```
Bike-Share: Google Data Analytics Capstone Project
```

Introduction Welcome to the Cyclistic bike-share analysis case study! Scenario You are a junior data analyst working in the marketing analyst team at Cyclistic, a bike-share company in Chicago. The director of marketing believes the company's future success depends on maximizing the number of annual memberships. Therefore, your team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, your team will design a new marketing strategy to convert casual riders into annual members. But first, Cyclistic executives must approve your recommendations, so they must be backed up with compelling data insights and professional data visualizations.

About the company In 2016, Cyclistic launched a successful bike-share offering. Since then, the program has grown to a fleet of 5,824 bicycles that are geotracked and locked into a network of 692 stations across Chicago. The bikes can be unlocked from one station and returned to any other station in the system anytime. Until now, Cyclistic's marketing strategy relied on building general awareness and appealing to broad consumer segments. One approach that helped make these things possible was the flexibility of its pricing plans: single-ride passes, full-day passes, and annual memberships. Customers who purchase single-ride or full-day passes are referred to as casual riders. Customers who purchase annual memberships are Cyclistic members. Cyclistic's finance analysts have concluded that annual members are much more profitable than casual riders. Although the pricing flexibility helps Cyclistic attract more customers, Moreno believes that maximizing the number of annual members will be key to future growth. Rather than creating a marketing campaign that targets all-new customers, Moreno believes there is a very good chance to convert casual riders into members. She notes that casual riders are already aware of the Cyclistic program and have chosen Cyclistic for their mobility needs. Moreno has set a clear goal: Design marketing strategies aimed at converting casual riders into annual members. In order to do that, however, the marketing analyst team needs to better understand how annual members and casual riders differ, why casual riders would buy a membership, and how digital media could affect their marketing tactics. Moreno and her team are interested in analyzing the Cyclistic historical bike trip data to identify trends

Ask Three questions will guide the future marketing program: 1.How do annual members and casual riders use Cyclistic bikes differently? 2.Why would casual riders buy Cyclistic annual memberships? 3.How can Cyclistic use digital media to influence casual riders to become members?

Follow these steps: 1. Download the previous 12 months of Cyclistic trip data. 2. Unzip the files. 3. Create a folder on your desktop or Drive to house the files. Use appropriate file-naming conventions. 4. Create subfolders for the .CSV file and the .XLS or Sheets file so that you have a copy of the original data. Move the downloaded files to the appropriate subfolder. 5. Follow these instructions for either Excel (a) or Google Sheets (b): a. Launch Excel, open each file, and choose to Save As an Excel Workbook file. Put it in the subfolder you created for .XLS files. b. Open each .CSV file in Google Sheets and save it to the appropriate subfolder. 6. Open your spreadsheet and create a column called "ride_length." Calculate the length of each ride by subtracting the column "started_at" from the column "ended_at" (for example, =D2-C2) and format as HH:MM:SS using Format > Cells > Time > 37:30:55. 7. Create a column called "day_of_week," and calculate the day of the week that each ride started using the "WEEKDAY" command (for example, =WEEKDAY(C2,1)) in each file. Format as General or as a number with no decimals, noting that 1 = Sunday and 7 = Saturday. 8. Proceed to the analyze step.

We decided to use 12 months of data from January 2021 to December 2021. Lets start coding . we have 12 files and we would first merge them into one dataframe

Business Objective: To maximize the number of annual memberships by converting casual riders to annual members.

In [388]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
```

In [389]:

```
jan= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cours
feb= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cours
march= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cou
april= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cou
may= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cours
june= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cour
july= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cour
aug= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cours
sep= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cours
octo= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cour
nov= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cours
dec= pd.read_csv("C:/Users/nse/OneDrive - Högskolan Dalarna/Coursera/Google analytics cours
```

In [390]:

```
#now lets merge all the dataframes into one
data_frames = [jan, feb, march,april,may,june,july,aug,sep,octo,nov,dec]
```

In [391]:

```
data_df = pd.concat(data_frames)
```

In [392]:

```
data_df.head()
```

Out[392]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id
0	E19E6F1B8D4C42ED	electric_bike	2021-01-23 16:14:19	2021-01-23 16:24:44	California Ave & Cortez St	17660
1	DC88F20C2C55F27F	electric_bike	2021-01-27 18:43:08	2021-01-27 18:47:12	California Ave & Cortez St	17660
2	EC45C94683FE3F27	electric_bike	2021-01-21 22:35:54	2021-01-21 22:37:14	California Ave & Cortez St	17660
3	4FA453A75AE377DB	electric_bike	2021-01-07 13:31:13	2021-01-07 13:42:55	California Ave & Cortez St	17660
4	BE5E8EB4E7263A0B	electric_bike	2021-01-23 02:24:02	2021-01-23 02:24:45	California Ave & Cortez St	17660

In [393]:

```
data_df.dtypes
```

Out[393]:

```
ride_id                object
rideable_type          object
started_at             object
ended_at               object
start_station_name     object
start_station_id       object
end_station_name       object
end_station_id         object
start_lat             float64
start_lng             float64
end_lat               float64
end_lng               float64
member_casual          object
dtype: object
```

As we need to to find ride lengths, We would need to convert variables involved into from object to date and time

In [394]:

```
data_df["started_at"] = pd.to_datetime(data_df["started_at"])
data_df["ended_at"] = pd.to_datetime(data_df["ended_at"])
```

In [395]:

```
data_df.dtypes
```

Out[395]:

```
ride_id                       object
rideable_type                 object
started_at            datetime64[ns]
ended_at              datetime64[ns]
start_station_name            object
start_station_id              object
end_station_name              object
end_station_id                object
start_lat                    float64
start_lng                    float64
end_lat                      float64
end_lng                      float64
member_casual                 object
dtype: object
```

In [396]:

```
#data_df.drop(columns=['Started_at'], axis =1) #Alternative to specifying axis (labels, axi
```

In [397]:

```
#data_df.head(2)
```

In [398]:

```
# calculating the ride length
data_df["ride_length"]= data_df["ended_at"] - data_df["started_at"]
```

In [399]:

```
data_df["ride_length"]
```

Out[399]:

```
0         00:10:25
1         00:04:04
2         00:01:20
3         00:11:42
4         00:00:43
            ...
247535    00:19:13
247536    00:07:01
247537    00:08:17
247538    00:14:13
247539    00:03:37
Name: ride_length, Length: 5595063, dtype: timedelta64[ns]
```

In [400]:

```
data_df[['start_date','start_time']] = data_df['started_at'].astype(str).str.split(' ',n=1,
```

In [401]:

```
data_df.dtypes
```

Out[401]:

```
ride_id                          object
rideable_type                    object
started_at               datetime64[ns]
ended_at                 datetime64[ns]
start_station_name               object
start_station_id                 object
end_station_name                 object
end_station_id                   object
start_lat                       float64
start_lng                       float64
end_lat                         float64
end_lng                         float64
member_casual                    object
ride_length             timedelta64[ns]
start_date                       object
start_time                       object
dtype: object
```

In [402]:

```
data_df.head(5)
```

Out[402]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id
0	E19E6F1B8D4C42ED	electric_bike	2021-01-23 16:14:19	2021-01-23 16:24:44	California Ave & Cortez St	17660
1	DC88F20C2C55F27F	electric_bike	2021-01-27 18:43:08	2021-01-27 18:47:12	California Ave & Cortez St	17660
2	EC45C94683FE3F27	electric_bike	2021-01-21 22:35:54	2021-01-21 22:37:14	California Ave & Cortez St	17660
3	4FA453A75AE377DB	electric_bike	2021-01-07 13:31:13	2021-01-07 13:42:55	California Ave & Cortez St	17660
4	BE5E8EB4E7263A0B	electric_bike	2021-01-23 02:24:02	2021-01-23 02:24:45	California Ave & Cortez St	17660

In [403]:

```
data_df["start_time"] = pd.to_datetime(data_df["start_time"])
data_df["start_date"] = pd.to_datetime(data_df["start_date"])
```

In [404]:

```
data_df.dtypes
```

Out[404]:

```
ride_id                          object
rideable_type                    object
started_at               datetime64[ns]
ended_at                 datetime64[ns]
start_station_name               object
start_station_id                 object
end_station_name                 object
end_station_id                   object
start_lat                       float64
start_lng                       float64
end_lat                         float64
end_lng                         float64
member_casual                    object
ride_length             timedelta64[ns]
start_date               datetime64[ns]
start_time               datetime64[ns]
dtype: object
```

In [405]:

```
data_df["week day"]=data_df["start_date"].apply(lambda x:x.weekday())# takes a column and a
data_df["week day"].unique() # show only unique values
#lambda function is a simple, short, throwaway function which is designed to be created inl
```

Out[405]:

```
array([5, 2, 3, 0, 6, 4, 1], dtype=int64)
```

In [406]:

```
data_df["week day"]
```

Out[406]:

```
0          5
1          2
2          3
3          3
4          5
          ..
247535     6
247536     0
247537     3
247538     0
247539     0
Name: week day, Length: 5595063, dtype: int64
```

In [407]:

```
day_dict={0:"Sunday", 1:"Monday",2:"Tuesday",3:"Wednesday",4:"Thursday",5:"Friday",6:"satur
```

In [408]:

```
data_df["weekday_name"]= data_df['week day'].apply(lambda y:day_dict[y])
```

In [409]:

```
data_df['year'] = pd.DatetimeIndex(data_df['start_date']).year
data_df['month']= pd.DatetimeIndex(data_df['start_date']).month
data_df.head()
```

Out[409]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id
0	E19E6F1B8D4C42ED	electric_bike	2021-01-23 16:14:19	2021-01-23 16:24:44	California Ave & Cortez St	17660
1	DC88F20C2C55F27F	electric_bike	2021-01-27 18:43:08	2021-01-27 18:47:12	California Ave & Cortez St	17660
2	EC45C94683FE3F27	electric_bike	2021-01-21 22:35:54	2021-01-21 22:37:14	California Ave & Cortez St	17660
3	4FA453A75AE377DB	electric_bike	2021-01-07 13:31:13	2021-01-07 13:42:55	California Ave & Cortez St	17660
4	BE5E8EB4E7263A0B	electric_bike	2021-01-23 02:24:02	2021-01-23 02:24:45	California Ave & Cortez St	17660

In [410]:

```
#sort the dataFrame by start date in ascending order
data_df.sort_values(by=['start_date'],inplace= True, ascending=True )
```

We will now look at the missing values now

In [411]:

```
data_df.isnull().sum()
```

Out[411]:

```
ride_id                    0
rideable_type              0
started_at                 0
ended_at                   0
start_station_name    690809
start_station_id      690806
end_station_name      739170
end_station_id        739170
start_lat                  0
start_lng                  0
end_lat                 4771
end_lng                 4771
member_casual              0
ride_length                0
start_date                 0
start_time                 0
week day                   0
weekday_name               0
year                       0
month                      0
dtype: int64
```

As we have no null values in the desired columns sych as start time as date etc there is no need to drop any of the rows.For our analysis the data seems reasonably in good shape. But in case we had to find some information regarding position of these rides start and end position then we would have had to think about the startegies for handling missing data.

In [412]:

```
data_df.duplicated().any()
```

Out[412]:

```
False
```

we see no duplicated rows of data

Analysis lets see the mean ride length of both groups of users such as member and casual

In [413]:

```
data_member = data_df[data_df["member_casual"]=="member"]
data_member_ride_len_mean = data_member["ride_length"].mean()
data_member_ride_len_mean
```

Out[413]:

```
Timedelta('0 days 00:13:37.970452')
```

In [414]:

```
data_casual = data_df[data_df["member_casual"]=="casual"]
data_casual_ride_len_mean = data_casual["ride_length"].mean()
data_casual_ride_len_mean
```

Out[414]:

```
Timedelta('0 days 00:32:00.056830')
```

In [415]:

```

data_member_ride_len_max = data_member["ride_length"].max()
data_member_ride_len_max
```

Out[415]:

```
Timedelta('1 days 01:59:56')
```

In [416]:

```
data_casual_ride_len_max = data_casual["ride_length"].max()
data_casual_ride_len_max
```

Out[416]:

```
Timedelta('38 days 20:24:09')
```

we can observe that the mean and max ride length of casual riders is higher there for it is worth for further analysis. lets take a look at max length of ride for each group of riders

In [417]:

```
# calculate the mode of week day for memember which means to see what day do they usually r
data_member_ride_day_mode = data_member["week day"].mode()
data_member_ride_day_mode
# 2 is tuesday
```

Out[417]:

```
0    2
dtype: int64
```

In [418]:

```
data_member_ride_day_mode = data_casual["week day"].mode()
data_member_ride_day_mode
# 5 is friday
```

Out[418]:

```
0    5
dtype: int64
```

In [419]:

```
mem_rides_pday= data_df[data_df["member_casual"]=="member"].groupby('week day')['ride_id'].
mem_rides_pday
```

Out[419]:

```
week day
0    416212
1    465513
2    477192
3    451524
4    446428
5    433047
6    376142
Name: ride_id, dtype: int64
```

In [420]:

```
cas_rides_pday = data_df[data_df["member_casual"]=="casual"].groupby('week day')['ride_id']
cas_rides_pday
```

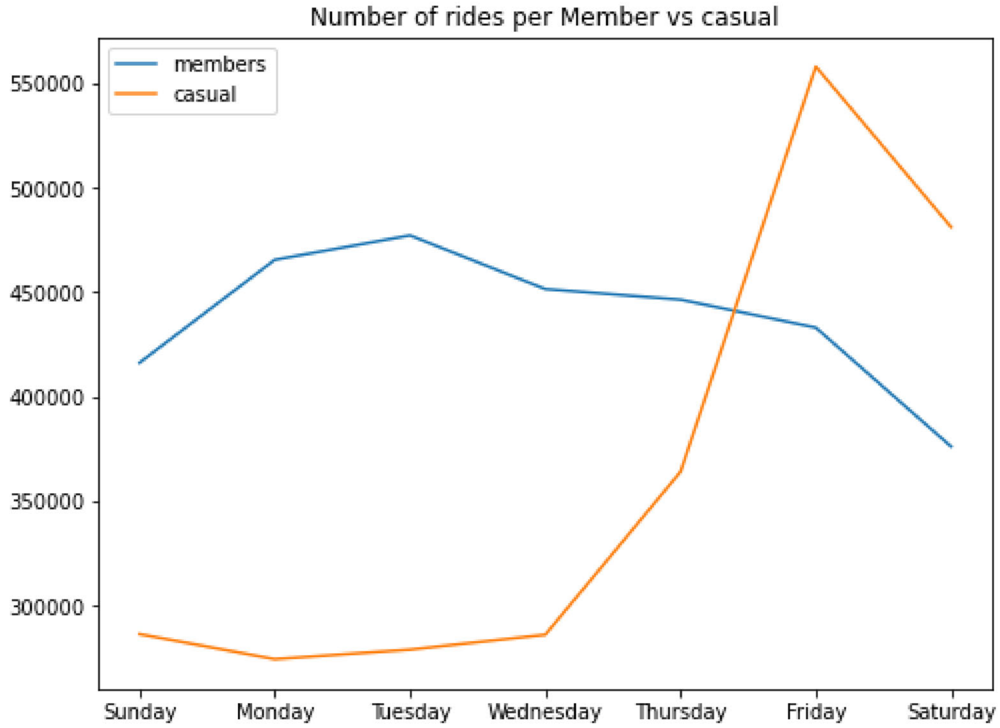
Out[420]:

```
week day
0    286376
1    274392
2    278950
3    286064
4    364080
5    558000
6    481143
Name: ride_id, dtype: int64
```

now lets plot rides per day of both member and compare the trend

In [421]:

```
plt.figure(figsize=(8,6))
plt.plot(mem_rides_pday.index, mem_rides_pday.values)
plt.plot(cas_rides_pday.index, cas_rides_pday.values)
plt.title("Number of rides per Member vs casual")
plt.legend(["members","casual"])
labels=["Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"]
plt.xticks(mem_rides_pday.index,labels)
plt.show()
```



From the above plot we can see that there is Rides of the members users are higher during the week days and those of Casual users seems that they use rides more often on the weekends. By these trends we see that the casual members

##Average

In [422]:

```
cas_rides_avg = data_df[data_df["member_casual"]=="casual"].groupby('week day')['ride_lengt
cas_rides_avg
```

Out[422]:

```
week day
0    00:31:52.506467
1    00:27:58.314185
2    00:27:39.426398
3    00:27:42.195477
4    00:30:20.890491
5    00:34:42.351462
6    00:37:33.650536
Name: ride_length, dtype: timedelta64[ns]
```

In [423]:

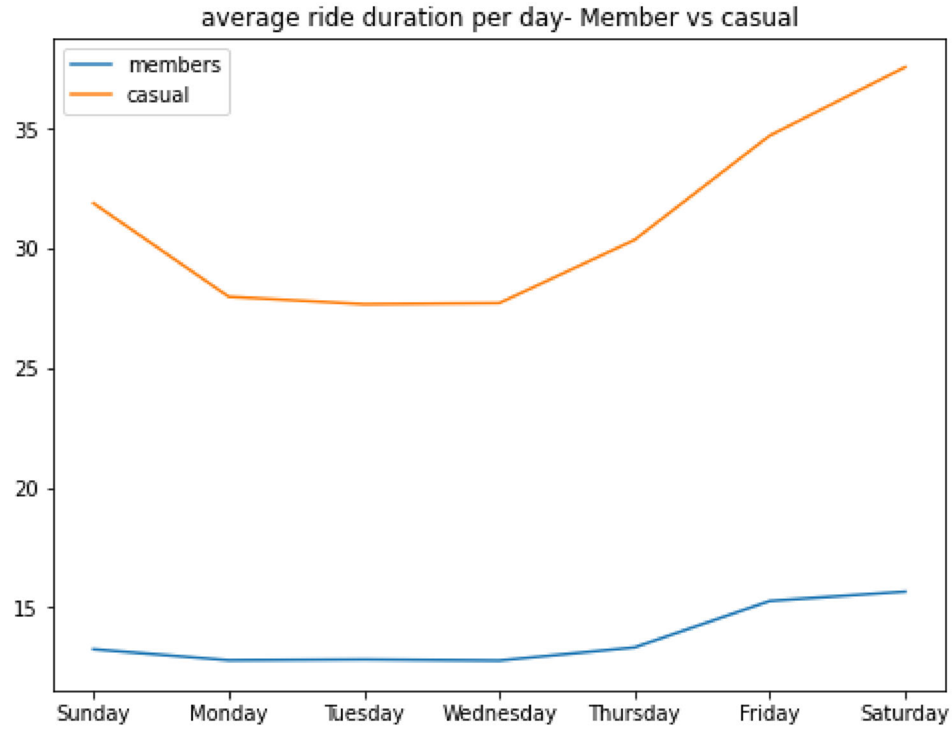
```
mem_rides_avg = data_df[data_df["member_casual"]=="member"].groupby('week day')['ride_lengt
mem_rides_avg
```

Out[423]:

```
week day
0    00:13:14.836198
1    00:12:47.280000
2    00:12:49.084446
3    00:12:46.564082
4    00:13:19.485428
5    00:15:15.861204
6    00:15:39.270318
Name: ride_length, dtype: timedelta64[ns]
```

In [431]:

```
plt.figure(figsize=(8,6))
plt.plot(mem_rides_avg/pd.Timedelta(minutes=1)) # gets average ride length in minutes
plt.plot(cas_rides_avg/pd.Timedelta(minutes=1))
plt.title("average ride duration per day- Member vs casual")
plt.legend(["members","casual"])
labels=["Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"]
plt.xticks(mem_rides_avg.index,labels)
plt.show()
```



In [425]:

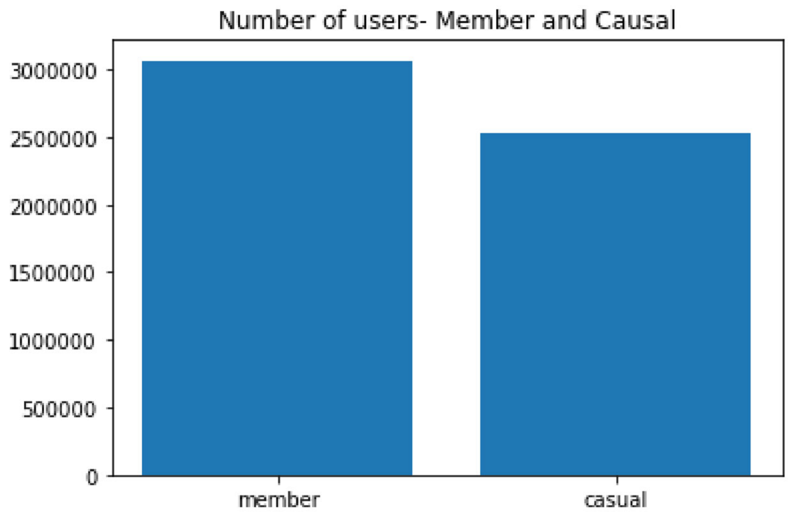
```
member_type =data_df["member_casual"].value_counts()
member_type
```

Out[425]:

```
member    3066058
casual    2529005
Name: member_casual, dtype: int64
```

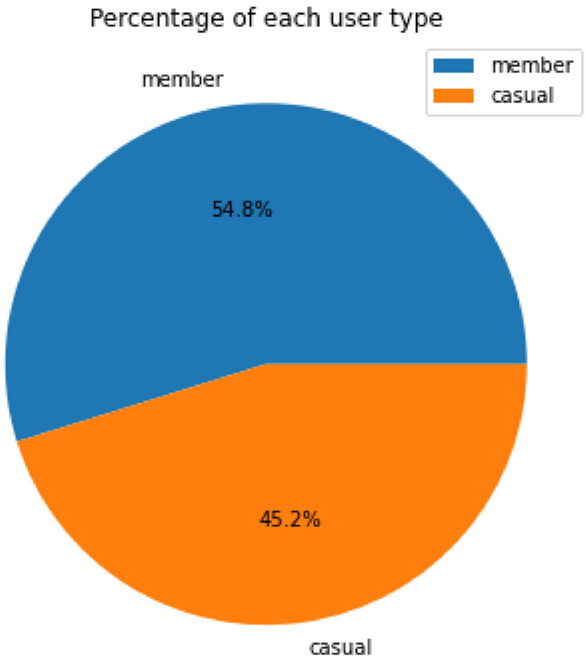
In [426]:

```
plt.title("Number of users- Member and Causal")
plt.bar(member_type.index, member_type.values)
plt.ticklabel_format(style='plain', axis='y')
```



In [427]:

```
# pie chart  to see the percentage
plt.figure(figsize=(8,6))
plt.pie(member_type.values, labels =member_type.index,autopct='%1.1f%%')
plt.title("Percentage of each user type")
plt.legend(member_type.index)
plt.show()
```



In [428]:

```
#to check yearly trends in the users
monthly_users = data_df.groupby("month") ['member_casual'].value_counts()
monthly_users
```

Out[428]:

```
month  member_casual
1      member            78717
       casual            18117
2      member            39491
       casual            10131
3      member           144463
       casual            84033
4      member           200629
       casual           136601
5      member           274717
       casual           256916
6      casual           370681
       member           358914
7      casual           442056
       member           380354
8      casual           412671
       member           391681
9      member           392257
       casual           363890
10     member           373984
       casual           257242
11     member           253049
       casual           106929
12     member           177802
       casual            69738
Name: member_casual, dtype: int64
```

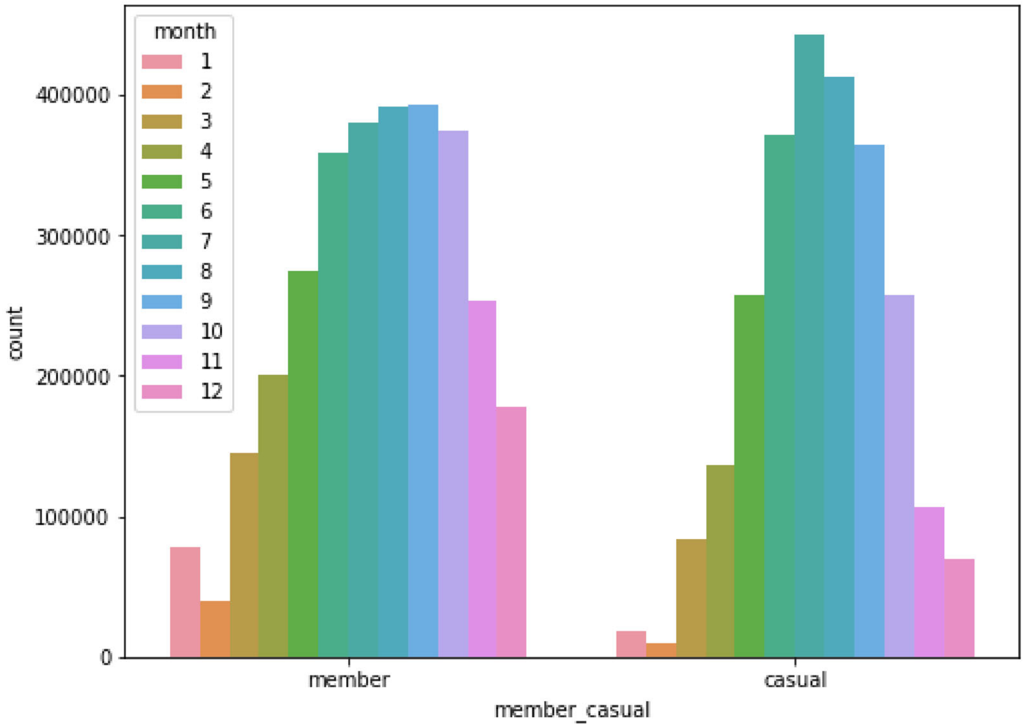
In [429]:

```
plt.figure(figsize=(8,6))

sns.countplot(x="member_casual",hue="month", data=data_df)
```

Out[429]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x251db72fa00>
```

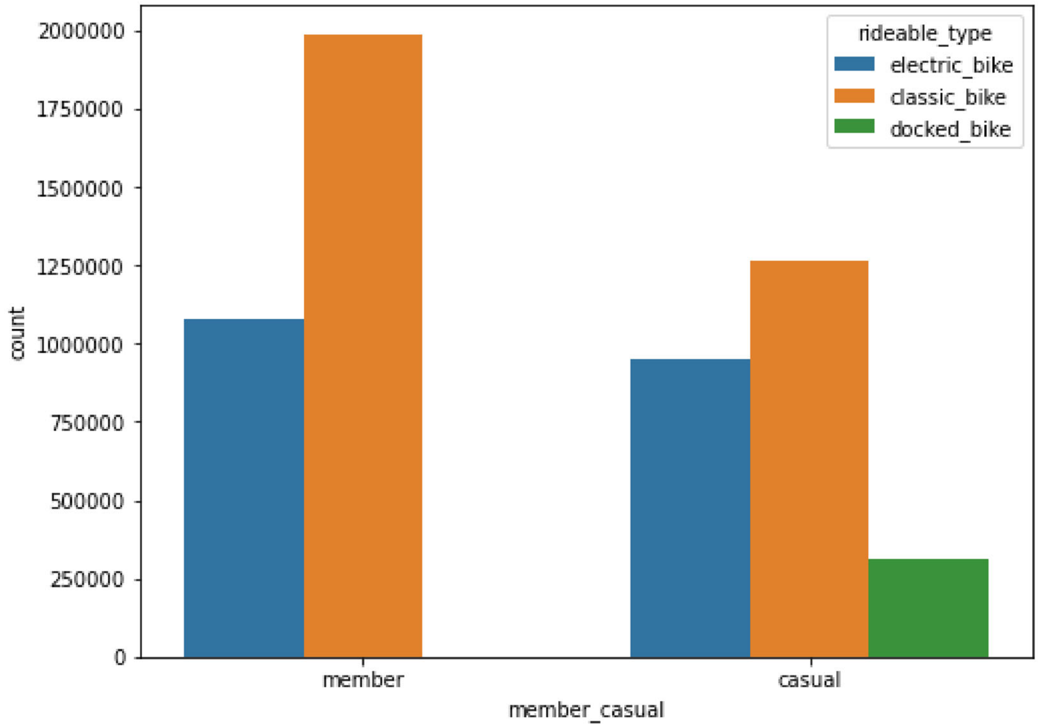


We see that number of rides are highest in both categories during summers a bit higher for casual users. one of the reason seems Vacations and good weather.

In [430]:

```
plt.figure(figsize=(8,6))

sns.countplot(x="member_casual", hue="rideable_type", data=data_df)
plt.ticklabel_format(style='plain', axis='y') # for removing scientific notation
```




```
Recommendations

We see that the Number of casual members are quite high, the company can give them some
incentive to become loyal costumers through becoming annual member .
1. The incentive could be that the company could lower the charges for the rides for
member customers than a casual users. This would encourage to casual users to become
regular members.
2. The bike company can also give bonus points for each ride to the member user. Bonus
points can be accumulated to secure free rides. Hopefully this would aim to see an
increase in number of rides and also member subscriptions.
3. We also see that casual users are more active on weekends so we lower weekend rates
for member users to encourage them to use rides on weekend too and this will also
encourage causal riders to become member subscribers.
4. Discounts can also be given e.g. may be after every 5 miles or 5% to 10% after 10
minutes of ride.
5. We see that the casual members also use Docker bikes so may be we can provide some
discounts on riding docker bikes for member users. so that it would encourage the causal
users to buy annual subscriptions.
6. Discounts can be offered during the months June-August to attract casual to buy
memberships.
```