Cyclistic Bike-Share is a fictional company where dataset is created only for study purpose. It is a final course of Google Data Analytic Career Certification offered by Coursera.As part of this, I completed this case study. Below are Case study question and my fingings, your comments always welcome

Case Study: How Does a Bike-Share Navigate Speedy Success?

Introduction

Welcome to the Cyclistic bike-share analysis case study! In this case study, you will perform many real-world tasks of a junior data analyst. You will work for a fictional company, Cyclistic, and meet different characters and team members. In order to answer the key business questions, you will follow the steps of the data analysis process: ask, prepare, process, analyze, share, and act. Along the way, the Case Study Roadmap tables — including guiding questions and key tasks — will help you stay on the right path. By the end of this lesson, you will have a portfolio-ready case study. Download the packet and reference the details of this case study anytime. Then, when you begin your job hunt, your case study will be a tangible way to demonstrate your knowledge and skills to potential employers

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.

Characters and teams

- **Cyclistic:** A bike-share program that features more than 5,800 bicycles and 600 docking stations. Cyclistic sets itself apart by also offering reclining bikes, hand tricycles, and cargo bikes, making bike-share more inclusive to people with disabilities and riders who can't use a standard two-wheeled bike. The majority of riders opt for traditional bikes; about 8% of riders use the assistive options. Cyclistic users are more likely to ride for leisure, but about 30% use them to commute to work each day.
- **Lily Moreno:** The director of marketing and your manager. Moreno is responsible for the development of campaigns and initiatives to promote the bike-share program. These may include email, social media, and other channels.
- **Cyclistic marketing analytics team:** A team of data analysts who are responsible for collecting, analyzing, and reporting data that helps guide Cyclistic marketing strategy. You joined this team six months ago and have been busy learning about Cyclistic's mission and business goals — as well as how you, as a junior data analyst, can help Cyclistic achieve them.

- **Cyclistic executive team:** The notoriously detail-oriented executive team will decide whether to approve the recommended marketing program.

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?
    Moreno has assigned you the first question to answer: How do annual members and casual riders use Cyclistic bikes differently?

As per the case study question,

Business Task:

To find and explore, How does annnual members and casual riders use Cyclistic bikes differently</P>

Key Stakeholders:

1. Lily Moreno: The director of marketing and your manager.
2. Cyclistic marketing analytics team

3. Cyclistic executive team

Prepare Phase

Key tasks

1. Download data and store it appropriately - done download previous 12 month data click here
   I download 12 month data of 2022
2. Identify how it's organized - Identified
3. Sort and filter the data - done
4. Determine the credibility of the data - verified

Process Phase

Data cleaning and Transforming or manipulating data into consistent format for Analysis by using right and prefered tool

For this case study, I choosed python. Although I kown spreadsheet, SQL and Python was not covered in this certification,I choosed python due to I had good experience and knowledge with python then others also intented to master it for data analysis and data is big for spreadsheet. I though to do casestudy:2 with SQL Following steps , we saw how does data cleaned and transformed for anlaysis

Importing necessary libraries

In [99]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

loading data and store python data frame objects

In [2]:
```
jan22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202201-divvy-tripdata.csv")
jan22
```

Out[2]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_
0	C2F7DD78E82EC875	electric_bike	2022-01-13 11:59:47	2022-01-13 12:02:44	Glenwood Ave & Touhy Ave	525	Clark St &
1	A6CF8980A652D272	electric_bike	2022-01-10 08:41:56	2022-01-10 08:46:17	Glenwood Ave & Touhy Ave	525	Clark St &
2	BD0F91DFF741C66D	classic_bike	2022-01-25 04:53:40	2022-01-25 04:58:01	Sheffield Ave & Fullerton Ave	TA1306000016	Greenview Fullertc
3	CBB80ED419105406	classic_bike	2022-01-04 00:18:04	2022-01-04 00:33:00	Clark St & Bryn Mawr Ave	KA1504000151	Paulin Montros
4	DDC963BFDDA51EEA	classic_bike	2022-01-20	2022-01-20	Michigan Ave & Jackson Blvd	TA1309000002	Stat Randc

			01:31:10	01:37:12			
...	...	...	...	...	...	...	
103765	8788DA3EDE8FD8AB	electric_bike	2022-01-18 12:36:48	2022-01-18 12:46:19	Clinton St & Washington Blvd	WL-012	
103766	C6C3B64FDC827D8C	electric_bike	2022-01-27 11:00:06	2022-01-27 11:02:40	Racine Ave & Randolph St	13155	
103767	CA281AE7D8B06F5A	electric_bike	2022-01-10 16:14:51	2022-01-10 16:20:58	Broadway & Waveland Ave	13325	Clark St & Gr
103768	44E348991862319B	electric_bike	2022-01-19 13:22:11	2022-01-19 13:24:27	Racine Ave & Randolph St	13155	
103769	E477C594A182AE58	electric_bike	2022-01-13 17:24:43	2022-01-13 17:28:14	Clinton St & Washington Blvd	WL-012	Desplaine Ki

103770 rows × 13 columns

In [3]:

```
feb22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202202-divvy-tripdata.csv")
mar22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202203-divvy-tripdata.csv")
arp22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202204-divvy-tripdata.csv")
may22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202205-divvy-tripdata.csv")
june22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202206-divvy-tripdata.csv")
july22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202207-divvy-tripdata.csv")
aug22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202208-divvy-tripdata.csv")
sep22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202209-divvy-publictripdata.csv")
oct22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202210-divvy-tripdata.csv")
nov22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202211-divvy-tripdata.csv")
dec22=pd.read_csv("D:DAGC-Cyclistc analysis/data/202212-divvy-tripdata.csv")#jan22=pd.re
```

Combining all month data to full year dataset

In [100…

```
annual2022=pd.concat([jan22,feb22,mar22,arp22,may22,june22,july22,aug22,sep22,oct22,nov2
```

In [5]:

```
annual2022=annual2022.reset_index(drop=True)#.drop(columns="index")
annual2022
```

Out[5]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station
0	C2F7DD78E82EC875	electric_bike	2022-01-13 11:59:47	2022-01-13 12:02:44	Glenwood Ave & Touhy Ave	525	Clark St &
1	A6CF8980A652D272	electric_bike	2022-01-10 08:41:56	2022-01-10 08:46:17	Glenwood Ave & Touhy Ave	525	Clark St &
2	BD0F91DFF741C66D	classic_bike	2022-01-25 04:53:40	2022-01-25 04:58:01	Sheffield Ave & Fullerton Ave	TA1306000016	Greenview Fuller
3	CBB80ED419105406	classic_bike	2022-01-04 00:18:04	2022-01-04 00:33:00	Clark St & Bryn Mawr Ave	KA1504000151	Pauli Montr
4	DDC963BFDDA51EEA	classic_bike	2022-01-20	2022-01-20	Michigan Ave & Jackson Blvd	TA1309000002	Sta Ranc

			01:31:10	01:37:12			
...	...	...	...	...	...	...	
5667712	43ABEE85B6E15DCA	classic_bike	2022-12-05 06:51:04	2022-12-05 06:54:48	Sangamon St & Washington Blvd	13409	Peo Jacks
5667713	F041C89A3D1F0270	electric_bike	2022-12-14 17:06:28	2022-12-14 17:19:27	Bernard St & Elston Ave	18016	Seele Rc
5667714	A2BECB88430BE156	classic_bike	2022-12-08 16:27:47	2022-12-08 16:32:20	Wacker Dr & Washington St	KA1503000072	Gre Mac
5667715	37B392960E566F58	classic_bike	2022-12-28 09:37:38	2022-12-28 09:41:34	Sangamon St & Washington Blvd	13409	Peo Jacks
5667716	2DD1587210BA45AE	classic_bike	2022-12-09 00:27:25	2022-12-09 00:35:28	Southport Ave & Waveland Ave	13235	Seele Rc

5667717 rows × 13 columns

Below the primary info about the data

In [6]:
```
annual2022.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5667717 entries, 0 to 5667716
Data columns (total 13 columns):
 #   Column              Dtype  
---  ------              -----  
 0   ride_id             object 
 1   rideable_type       object 
 2   started_at          object 
 3   ended_at            object 
 4   start_station_name  object 
 5   start_station_id    object 
 6   end_station_name    object 
 7   end_station_id      object 
 8   start_lat           float64
 9   start_lng           float64
 10  end_lat             float64
 11  end_lng             float64
 12  member_casual       object 
dtypes: float64(4), object(9)
memory usage: 562.1+ MB
```

Changing into respective data Types

In [7]:
```
annual2022["ride_id"]=annual2022["ride_id"].astype("str")
annual2022["rideable_type"]=annual2022["rideable_type"].astype('str')
annual2022["started_at"]=pd.to_datetime(annual2022["started_at"],format="%Y-%m-%d %H:%M:
annual2022["ended_at"]=pd.to_datetime(annual2022["ended_at"],format="%Y-%m-%d %H:%M:%S")
annual2022["start_station_name"]=annual2022["start_station_name"].astype("str")
annual2022["start_station_id"]=annual2022["start_station_id"].astype("str")
annual2022["end_station_name"]=annual2022["end_station_name"].astype("str")
```

```
annual2022["end_station_id"]=annual2022["end_station_id"].astype("str")
annual2022["member_casual"]=annual2022["member_casual"].astype("str")
```

In [8]:
```
annual2022.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5667717 entries, 0 to 5667716
Data columns (total 13 columns):
 #   Column              Dtype
---  ------              -----
 0   ride_id             object
 1   rideable_type       object
 2   started_at          datetime64[ns]
 3   ended_at            datetime64[ns]
 4   start_station_name  object
 5   start_station_id    object
 6   end_station_name    object
 7   end_station_id      object
 8   start_lat           float64
 9   start_lng           float64
 10  end_lat             float64
 11  end_lng             float64
 12  member_casual       object
dtypes: datetime64[ns](2), float64(4), object(7)
memory usage: 562.1+ MB
```

checking for null values

In [9]:
```
annual2022.isnull().sum()
```

Out[9]:
```
ride_id                  0
rideable_type            0
started_at               0
ended_at                 0
start_station_name       0
start_station_id         0
end_station_name         0
end_station_id           0
start_lat                0
start_lng                0
end_lat               5858
end_lng               5858
member_casual            0
dtype: int64
```

Checking for duplicate values

In [10]:
```
annual2022.duplicated().sum()
```

Out[10]:
```
0
```

In [11]:
```
annual2022[annual2022["end_lat"].isna()]
```

Out[11]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station
19717	C1AB102E01C34020	classic_bike	2022-01-26 16:56:03	2022-01-27 17:55:56	Michigan Ave & Jackson Blvd	TA1309000002	
19746	1A51C738B3CD1B3A	classic_bike	2022-01-10 18:50:12	2022-01-11 19:50:05	Western Ave & Leland Ave	TA1307000140	
20465	17BC9F8B24C3D9B7	classic_bike	2022-01-	2022-01-	Christiana Ave &	15615	

			25 21:38:09	26 22:38:04	Lawrence Ave	
20525	6C05E25B083BCA23	classic_bike	2022-01-15 15:10:21	2022-01-16 16:10:14	Theater on the Lake	TA1308000001
20653	AF572A09F5BF185F	classic_bike	2022-01-19 00:54:32	2022-01-20 01:54:25	Kedzie Ave & Milwaukee Ave	13085
...	...	...	...	...	...	...
5661040	0CA67381200B0B0F	classic_bike	2022-12-21 16:29:07	2022-12-22 17:28:47	Franklin St & Lake St	TA1307000111
5661168	D91304881E077C2B	classic_bike	2022-12-20 12:51:05	2022-12-21 13:50:58	Jeffery Blvd & 71st St	KA1503000018
5661311	CBBB0A4A1498F790	docked_bike	2022-12-30 20:35:36	2022-12-31 21:35:36	Michigan Ave & Washington St	13001
5661529	866EF596D264C2B4	classic_bike	2022-12-08 19:42:40	2022-12-09 20:42:32	Jeffery Blvd & 71st St	KA1503000018
5661604	24B4E1CF232C216B	classic_bike	2022-12-02 14:49:05	2022-12-03 15:48:57	Morgan St & 18th St	13163

5858 rows × 13 columns

Some null fields detected and no duplicates detected. so, droping null records

In [12]:
```
annual2022.dropna(inplace=True)
```

In [13]:
```
annual2022.isnull().sum()
```

Out[13]:
```
ride_id               0
rideable_type         0
started_at            0
ended_at              0
start_station_name    0
start_station_id      0
end_station_name      0
end_station_id        0
start_lat             0
start_lng             0
end_lat               0
end_lng               0
member_casual         0
dtype: int64
```

In [14]:
```
annual2022[annual2022["start_station_name"]=="Theater on the Lake"]["end_station_name"].
```

Out[14]:
```
Theater on the Lake                         3070
Streeter Dr & Grand Ave                     2068
nan                                         1658
Michigan Ave & Oak St                       1558
DuSable Lake Shore Dr & North Blvd           958
                                            ...
Public Rack - Kenton Ave & Palmer St           1
Dorchester Ave & 63rd St                       1
```

```
Racine Ave & 15th St                    1
Halsted St & Polk St                    1
Leavitt St & Chicago Ave                1
Name: end_station_name, Length: 417, dtype: int64
```

using below code check for invalid data starting and end station same , it is impossible right.

we detected 7 lakhs + invalid records and droped it

In [15]:
```
annual2022[annual2022["start_station_name"]==annual2022["end_station_name"]].shape
```
Out[15]:
```
(721068, 13)
```

In [16]:
```
annual2022[annual2022["start_station_name"]=="Kedzie Ave & Milwaukee Ave"]["end_station_
```
Out[16]:
```
nan                                    3300
Kedzie Ave & Milwaukee Ave              960
Kosciuszko Park                         577
California Ave & Milwaukee Ave          411
St. Louis Ave & Fullerton Ave           409
                                       ...
Public Rack - Jensen Park                 1
N Carpenter St & W Lake St                1
Mies van der Rohe Way & Chicago Ave       1
Central Ave & Chicago Ave                 1
Glenwood Ave & Touhy Ave                  1
Name: end_station_name, Length: 488, dtype: int64
```

below we checked for nan invalid data and cleaned it

In [17]:
```
annual2022[annual2022["start_station_name"]=="nan"].shape
```
Out[17]:
```
(833064, 13)
```

In [18]:
```
annual2022[annual2022["end_station_name"]=="nan"].shape
```
Out[18]:
```
(886884, 13)
```

In [19]:
```
annual2022[annual2022["end_station_name"]=="nan"].head()
```
Out[19]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_na
510	88276B47FFBB9910	electric_bike	2022-01-25 07:39:35	2022-01-25 07:41:01	Larrabee St & Kingsbury St	TA1306000009	
512	1B66EC28DD618680	electric_bike	2022-01-21 14:26:57	2022-01-21 14:32:17	Central Park Ave & Ohio St	369	
19594	20994C14E5606D05	electric_bike	2022-01-29 22:20:02	2022-01-29 22:24:20	Seeley Ave & Roscoe St	13144	
19595	3C647408C8ED11FA	electric_bike	2022-01-31 08:28:21	2022-01-31 09:10:13	Spaulding Ave & Division St	15654	
19596	A1ED4CB525BE0030	electric_bike	2022-01-11 16:07:57	2022-01-11 16:26:55	Kedzie Ave & Bryn Mawr Ave	KA1504000167	

In [20]:
```
annual2022[(annual2022["end_lat"]==41.90)&(annual2022["end_lng"]==-87.64)]["end_station_
```
Out[20]:
```
nan    5605
Name: end_station_name, dtype: int64
```
In [21]:
```
annual2022[annual2022["started_at"]=="nan"].shape
```
Out[21]:
```
(0, 13)
```
In [22]:
```
annual2022[annual2022["ended_at"]=="nan"].shape
```
Out[22]:
```
(0, 13)
```
In [23]:
```
annual2022[annual2022["ride_id"]=="nan"].shape
```
Out[23]:
```
(0, 13)
```
In [24]:
```
annual2022[annual2022["rideable_type"]=="nan"].shape
```
Out[24]:
```
(0, 13)
```
In [25]:
```
annual2022[annual2022["start_station_name"]==" "].shape
```
Out[25]:
```
(0, 13)
```
In [26]:
```
annual2022[annual2022["end_station_name"]==" "].shape
```
Out[26]:
```
(0, 13)
```
In [27]:
```
annual2022[annual2022["member_casual"]=="nan"].shape
```
Out[27]:
```
(0, 13)
```
In [28]:
```
annual2022_clean=annual2022[~(annual2022["start_station_name"]==annual2022["end_station_
```
In [29]:
```
annual2022_clean=annual2022_clean[~(annual2022_clean["start_station_name"]=="nan")]
```
In [30]:
```
annual2022_clean=annual2022_clean[~(annual2022_clean["end_station_name"]=="nan")]
```
In [31]:
```
annual2022_clean.info()
```
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4075741 entries, 0 to 5667716
Data columns (total 13 columns):
 #   Column              Dtype
---  ------              -----
 0   ride_id             object
 1   rideable_type       object
 2   started_at          datetime64[ns]
 3   ended_at            datetime64[ns]
 4   start_station_name  object
 5   start_station_id    object
 6   end_station_name    object
 7   end_station_id      object
 8   start_lat           float64
 9   start_lng           float64
 10  end_lat             float64
 11  end_lng             float64
 12  member_casual       object
dtypes: datetime64[ns](2), float64(4), object(7)
memory usage: 435.3+ MB
```

Transform and manipulating data(Feature Engineering)

We created new column or property *ride_length* to measure total time for each rider _id by difference of ride ride ending and starting time

In [32]:
```
annual2022_clean["ride_length"]=annual2022_clean["ended_at"]-annual2022_clean["started_a
```

In [33]:
```
ts=pd.to_timedelta("0 days 00:00:00")
```

In [34]:
```
annual2022_clean[annual2022_clean["ride_length"]<=ts].shape
```

Out[34]:
```
(32, 14)
```

negative value of ride length is impossible. so, we clean the that error data

In [35]:
```
annual2022_clean=annual2022_clean[~(annual2022_clean["ride_length"]<=ts)]
```

Creating another two features such as month and weekday(day of the week)

In [36]:
```
import warnings
warnings.filterwarnings('ignore')
annual2022_clean["month"]=annual2022_clean["started_at"].dt.month
```

In [37]:
```
annual2022_clean["weekday"]=annual2022_clean["started_at"].dt.weekday
```

In [38]:
```
annual2022_clean.head()
```

Out[38]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name
0	C2F7DD78E82EC875	electric_bike	2022-01-13 11:59:47	2022-01-13 12:02:44	Glenwood Ave & Touhy Ave	525	Clark St & Touhy Ave
1	A6CF8980A652D272	electric_bike	2022-01-10 08:41:56	2022-01-10 08:46:17	Glenwood Ave & Touhy Ave	525	Clark St & Touhy Ave
2	BD0F91DFF741C66D	classic_bike	2022-01-25 04:53:40	2022-01-25 04:58:01	Sheffield Ave & Fullerton Ave	TA1306000016	Greenview Ave & Fullerton Ave
3	CBB80ED419105406	classic_bike	2022-01-04 00:18:04	2022-01-04 00:33:00	Clark St & Bryn Mawr Ave	KA1504000151	Paulina St & Montrose Ave
4	DDC963BFDDA51EEA	classic_bike	2022-01-20 01:31:10	2022-01-20 01:37:12	Michigan Ave & Jackson Blvd	TA1309000002	State St & Randolph St

Analyze Phase

Descriptive Statistics about data

```
In [101…: annual2022_clean.describe()
```

Out[101]:

	start_lat	start_lng	end_lat	end_lng	ride_length	month	weekday
count	4.075709e+06	4.075709e+06	4.075709e+06	4.075709e+06	4075709	4.075709e+06	4.075709e+06
mean	4.190229e+01	-8.764495e+01	4.190254e+01	-8.764506e+01	0 days 00:16:24.684727736	7.094115e+00	3.037280e+00
std	4.162281e-02	2.428444e-02	7.208026e-02	1.252128e-01	0 days 00:47:17.354612684	2.525271e+00	1.973683e+00
min	4.164850e+01	-8.783332e+01	0.000000e+00	-8.783000e+01	0 days 00:00:01	1.000000e+00	0.000000e+00
25%	4.188132e+01	-8.765814e+01	4.188169e+01	-8.765840e+01	0 days 00:06:15	5.000000e+00	1.000000e+00
50%	4.189724e+01	-8.764263e+01	4.189776e+01	-8.764288e+01	0 days 00:10:36	7.000000e+00	3.000000e+00
75%	4.192877e+01	-8.762932e+01	4.192889e+01	-8.762932e+01	0 days 00:18:27	9.000000e+00	5.000000e+00
max	4.206487e+01	-8.752531e+01	4.206485e+01	0.000000e+00	23 days 20:34:04	1.200000e+01	6.000000e+00

Descriptive statistics of ride length of casual and member riders

below stat shows no of member riders is greater than the no of casual riders, but average and max ride length of casual riders is greater than member riders

```
In [40]: annual2022_clean.groupby("member_casual")["ride_length"].describe()
```

Out[40]:

member_casual	count	mean	std	min	25%	50%	75%	max
casual	1582601	0 days 00:22:41.256003250	0 days 01:11:45.255480542	0 days 00:00:01	0 days 00:08:08	0 days 00:13:40	0 days 00:24:18	23 days 20:34:04
member	2493108	0 days 00:12:25.640900835	0 days 00:18:37.083677047	0 days 00:00:01	0 days 00:05:26	0 days 00:09:04	0 days 00:15:17	1 days 00:53:14

Total ride length of casual and member riders

```
In [41]: annual2022_clean.groupby("member_casual")["ride_length"].sum()
```

```
Out[41]: member_casual
casual    24934 days 07:38:32
member    21515 days 18:41:35
Name: ride_length, dtype: timedelta64[ns]
```

Descriptive Statistic of ride length of member and casual riders with respect to each month of 2022

```
In [42]: annual2022_clean.groupby(["member_casual","month"])["ride_length"].describe()
```

Out[42]:

member_casual	month	count	mean	std	min	25%	50%	75%
casual	**1**	11555	0 days 00:26:07.423366508	0 days 06:58:03.559627886	0 days 00:00:49	0 days 00:06:20	0 days 00:09:55	0 days 00:16:3
	2	13752	0 days 00:22:40.954770215	0 days 02:18:50.353370239	0 days 00:00:49	0 days 00:06:50	0 days 00:10:52.500000	0 days 00:19:1

	3	59408	0 days 00:26:42.440849717	0 days 02:56:49.193552226	0 days 00:00:01	0 days 00:08:26	0 days 00:14:34	0 days 00:26:41
	4	82097	0 days 00:24:14.337807715	0 days 00:56:08.839359356	0 days 00:00:05	0 days 00:08:27	0 days 00:14:24	0 days 00:26:05
	5	194815	0 days 00:25:54.253753561	0 days 00:52:18.377877507	0 days 00:00:02	0 days 00:09:20	0 days 00:15:48	0 days 00:28:13
	6	261963	0 days 00:23:35.851643170	0 days 00:42:22.927217315	0 days 00:00:01	0 days 00:08:56	0 days 00:14:50	0 days 00:25:33
	7	278177	0 days 00:23:49.899847938	0 days 01:17:53.127232592	0 days 00:00:01	0 days 00:08:45	0 days 00:14:42	0 days 00:25:50
	8	243550	0 days 00:22:05.164356394	0 days 00:39:05.632964727	0 days 00:00:01	0 days 00:08:13	0 days 00:13:38	0 days 00:24:02
	9	201823	0 days 00:20:48.278749201	0 days 00:37:22.687672902	0 days 00:00:01	0 days 00:07:44	0 days 00:12:47	0 days 00:22:27
	10	138418	0 days 00:19:47.095688422	0 days 00:43:18.847808215	0 days 00:00:09	0 days 00:07:01	0 days 00:11:43	0 days 00:21:04
	11	67705	0 days 00:16:32.011815966	0 days 00:36:35.770646853	0 days 00:00:13	0 days 00:06:01	0 days 00:09:53	0 days 00:17:14
	12	29338	0 days 00:14:22.037528120	0 days 00:37:41.749823738	0 days 00:00:20	0 days 00:05:28	0 days 00:08:39	0 days 00:14:21
member	**1**	65085	0 days 00:10:11.091357455	0 days 00:14:12.501615017	0 days 00:00:32	0 days 00:04:44	0 days 00:07:29	0 days 00:12:12
	2	71118	0 days 00:10:37.089977220	0 days 00:18:36.824827532	0 days 00:00:17	0 days 00:04:46	0 days 00:07:36	0 days 00:12:32
	3	143082	0 days 00:11:42.397359556	0 days 00:21:51.307341070	0 days 00:00:05	0 days 00:05:01	0 days 00:08:16	0 days 00:13:53
	4	173556	0 days 00:11:32.231913618	0 days 00:20:50.593879643	0 days 00:00:02	0 days 00:05:00	0 days 00:08:16	0 days 00:13:51
	5	269034	0 days 00:13:14.071496539	0 days 00:17:52.555847837	0 days 00:00:07	0 days 00:05:43	0 days 00:09:43	0 days 00:16:33
	6	313126	0 days 00:13:39.736997247	0 days 00:18:16.337432800	0 days 00:00:01	0 days 00:06:05	0 days 00:10:13	0 days 00:17:07
	7	314659	0 days 00:13:30.847803495	0 days 00:16:11.309659336	0 days 00:00:02	0 days 00:06:01	0 days 00:10:07	0 days 00:17:03
	8	318807	0 days 00:13:05.060575834	0 days 00:17:21.041512133	0 days 00:00:03	0 days 00:05:53	0 days 00:09:48	0 days 00:16:21
	9	299197	0 days 00:12:37.447454352	0 days 00:19:25.995638146	0 days 00:00:03	0 days 00:05:34	0 days 00:09:18	0 days 00:15:32
	10	250969	0 days 00:11:42.129314775	0 days 00:20:15.366028543	0 days 00:00:01	0 days 00:05:03	0 days 00:08:24	0 days 00:14:06
	11	174962	0 days 00:10:48.904459254	0 days 00:19:09.796608946	0 days 00:00:10	0 days 00:04:47	0 days 00:07:49	0 days 00:12:58
	12	99513	0 days 00:10:14.307115653	0 days 00:17:30.295363930	0 days 00:00:03	0 days 00:04:40	0 days 00:07:28	0 days 00:12:07

Total ride length of member and casual riders in each month of 2022

In [43]:
```
annual2022_clean.groupby(["member_casual","month"])["ride_length"].sum()
```

Out[43]:
```
member_casual  month
casual         1         209 days 14:59:37
               2         216 days 14:50:50
               3        1101 days 19:50:06
               4        1381 days 21:46:11
               5        3504 days 12:52:25
               6        4292 days 19:59:04
               7        4603 days 18:20:50
               8        3735 days 11:02:59
               9        2915 days 20:56:02
               10       1901 days 19:10:11
               11        777 days 08:42:40
               12        292 days 17:07:37
member         1         460 days 08:01:21
               2         524 days 09:42:45
               3        1163 days 04:46:59
               4        1390 days 12:30:02
               5        2472 days 14:17:11
               6        2970 days 20:16:07
               7        2953 days 00:22:39
               8        2896 days 19:00:07
               9        2622 days 23:40:06
               10       2039 days 11:58:12
               11       1314 days 01:07:02
               12        707 days 12:59:04
Name: ride_length, dtype: timedelta64[ns]
```

Descriptive Statistic of ride length of member and casual riders with respect to Day of Week

In [44]:
```
annual2022_clean.groupby(["member_casual","weekday"])["ride_length"].describe()
```

Out[44]:

member_casual	weekday	count	mean	std	min	25%	50%	75%	
casual	**0**	187089	0 days 00:23:10.448642090	0 days 01:35:26.839146753	0 days 00:00:01	0 days 00:07:50	0 days 00:13:26	0 days 00:24:32	22 05
	1	178112	0 days 00:20:02.904818316	0 days 00:47:36.035915557	0 days 00:00:02	0 days 00:07:16	0 days 00:11:59	0 days 00:20:58	7 13
	2	185190	0 days 00:19:26.011048112	0 days 00:58:42.159117037	0 days 00:00:01	0 days 00:07:17	0 days 00:11:54	0 days 00:20:31	9 16
	3	209520	0 days 00:20:13.467210767	0 days 01:20:48.693873970	0 days 00:00:01	0 days 00:07:28	0 days 00:12:14	0 days 00:21:12	18 19
	4	226160	0 days 00:21:19.125446586	0 days 00:46:14.786953670	0 days 00:00:05	0 days 00:07:56	0 days 00:13:08	0 days 00:22:57	6 22
	5	330103	0 days 00:25:44.469265653	0 days 01:32:38.023633102	0 days 00:00:01	0 days 00:09:24	0 days 00:15:54	0 days 00:28:08	23 20
	6	266427	0 days 00:25:41.268332413	0 days 00:51:31.185352472	0 days 00:00:02	0 days 00:09:14	0 days 00:15:49	0 days 00:28:03	7 12
member	**0**	358212	0 days 00:11:56.636642546	0 days 00:18:41.581591145	0 days 00:00:01	0 days 00:05:13	0 days 00:08:39	0 days 00:14:36	1 00
	1	394755	0 days 00:11:45.416823092	0 days 00:18:04.487744863	0 days 00:00:02	0 days 00:05:16	0 days 00:08:41	0 days 00:14:26	1 00
	2	395802	0 days 00:11:49.652791041	0 days 00:16:47.433853127	0 days 00:00:01	0 days 00:05:19	0 days 00:08:48	0 days 00:14:37	1 00

3	398607	0 days 00:12:01.153652595	0 days 00:18:02.534944402	0 days 00:00:03	0 days 00:05:21	0 days 00:08:52	0 days 00:14:48	00
4	343736	0 days 00:12:13.835711127	0 days 00:18:50.634209807	0 days 00:00:03	0 days 00:05:23	0 days 00:08:56	0 days 00:14:57	00
5	320800	0 days 00:14:01.989725685	0 days 00:21:04.657554543	0 days 00:00:03	0 days 00:05:58	0 days 00:10:12	0 days 00:17:24	00
6	281196	0 days 00:13:48.936713182	0 days 00:19:04.336947157	0 days 00:00:03	0 days 00:05:47	0 days 00:09:56	0 days 00:17:13	00

Total of ride length of member and casual riders with respect to day of the week

In [45]:
```
annual2022_clean.groupby(["member_casual","weekday"])["ride_length"].sum()
```

Out[45]:
```
member_casual  weekday
casual         0          3010 days 20:27:26
               1          2479 days 18:23:03
               2          2499 days 05:33:06
               3          2942 days 15:47:30
               4          3348 days 05:30:11
               5          5900 days 20:32:18
               6          4752 days 17:24:58
member         0          2971 days 03:44:05
               1          3222 days 23:53:38
               2          3250 days 22:46:34
               3          3327 days 01:08:14
               4          2919 days 12:15:52
               5          3126 days 06:38:24
               6          2697 days 20:14:48
Name: ride_length, dtype: timedelta64[ns]
```

Descriptive Statistic of ride length with respect to rideable_type

In [46]:
```
annual2022_clean.groupby("rideable_type")["ride_length"].describe()
```

Out[46]:

	count	mean	std	min	25%	50%	75%	max
rideable_type								
classic_bike	2439557	0 days 00:16:28.962037779	0 days 00:30:35.089376281	0 days 00:00:03	0 days 00:06:19	0 days 00:10:50	0 days 00:18:59	1 days 00:59:25
docked_bike	143456	0 days 00:47:42.387073388	0 days 03:31:09.450136473	0 days 00:01:10	0 days 00:15:41	0 days 00:26:07	0 days 00:47:23	23 days 20:34:04
electric_bike	1492696	0 days 00:13:17.237042907	0 days 00:13:43.194771248	0 days 00:00:01	0 days 00:05:52	0 days 00:09:35	0 days 00:15:50	0 days 08:00:00

Total ride length of each bike type

In [47]:
```
annual2022_clean.groupby("rideable_type")["ride_length"].sum()
```

Out[47]:
```
rideable_type
classic_bike    27923 days 22:47:42
docked_bike      4752 days 14:56:40
electric_bike   13773 days 12:35:45
Name: ride_length, dtype: timedelta64[ns]
```

Total ride length of member and casual riders with respect to each bike type

In [48]:
```
annual2022_clean.groupby(["member_casual","rideable_type"])["ride_length"].sum()
```

Out[48]:
```
member_casual  rideable_type
casual         classic_bike    13036 days 08:24:15
               docked_bike      4752 days 14:56:40
               electric_bike    7145 days 08:17:37
member         classic_bike    14887 days 14:23:27
               electric_bike    6628 days 04:18:08
Name: ride_length, dtype: timedelta64[ns]
```

Descriptive Statistic of ride length of member and casual riders with respect to each bike type

In [49]:
```
annual2022_clean.groupby(["member_casual","rideable_type"])["ride_length"].describe()
```

Out[49]:

member_casual	rideable_type	count	mean	std	min	25%	50%	75%
casual	classic_bike	804150	0 days 00:23:20.659895541	0 days 00:42:43.618945608	0 days 00:00:12	0 days 00:08:37	0 days 00:14:21	0 day 00:25:1
	docked_bike	143456	0 days 00:47:42.387073388	0 days 03:31:09.450136473	0 days 00:01:10	0 days 00:15:41	0 days 00:26:07	0 day 00:47:2
	electric_bike	634995	0 days 00:16:12.224752950	0 days 00:16:13.330406339	0 days 00:00:01	0 days 00:06:59	0 days 00:11:18	0 day 00:19:0
member	classic_bike	1635407	0 days 00:13:06.525071129	0 days 00:21:31.315618192	0 days 00:00:03	0 days 00:05:33	0 days 00:09:25	0 day 00:16:1
	electric_bike	857701	0 days 00:11:07.685694665	0 days 00:11:02.200297767	0 days 00:00:01	0 days 00:05:14	0 days 00:08:31	0 day 00:13:4

Descriptive Statistic of ride length of member and casual riders with respect to day of the week and each bike type

In [50]:
```
annual2022_clean.groupby(["member_casual","weekday","rideable_type"])["ride_length"].des
```

Out[50]:

member_casual	weekday	rideable_type	count	mean	std	min	25%
casual	0	classic_bike	92734	0 days 00:23:28.543457631	0 days 00:41:45.034947361	0 days 00:00:52	0 days 00:08:22
		docked_bike	17798	0 days 00:52:09.706933363	0 days 04:50:35.394448466	0 days 00:01:15	0 days 00:16:07
		electric_bike	76557	0 days 00:16:04.186854239	0 days 00:16:36.396928618	0 days 00:00:01	0 days 00:06:39
	1	classic_bike	87202	0 days 00:21:14.912891906	0 days 00:43:37.021037150	0 days 00:00:47	0 days 00:07:45
		docked_bike	14386	0 days 00:43:54.553454747	0 days 02:01:04.342190918	0 days 00:01:25	0 days 00:14:56
		electric_bike	76524	0 days 00:14:11.708522816	0 days 00:14:24.093086108	0 days 00:00:02	0 days 00:06:21

	2	**classic_bike**	89413	0 days 00:20:18.470859942	0 days 00:38:30.500726108	0 days 00:00:20	0 days 00:07:46
		docked_bike	14170	0 days 00:44:50.942201834	0 days 03:03:43.595343698	0 days 00:02:03	0 days 00:14:39.250000
		electric_bike	81607	0 days 00:14:03.748698028	0 days 00:13:52.657899419	0 days 00:00:01	0 days 00:06:24
	3	**classic_bike**	103520	0 days 00:20:54.789074574	0 days 00:40:48.602460312	0 days 00:00:19	0 days 00:07:54
		docked_bike	16140	0 days 00:47:28.457496902	0 days 04:28:18.771088837	0 days 00:01:10	0 days 00:15:04
		electric_bike	89860	0 days 00:14:32.198764745	0 days 00:14:33.440610531	0 days 00:00:01	0 days 00:06:36
	4	**classic_bike**	111883	0 days 00:22:05.587488715	0 days 00:42:33.548107369	0 days 00:00:12	0 days 00:08:18
		docked_bike	19342	0 days 00:43:52.791438320	0 days 01:52:32.063194849	0 days 00:01:18	0 days 00:15:03
		electric_bike	94935	0 days 00:15:48.573803128	0 days 00:15:40.312378706	0 days 00:00:05	0 days 00:06:59
	5	**classic_bike**	178250	0 days 00:25:52.124577840	0 days 00:44:16.173955974	0 days 00:00:29	0 days 00:09:49
		docked_bike	33457	0 days 00:49:49.136533460	0 days 04:28:56.330285462	0 days 00:01:23	0 days 00:16:21
		electric_bike	118396	0 days 00:18:44.701772019	0 days 00:17:58.777660825	0 days 00:00:01	0 days 00:08:00
	6	**classic_bike**	141148	0 days 00:26:03.791346671	0 days 00:44:29.300714198	0 days 00:00:23	0 days 00:09:43
		docked_bike	28163	0 days 00:48:31.182828533	0 days 01:55:38.263155993	0 days 00:01:37	0 days 00:16:21
		electric_bike	97116	0 days 00:18:31.267298900	0 days 00:18:01.228890089	0 days 00:00:02	0 days 00:07:48
member	**0**	**classic_bike**	236606	0 days 00:12:36.683249790	0 days 00:21:38.808372950	0 days 00:00:25	0 days 00:05:20
		electric_bike	121606	0 days 00:10:38.718879002	0 days 00:10:43.542618858	0 days 00:00:01	0 days 00:05:00
	1	**classic_bike**	258080	0 days 00:12:25.038092839	0 days 00:21:07.836659469	0 days 00:00:16	0 days 00:05:22
		electric_bike	136675	0 days 00:10:30.600965794	0 days 00:09:54.265898557	0 days 00:00:02	0 days 00:05:06
	2	**classic_bike**	256019	0 days 00:12:28.755393154	0 days 00:19:30.153873268	0 days 00:00:09	0 days 00:05:25
		electric_bike	139783	0 days 00:10:38.034575019	0 days 00:09:58.342452760	0 days 00:00:01	0 days 00:05:09
	3	**classic_bike**	256856	0 days 00:12:41.806444077	0 days 00:21:07.142494313	0 days 00:00:10	0 days 00:05:27
		electric_bike	141751	0 days 00:10:47.489880141	0 days 00:10:14.384988869	0 days 00:00:03	0 days 00:05:11
	4	**classic_bike**	221776	0 days 00:12:52.365675275	0 days 00:21:51.656778546	0 days 00:00:16	0 days 00:05:29

	electric_bike	121960	0 days 00:11:03.771580846	0 days 00:11:23.217323242	0 days 00:00:03	0 days 00:05:14
5	**classic_bike**	215913	0 days 00:14:45.314737880	0 days 00:24:09.051428453	0 days 00:00:03	0 days 00:06:08
	electric_bike	104887	0 days 00:12:32.803903248	0 days 00:12:26.646082840	0 days 00:00:04	0 days 00:05:42
6	**classic_bike**	190157	0 days 00:14:28.545912062	0 days 00:21:17.980980634	0 days 00:00:15	0 days 00:05:55
	electric_bike	91039	0 days 00:12:26.203308472	0 days 00:13:09.436883247	0 days 00:00:03	0 days 00:05:31

Total of ride length of member and casual riders with respect to day of the week and each bike type

In [51]:
```
annual2022_clean.groupby(["member_casual","weekday","rideable_type"])["ride_length"].sum
```

Out[51]:
```
member_casual  weekday  rideable_type
casual         0        classic_bike     1511 days 19:17:49
                        docked_bike       644 days 16:55:24
                        electric_bike     854 days 08:14:13
               1        classic_bike     1286 days 17:55:54
                        docked_bike       438 days 15:58:06
                        electric_bike     754 days 08:29:03
               2        classic_bike     1260 days 23:05:35
                        docked_bike       441 days 07:50:51
                        electric_bike     796 days 22:36:40
               3        classic_bike     1503 days 10:09:25
                        docked_bike       532 days 02:35:04
                        electric_bike     907 days 03:03:01
               4        classic_bike     1716 days 13:25:05
                        docked_bike       589 days 09:24:12
                        electric_bike    1042 days 06:40:54
               5        classic_bike     3202 days 03:43:26
                        docked_bike      1157 days 11:52:21
                        electric_bike    1541 days 04:56:31
               6        classic_bike     2554 days 16:47:01
                        docked_bike       948 days 22:20:42
                        electric_bike    1249 days 02:17:15
member         0        classic_bike     2072 days 04:09:57
                        electric_bike     898 days 23:34:08
               1        classic_bike     2225 days 10:57:11
                        electric_bike     997 days 12:56:27
               2        classic_bike     2218 days 16:46:47
                        electric_bike    1032 days 05:59:47
               3        classic_bike     2264 days 18:02:36
                        electric_bike    1062 days 07:05:38
               4        classic_bike     1982 days 13:09:30
                        electric_bike     936 days 23:06:22
               5        classic_bike     2212 days 09:29:21
                        electric_bike     913 days 21:09:03
               6        classic_bike     1911 days 13:48:05
                        electric_bike     786 days 06:26:43
Name: ride_length, dtype: timedelta64[ns]
```

Top 10 starting stations of the member riders

In [52]:
```
annual2022_clean[annual2022_clean["member_casual"]=="member"]["start_station_name"].valu
```

Out[52]:
```
Kingsbury St & Kinzie St        22934
```

```
Clark St & Elm St                19703
Wells St & Concord Ln            19090
Clinton St & Washington Blvd     18447
Clinton St & Madison St          17591
Loomis St & Lexington St         16882
Wells St & Elm St                16867
University Ave & 57th St         16695
Ellis Ave & 60th St              16684
Broadway & Barry Ave             15459
Name: start_station_name, dtype: int64
```

Top 10 starting stations of the casual riders

In [53]:
```
annual2022_clean[annual2022_clean["member_casual"]=="casual"]["start_station_name"].valu
```

Out[53]:
```
Streeter Dr & Grand Ave               44429
DuSable Lake Shore Dr & Monroe St     23647
Millennium Park                       19890
DuSable Lake Shore Dr & North Blvd    19718
Michigan Ave & Oak St                 19149
Shedd Aquarium                        16942
Theater on the Lake                   15079
Wells St & Concord Ln                 14111
Clark St & Armitage Ave               11815
Clark St & Lincoln Ave                11614
Name: start_station_name, dtype: int64
```

Top 10 Ending stations of the member riders

In [54]:
```
annual2022_clean[annual2022_clean["member_casual"]=="member"]["end_station_name"].value_
```

Out[54]:
```
Kingsbury St & Kinzie St         22624
Clark St & Elm St                20024
Wells St & Concord Ln            19688
Clinton St & Washington Blvd     19115
Clinton St & Madison St          18104
University Ave & 57th St         17553
Wells St & Elm St                16837
Loomis St & Lexington St         16695
Ellis Ave & 60th St              16499
Broadway & Barry Ave             15802
Name: end_station_name, dtype: int64
```

Top 10 Ending stations of the casual riders

In [55]:
```
annual2022_clean[annual2022_clean["member_casual"]=="casual"]["end_station_name"].value_
```

Out[55]:
```
Streeter Dr & Grand Ave               47178
DuSable Lake Shore Dr & North Blvd    22866
DuSable Lake Shore Dr & Monroe St     21928
Millennium Park                       21615
Michigan Ave & Oak St                 20761
Theater on the Lake                   16395
Shedd Aquarium                        15568
Wells St & Concord Ln                 13693
Clark St & Armitage Ave               12066
Clark St & Lincoln Ave                11996
Name: end_station_name, dtype: int64
```

Share Phase

share our findings using Visualizations

In [56]:
```
mem_month=(annual2022_clean[annual2022_clean["member_casual"]=="member"].groupby("month"
```

In [57]:
```
def to_minutes(td):
    return td.total_seconds() / 60
```

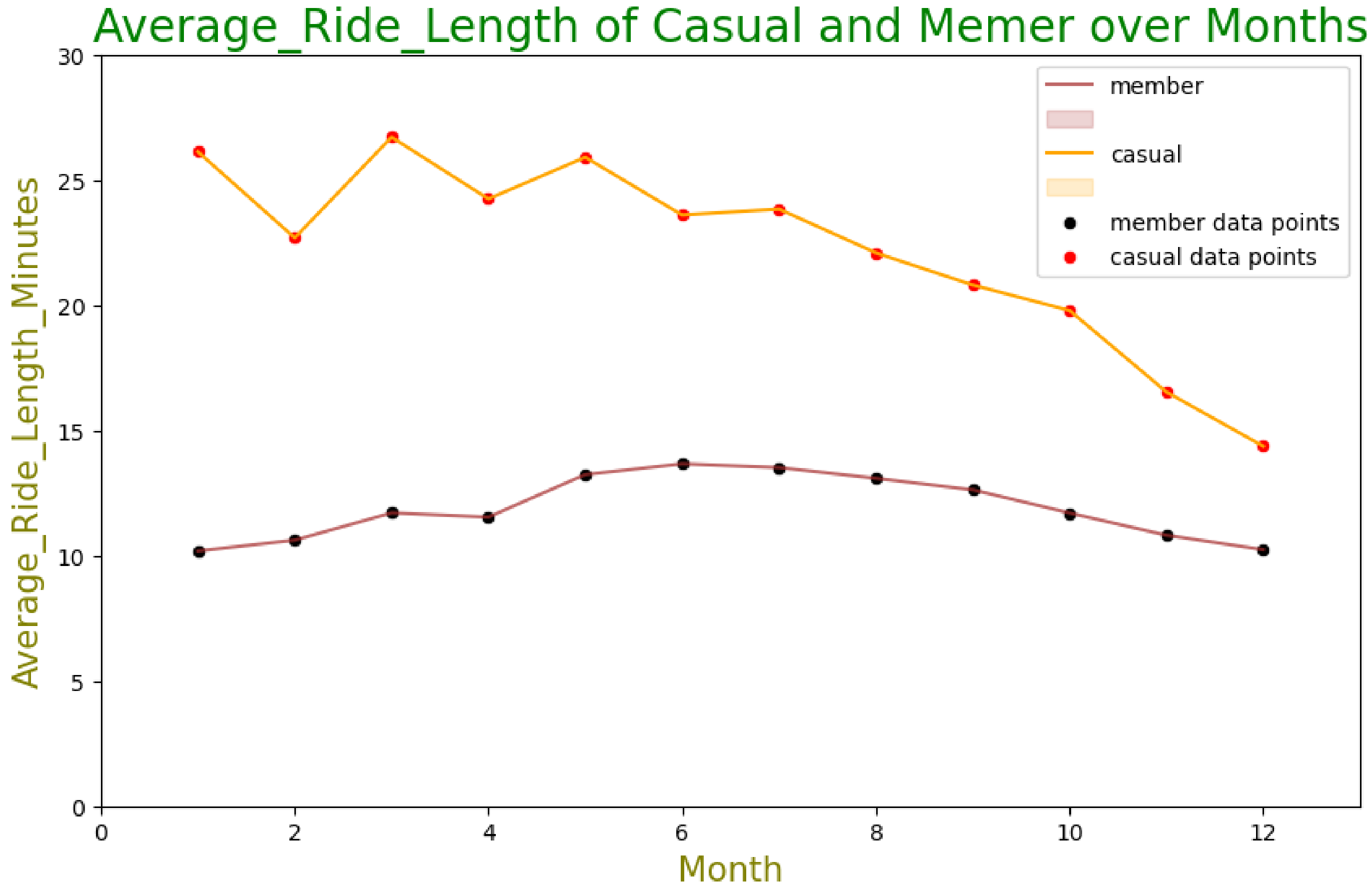
In [58]:
```
def to_days(td):
    return td.total_seconds() / (60*60*24)
```

In [59]:
```
mem_month.apply(to_minutes)
```

Out[59]:
```
month
1     10.184856
2     10.618166
3     11.706623
4     11.537199
5     13.234525
6     13.662283
7     13.514130
8     13.084343
9     12.624124
10    11.702155
11    10.815074
12    10.238452
Name: ride_length, dtype: float64
```

In [60]:
```
casual_month=(annual2022_clean[annual2022_clean["member_casual"]=="casual"].groupby("mon
```

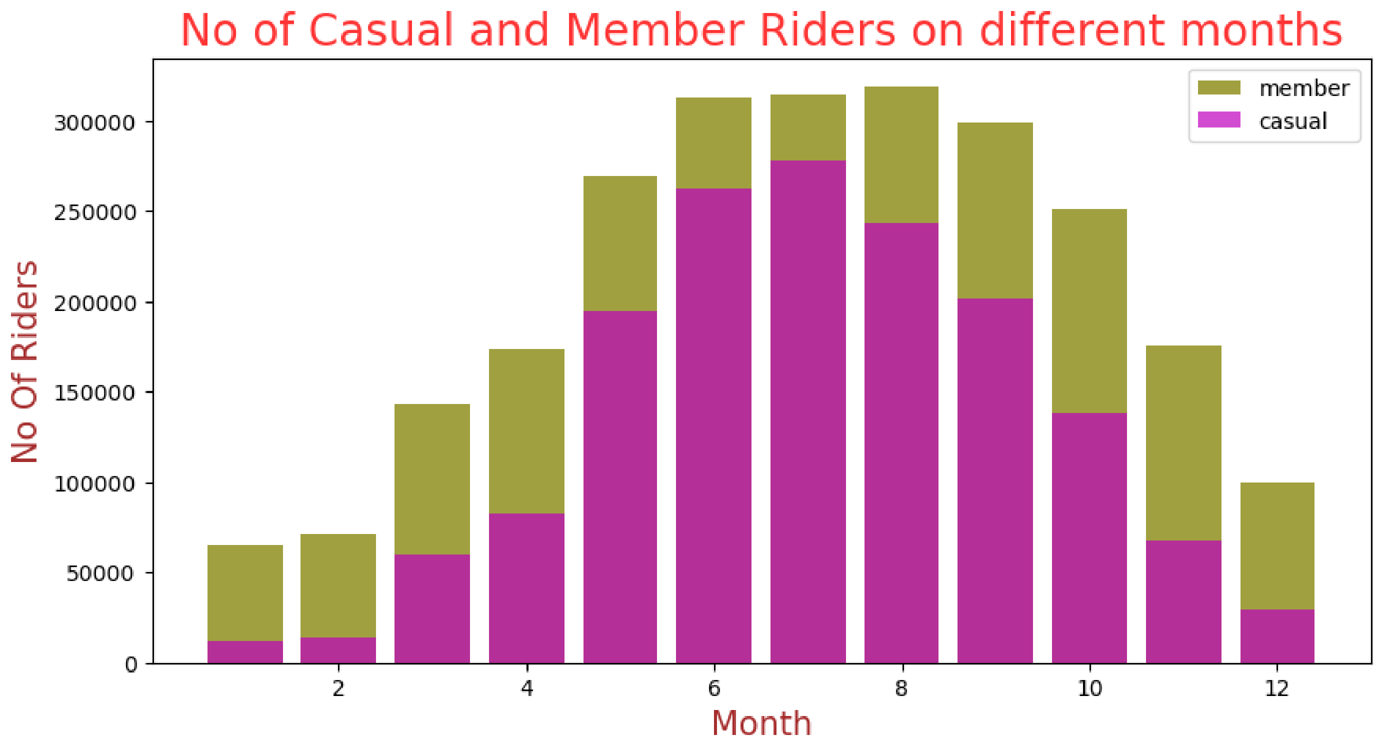
In [61]:
```
plt.figure(figsize=(10,6))
sns.lineplot(data=mem_month,x=mem_month.index,y=mem_month.apply(to_minutes),color="brown
sns.lineplot(data=casual_month,x=casual_month.index,y=casual_month.apply(to_minutes),col
sns.scatterplot(data=mem_month,x=mem_month.index,y=mem_month.apply(to_minutes),color="bl
sns.scatterplot(data=casual_month,x=casual_month.index,y=casual_month.apply(to_minutes),
plt.ylim(0,30)
plt.xlim(0,13)
plt.ylabel("Average_Ride_Length_Minutes",color="olive",size=15)
plt.xlabel("Month",color="olive",size=15)
plt.title("Average_Ride_Length of Casual and Memer over Months",size=20,color='g')
plt.legend(["member","","casual","","member data points","casual data points"],loc="uppe
plt.show()
```



Above graph shows the decreasing trend of average ride length of casual riders and seasonal changes in ride length of member riders, particularly ride increased in 5th, 6th, 7th month of 2022

In [62]:
```
mem_mcount=annual2022_clean[annual2022_clean["member_casual"]=="member"].groupby("month"
casual_mcount=annual2022_clean[annual2022_clean["member_casual"]=="casual"].groupby("mon
```

In [63]:
```
plt.figure(figsize=(10,5))
plt.bar(x=mem_mcount.index,height=mem_mcount,color="olive",alpha=0.75)
plt.bar(x=casual_mcount.index,height=casual_mcount,color="m",alpha=0.7)
plt.ylabel("No Of Riders",color="brown",size=15)
plt.xlabel("Month",color="brown",size=15)
plt.title("No of Casual and Member Riders on different months",color="red",size=20,alpha
plt.legend(labels=["member","casual"],loc="upper right")
plt.show()
```



Above chart shows no of member riders is greater than casual riders in each month.5th, 6th, 7th, 8th, 9th month has more no of both casual and member riders than other months

In [64]:
```
mem_mcount.index
```

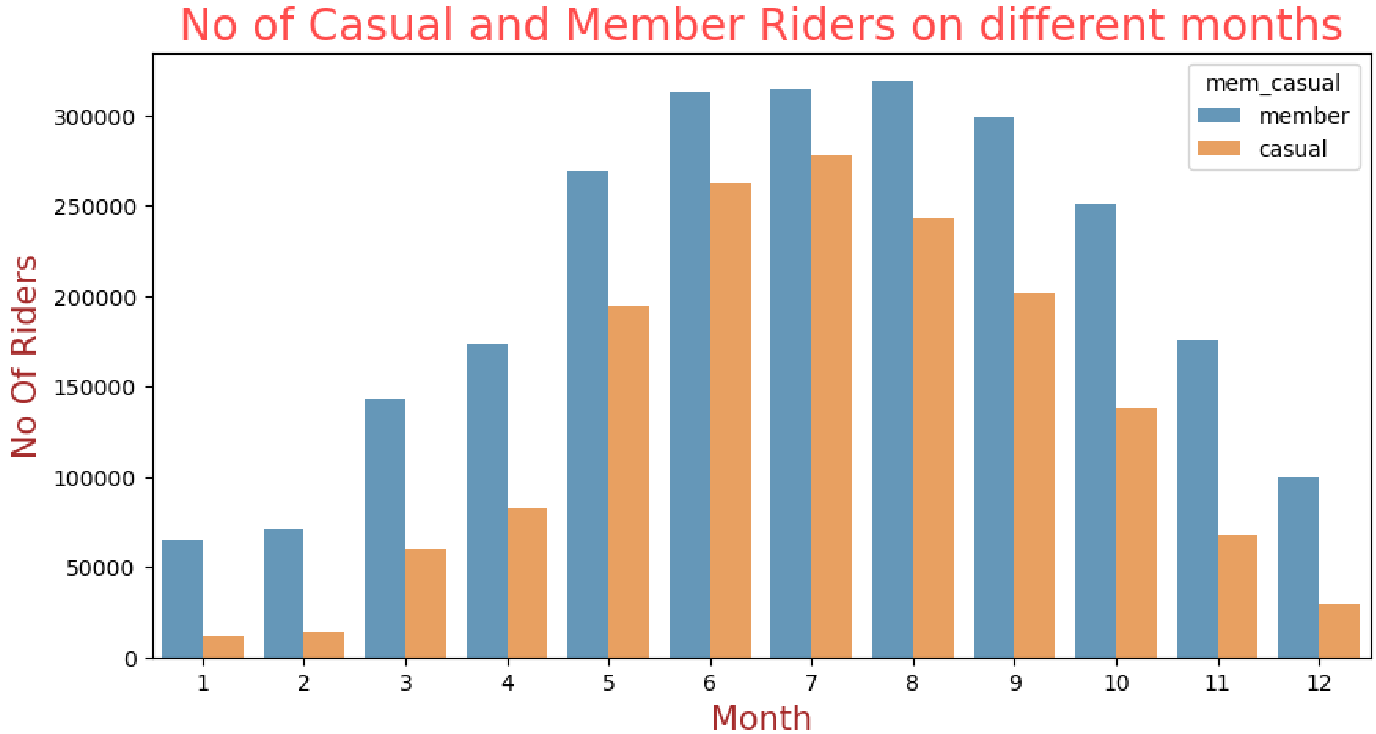
Out[64]:
```
Int64Index([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], dtype='int64', name='month')
```

In [65]:
```
df1=pd.DataFrame({"month":mem_mcount.index,"no of riders":mem_mcount.values})
df2=pd.DataFrame({"month":casual_mcount.index,"no of riders":casual_mcount.values})
df1["mem_casual"]="member"
df2["mem_casual"]="casual"
df=pd.concat([df1,df2],axis=0)
df.head()
```

Out[65]:

	month	no of riders	mem_casual
0	1	65085	member
1	2	71118	member
2	3	143082	member
3	4	173556	member
4	5	269034	member

In [66]:
```
plt.figure(figsize=(10,5))
sns.barplot(data=df,x="month",y="no of riders",hue="mem_casual",alpha=0.75)
plt.ylabel("No Of Riders",color="brown",size=15)
plt.xlabel("Month",color="brown",size=15)
plt.title("No of Casual and Member Riders on different months",color="red",size=20,alpha
plt.show()
```

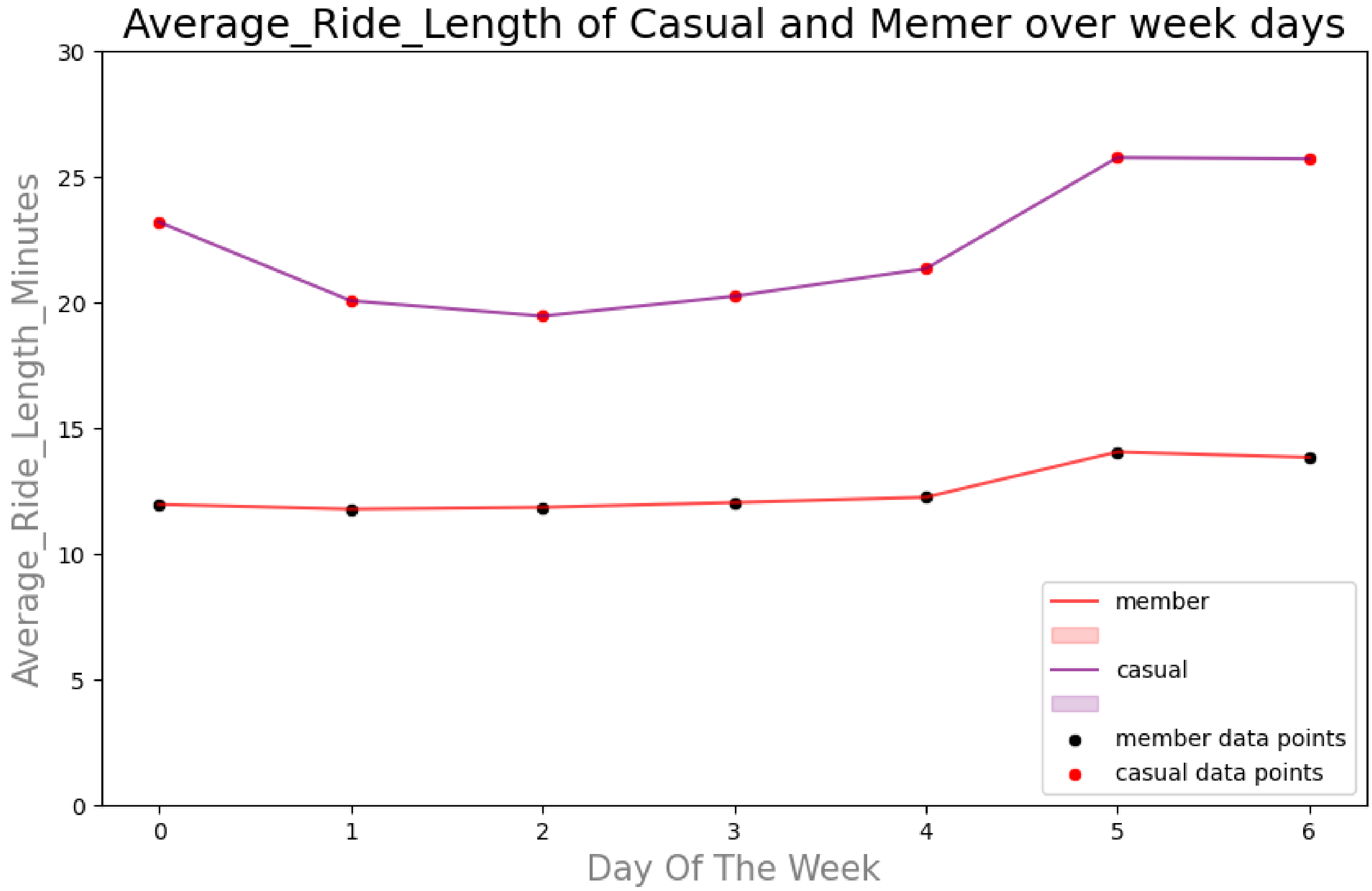


previous chart and this chart shows the same thing

In [67]:
```
mem_wd=annual2022_clean[annual2022_clean["member_casual"]=="member"].groupby("weekday")[
casual_wd=annual2022_clean[annual2022_clean["member_casual"]=="casual"].groupby("weekday
```

In [68]:
```
mem_wds=annual2022_clean[annual2022_clean["member_casual"]=="member"].groupby("weekday")
casual_wds=annual2022_clean[annual2022_clean["member_casual"]=="casual"].groupby("weekda
```

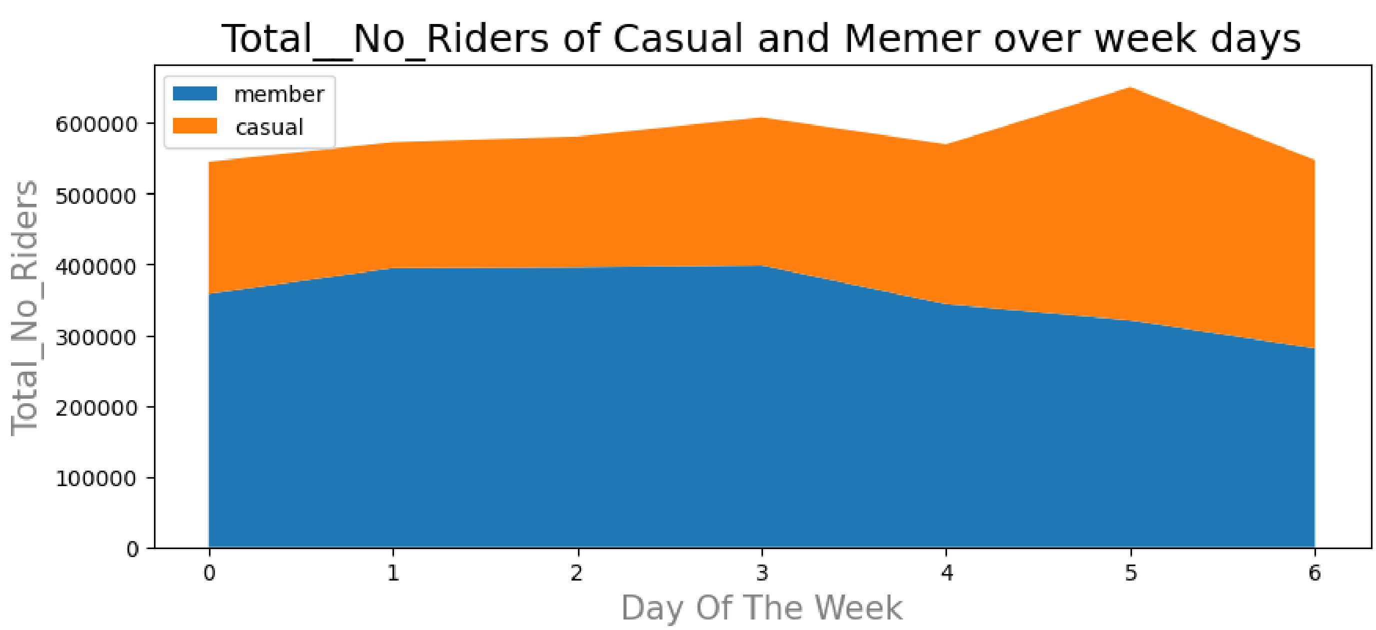
In [69]:
```
plt.figure(figsize=(10,6))
sns.lineplot(data=mem_wd,x=mem_wd.index,y=mem_wd.apply(to_minutes),color="red",alpha=0.7
sns.lineplot(data=casual_wd,x=casual_wd.index,y=casual_wd.apply(to_minutes),color="purpl
sns.scatterplot(data=mem_wd,x=mem_wd.index,y=mem_wd.apply(to_minutes),color="black")
sns.scatterplot(data=casual_wd,x=casual_wd.index,y=casual_wd.apply(to_minutes),color="re
plt.ylim(0,30)
plt.ylabel("Average_Ride_Length_Minutes",color="grey",size=15)
plt.xlabel("Day Of The Week",color="grey",size=15)
plt.title("Average_Ride_Length of Casual and Memer over week days",size=18,color='black'
plt.legend(["member","","casual","","member data points","casual data points"],loc="lowe
plt.show()
```



By saw this graph, average ride length of both casual and member riders in weekend, for more to casual riders

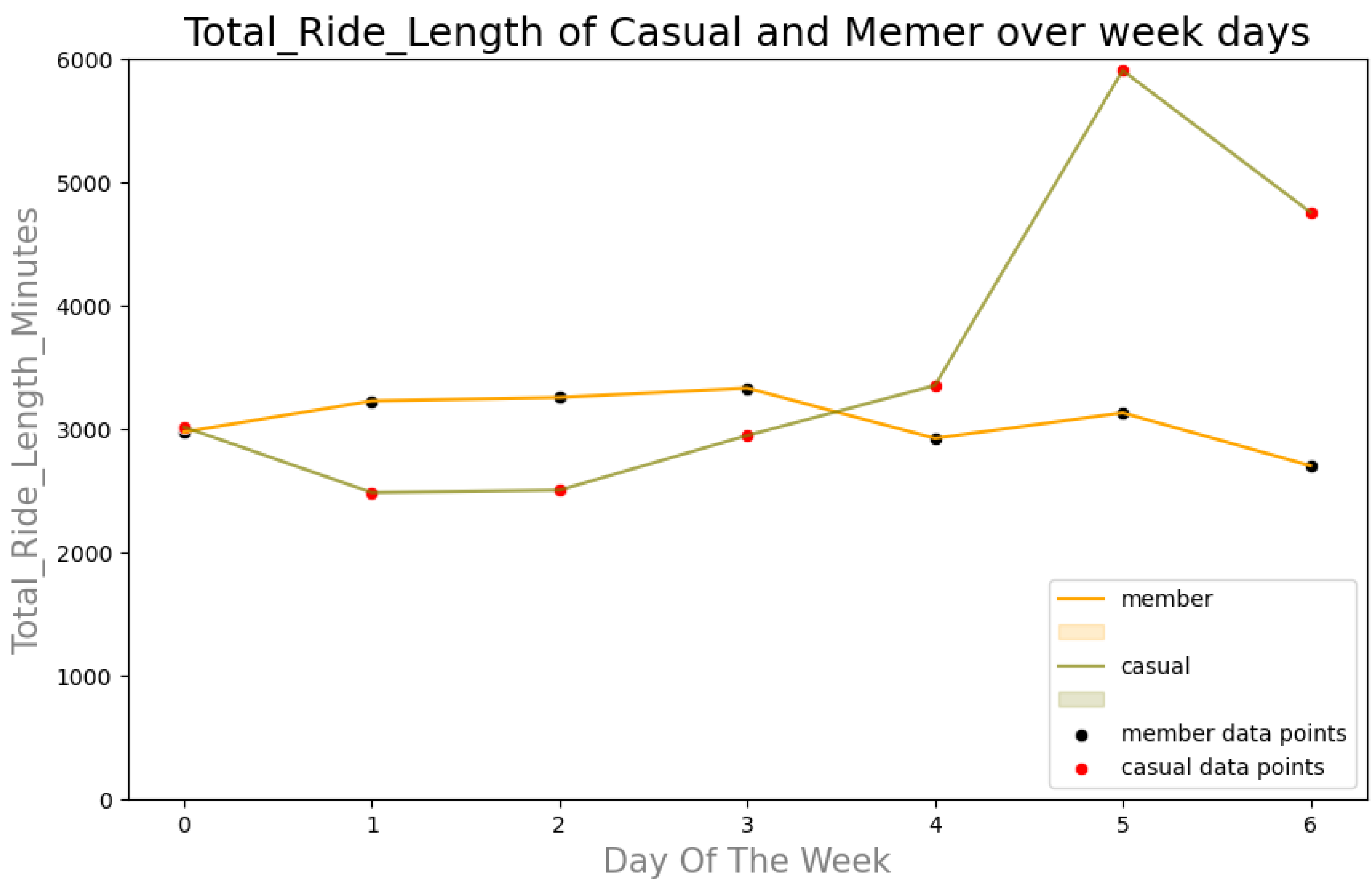
In [70]:
```
mem_wdc=annual2022_clean[annual2022_clean["member_casual"]=="member"].groupby("weekday")
casual_wdc=annual2022_clean[annual2022_clean["member_casual"]=="casual"].groupby("weekda
```

In [71]:
```
plt.figure(figsize=(10,4))
plt.stackplot(mem_wdc.index,mem_wdc,casual_wdc)
plt.ylabel("Total_No_Riders",color="grey",size=15)
plt.xlabel("Day Of The Week",color="grey",size=15)
plt.title("Total__No_Riders of Casual and Memer over week days",size=18,color='black')
plt.legend(["member","casual"],loc="upper left")
plt.show()
```



This chart shows, member riders are use bicyce in working days, but casual riders use bicylce mostly on weekend

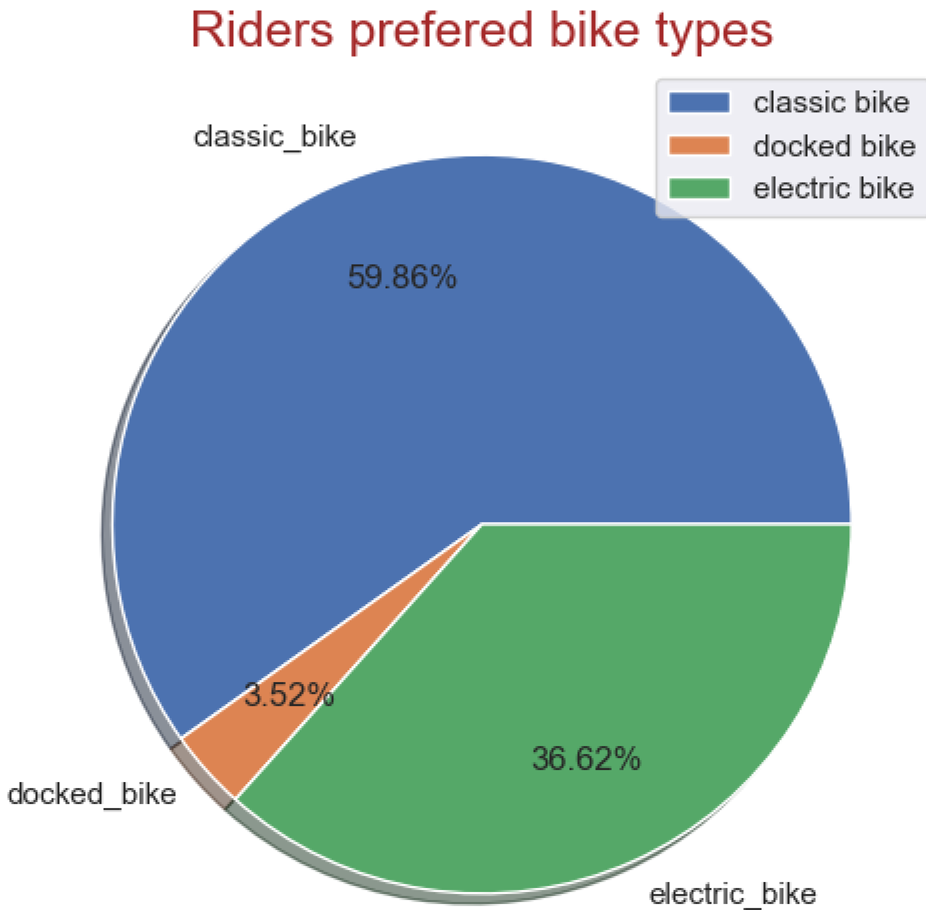
In [72]:
```
plt.figure(figsize=(10,6))
sns.lineplot(data=mem_wds,x=mem_wds.index,y=mem_wds.apply(to_days),color="orange",alpha=
sns.lineplot(data=casual_wds,x=casual_wds.index,y=casual_wds.apply(to_days),color="olive
sns.scatterplot(data=mem_wds,x=mem_wds.index,y=mem_wds.apply(to_days),color="black")
sns.scatterplot(data=casual_wds,x=casual_wds.index,y=casual_wds.apply(to_days),color="re
plt.ylim(0,6000)
plt.ylabel("Total_Ride_Length_Minutes",color="grey",size=15)
plt.xlabel("Day Of The Week",color="grey",size=15)
plt.title("Total_Ride_Length of Casual and Memer over week days",size=18,color='black')
plt.legend(["member","","casual","","member data points","casual data points"],loc="lowe
plt.show()
```



this also say same thing that member riders are use bicyce in working days, but casual riders use bicylce mostly on weekend in terms of ride length

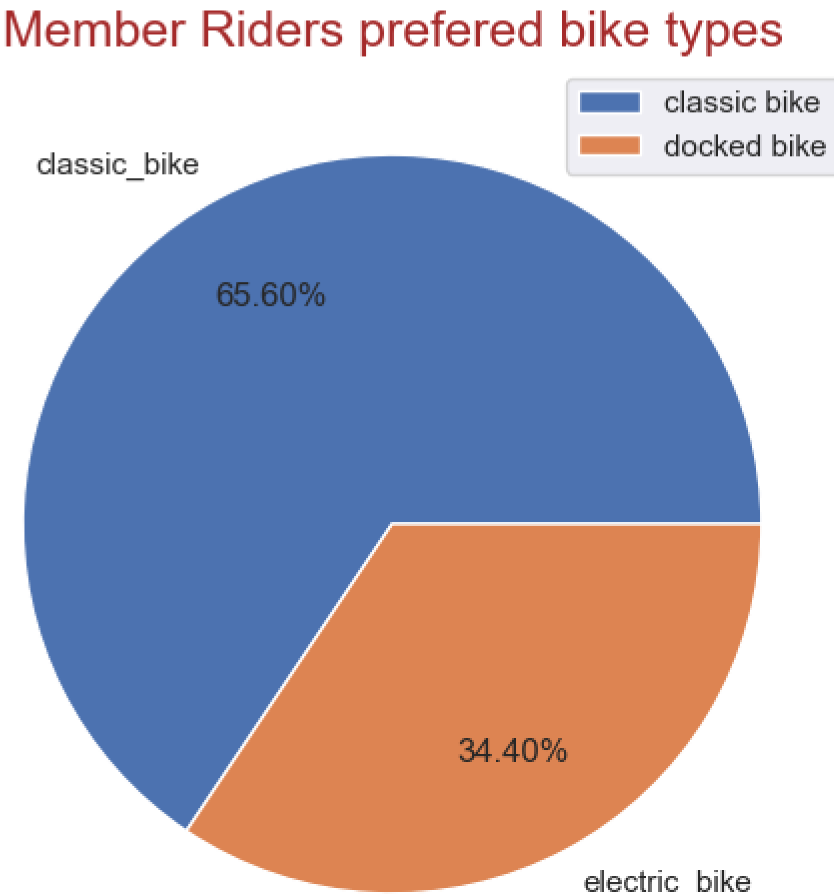
In [73]:
```
nrtype=annual2022_clean.groupby("rideable_type")["ride_id"].count()
nrtype_mem=annual2022_clean[annual2022_clean["member_casual"]=="member"].groupby("rideab
nrtype_casual=annual2022_clean[annual2022_clean["member_casual"]=="casual"].groupby("rid
```

In [74]:
```
sns.set_theme()
plt.figure(figsize=(6,6))
plt.title("Riders prefered bike types",size=18,color='brown')
plt.pie(nrtype,labels=nrtype.index,autopct='%.2f%%',pctdistance=0.7,shadow=True)
plt.legend(["classic bike","docked bike","electric bike"],loc="upper right")
plt.show()
```



This pie chart shows , most of the riders prefer classic bikes then others. Electric bikes got second most prefered

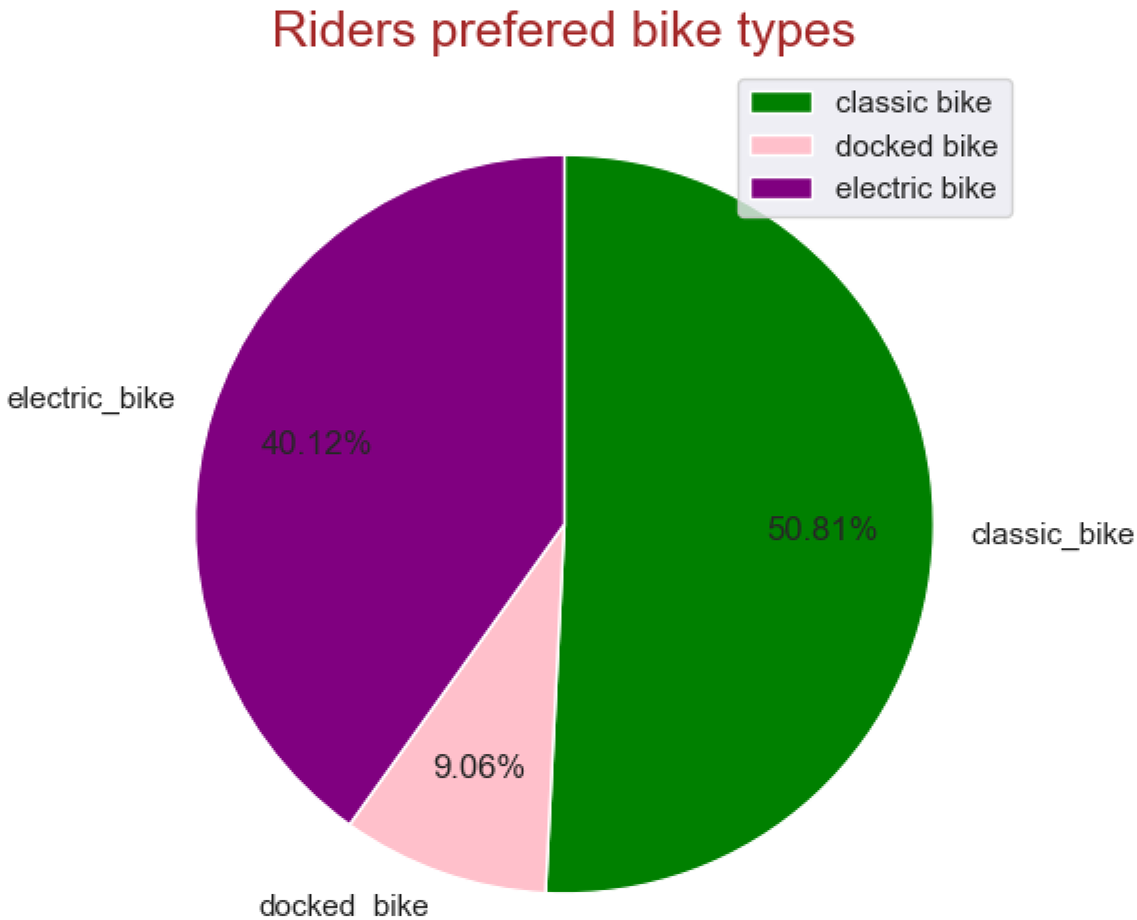
In [75]:
```
plt.figure(figsize=(6,6))
plt.title("Member Riders prefered bike types",size=18,color='brown')
plt.pie(nrtype_mem,labels=nrtype_mem.index,autopct='%.2f%%',pctdistance=0.7)
plt.legend(["classic bike","docked bike","electric bike"],loc="upper right")
plt.show()
```



This pie chart shows , most of the member riders prefer classic bikes and Electric bikes got second preference. They didn't like docked bikes

In [102…

```
plt.figure(figsize=(6,6))
plt.title("Riders prefered bike types",size=18,color='brown')
plt.pie(nrtype_casual,labels=nrtype_casual.index,autopct='%.2f%%',pctdistance=0.7,starta
plt.legend(["classic bike","docked bike","electric bike"],loc="upper right")
plt.show()
```

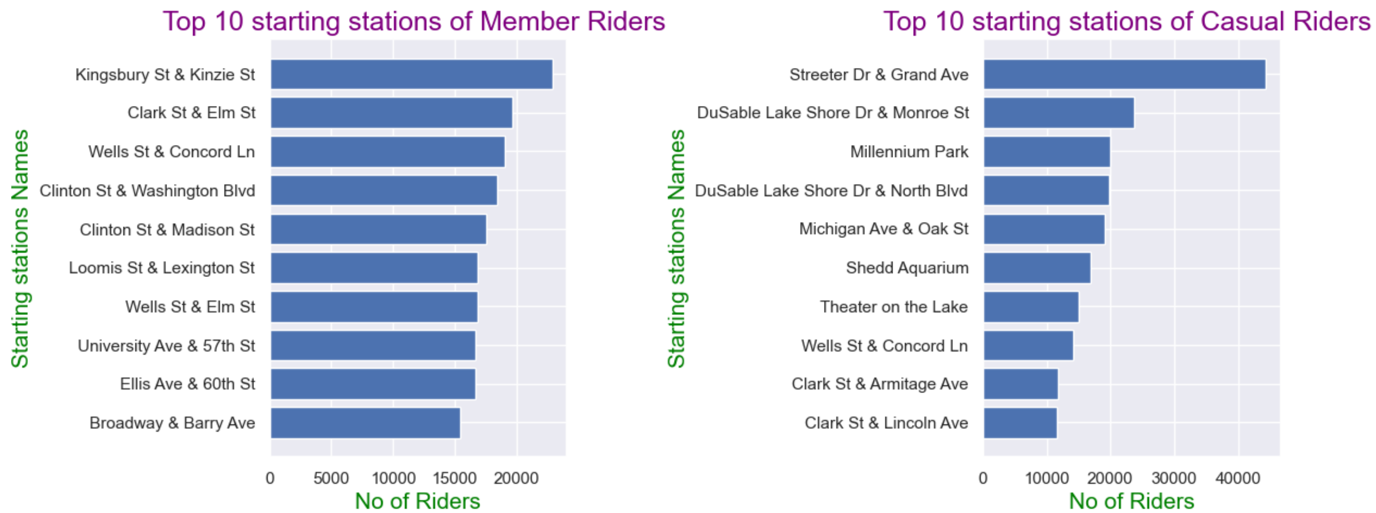


This pie chart shows , most of the casual riders prefer classic bikes and Electric bikes got second preference. but They like docked bikes also

In [81]:
```
mem_ss=annual2022_clean[annual2022_clean["member_casual"]=="member"]["start_station_name
casual_ss=annual2022_clean[annual2022_clean["member_casual"]=="casual"]["start_station_n
```

In [92]:
```
plt.figure(figsize=(12,5),)
plt.subplot(1,3,1)
plt.title("Top 10 starting stations of Member Riders ",size=18,color='purple')
plt.barh(mem_ss.sort_values().index,mem_ss.sort_values())
plt.ylabel("Starting stations Names",color="green",size=15)
plt.xlabel("No of Riders",color="green",size=15)
plt.subplot(1,3,3)
plt.title("Top 10 starting stations of Casual Riders ",size=18,color='purple')
plt.ylabel("Starting stations Names",color="green",size=15)
plt.xlabel("No of Riders",color="green",size=15)
plt.barh(casual_ss.sort_values().index,casual_ss.sort_values())
```

Out[92]:
```
<BarContainer object of 10 artists>
```



In [93]:

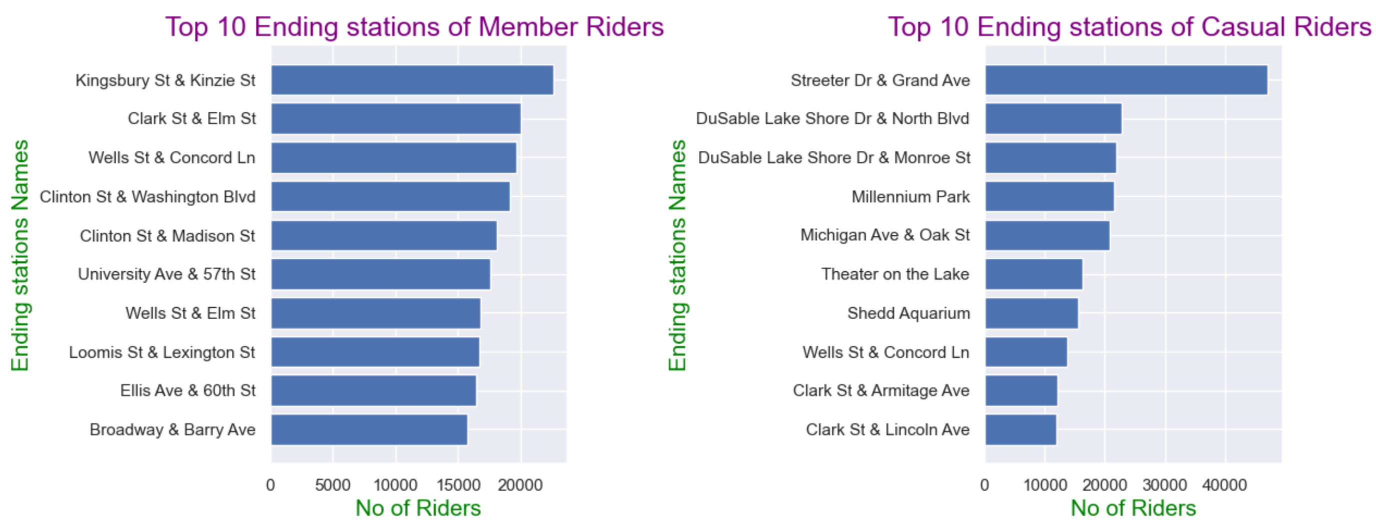
```
mem_es=annual2022_clean[annual2022_clean["member_casual"]=="member"]["end_station_name"]
casual_es=annual2022_clean[annual2022_clean["member_casual"]=="casual"]["end_station_nam
```

In [95]:

```
plt.figure(figsize=(12,5),)
plt.subplot(1,3,1)
plt.title("Top 10 Ending stations of Member Riders ",size=18,color='purple')
plt.barh(mem_es.sort_values().index,mem_es.sort_values())
plt.ylabel("Ending stations Names",color="green",size=15)
plt.xlabel("No of Riders",color="green",size=15)
plt.subplot(1,3,3)
plt.title("Top 10 Ending stations of Casual Riders ",size=18,color='purple')
plt.ylabel("Ending stations Names",color="green",size=15)
plt.xlabel("No of Riders",color="green",size=15)
plt.barh(casual_es.sort_values().index,casual_es.sort_values())
```

Out[95]:

```
<BarContainer object of 10 artists>
```



Key Findings

- There is no common starting and ending station between casual and member riders. This shows lack of service or poor service in casual riders location
- Most of the riders prefered classic bikes, docked and classic is used for long ride
- most of the casual riders use bicycle share in week days. This can also due to lack of services or user friendly serbices
- There is a seasonal changes in no of riders. 5th, 6th, 7th ,8th months attract more no of riders

- Dramatic drop trend in average ride length of casual riders shows existing of problem in services for casual riders

Recommandations

- Address solution for existing problems in service providing in all region and provide proper userfriendlt services evenly for all branches
- make available more no of classic bikes , particularly in week days
- Concerntrate more advertising and awareness on 5th, 6th, 7th, 8th months to covert riders from casual to member

THANK YOU ALL

In [ ]: