
4        10000015     2016-01-03  Pink Cab  ATLANTA GA          8.73

   Price Charged  Cost of Trip
0         370.95       313.635
1         358.52       334.854
2         125.20        97.632
3         377.40       351.602
4         114.62        97.776
```

```
# Clean 'Population' column in City data
city_data['Population'] = city_data['Population'].str.replace(',', '').astype(float)

# Clean 'Price Charged' in Cab data
cab_data['Price Charged'] = pd.to_numeric(cab_data['Price Charged'], errors='coerce')

# Verify cleaned columns
print(city_data.head())
print(cab_data.head())
```

```
             City  Population      Users
0     NEW YORK NY   8405837.0   302,149
1      CHICAGO IL   1955130.0   164,468
2  LOS ANGELES CA   1595037.0   144,132
3        MIAMI FL   1339155.0    17,675
4  SILICON VALLEY   1177609.0    27,247
   Transaction ID Date of Travel   Company        City  KM Travelled  \
0        10000011     2016-01-08  Pink Cab  ATLANTA GA         30.45
1        10000012     2016-01-06  Pink Cab  ATLANTA GA         28.62
2        10000013     2016-01-02  Pink Cab  ATLANTA GA          9.04
3        10000014     2016-01-07  Pink Cab  ATLANTA GA         33.17
4        10000015     2016-01-03  Pink Cab  ATLANTA GA          8.73

   Price Charged  Cost of Trip
0         370.95       313.635
1         358.52       334.854
2         125.20        97.632
3         377.40       351.602
4         114.62        97.776
```

```
# Merge Cab_Data with Transaction_ID
merged_data = pd.merge(cab_data, transaction_data, on='Transaction ID')

# Merge with Customer_ID
```

```
master_data = pd.merge(merged_data, customer_data, on='Customer ID')

# Merge with City data
master_data = pd.merge(master_data, city_data, on='City', how='left')  # Ensure left join for matching


# Group by City for Population and Revenue
city_revenue = master_data.groupby('City')[['Population', 'Price Charged']].sum().reset_index()

# Verify if the data is populated
print(city_revenue.head())

# Plot the data
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.scatterplot(x='Population', y='Price Charged', data=city_revenue)
plt.title("Population vs Revenue")
plt.xlabel("City Population")
plt.ylabel("Total Revenue")
plt.show()
```

```
         City    Population  Price Charged
0  ATLANTA GA  6.158086e+09     2980241.72
1   AUSTIN TX  3.419224e+09     1877142.50
2   BOSTON MA  7.392358e+09    10359755.42
3  CHICAGO IL  1.107092e+11    19841318.52
4   DALLAS TX  6.616385e+09     3142429.91
```