Google

Data Analytics Professional Certification Capstone Project

Case Study: Cyclistic bike share

How does a bike-share navigate speedy success?

Problem Statement

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

Cyclists: 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 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 tracked 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 at a time.

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. 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.

Phase 1 - Ask

To design marketing strategies aimed at converting casual riders into annual members, the goal is to determine

“how do annual members and casual riders use Cyclistic bikes differently?”
“How can we attract Potential Customers”

Phase 2 - Prepare

To analyze and identify trends, I'm going to use the previous 12 months of Cyclistic's historical trip data made available by Motivate International Inc.

The data is reliable as it was obtained directly from the company.

The files are in CSV format and have:

1. The unique identification information for each trip (primary key) "ride_id"
2. The type of transport used as "rideable_type"
3. The type of user (casual or member) is "member_casual"
4. The date and time of the start of the tour as "started_at"
5. The date and time of the end of the tour as "ended_at"
6. The name of the start station is "start_station_name"
7. The identification key of the start station is "start_station_id"
8. The name of the end station is "end_station_name"
9. The identification key of the end station is "end_station_id"
10. Geographic data (latitude and longitude) of the start-end stations as "start_lat", "start_lng", "end_lat" and "end_lng".

Phase 3 - Process

We are using the Pandas library for Data Processing. With Python, we can handle large amounts of data efficiently and quickly.

Importing the Pandas library:

```
import pandas as pd
```

Reading the files and defining variables:

```
df_2022_02 = pd.read_csv('202202.csv')
df_2022_03 = pd.read_csv('202203.csv')
df_2022_04 = pd.read_csv('202204.csv')
df_2022_05 = pd.read_csv('202205.csv')
df_2022_06 = pd.read_csv('202206.csv')
df_2022_07 = pd.read_csv('202207.csv')
df_2022_08 = pd.read_csv('202208.csv')
df_2022_09 = pd.read_csv('202209.csv')
df_2022_10 = pd.read_csv('202210.csv')
df_2022_11 = pd.read_csv('202211.csv')
df_2022_12 = pd.read_csv('202212.csv')
df_2023_01 = pd.read_csv('202301.csv')
```

Checking the structure and formatting of data

frames:

Input -

```
#  Display information from the DataFrame
df_2022_02.info()
```

Output-

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115609 entries, 0 to 115608
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   ride_id             115609 non-null  object 
 1   rideable_type       115609 non-null  object 
 2   started_at          115609 non-null  object 
 3   ended_at            115609 non-null  object 
 4   start_station_name  97029 non-null   object 
 5   start_station_id    97029 non-null   object 
 6   end_station_name    95254 non-null   object 
 7   end_station_id      95254 non-null   object 
 8   start_lat           115609 non-null  float64
 9   start_lng           115609 non-null  float64
 10  end_lat             115532 non-null  float64
 11  end_lng             115532 non-null  float64
 12  member_casual       115609 non-null  object 
dtypes: float64(4), object(9)
memory usage: 11.5+ MB
```

Some inconsistencies are noted:

1 - The time attributes "started_at" and "ended_at" are in string format.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115609 entries, 0 to 115608
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   ride_id             115609 non-null  object
 1   rideable_type       115609 non-null  object
 2   started_at          115609 non-null  object
 3   ended_at            115609 non-null  object
 4   start_station_name  97029 non-null   object
 5   start_station_id    97029 non-null   object
 6   end_station_name    95254 non-null   object
 7   end_station_id      95254 non-null   object
 8   start_lat           115609 non-null  float64
 9   start_lng           115609 non-null  float64
 10  end_lat             115532 non-null  float64
 11  end_lng             115532 non-null  float64
 12  member_casual       115609 non-null  object
dtypes: float64(4), object(9)
memory usage: 11.5+ MB
```

2. Null values are shown in the attributes "start_station_name", "start_station_id", "end_station_name", "end_station_id", "end_lat" and "end_lng".

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115609 entries, 0 to 115608
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   ride_id             115609 non-null  object
 1   rideable_type       115609 non-null  object
 2   started_at          115609 non-null  object
 3   ended_at            115609 non-null  object
 4   start_station_name  97029 non-null   object
 5   start_station_id    97029 non-null   object
 6   end_station_name    95254 non-null   object
 7   end_station_id      95254 non-null   object
 8   start_lat           115609 non-null  float64
 9   start_lng           115609 non-null  float64
 10  end_lat             115532 non-null  float64
 11  end_lng             115532 non-null  float64
 12  member_casual       115609 non-null  object
dtypes: float64(4), object(9)
memory usage: 11.5+ MB
```

Checking Other Dataframes:

'df_2022_03'

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284042 entries, 0 to 284041
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   ride_id             284042 non-null  object
 1   rideable_type       284042 non-null  object
 2   started_at          284042 non-null  object
 3   ended_at            284042 non-null  object
 4   start_station_name  236796 non-null  object
 5   start_station_id    236796 non-null  object
 6   end_station_name    232885 non-null  object
 7   end_station_id      232885 non-null  object
 8   start_lat           284042 non-null  float64
 9   start_lng           284042 non-null  float64
 10  end_lat             283776 non-null  float64
 11  end_lng             283776 non-null  float64
 12  member_casual       284042 non-null  object
dtypes: float64(4), object(9)
memory usage: 28.2+ MB
```

'df_2022_04'

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 371249 entries, 0 to 371248
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   ride_id             371249 non-null  object
 1   rideable_type       371249 non-null  object
 2   started_at          371249 non-null  object
 3   ended_at            371249 non-null  object
 4   start_station_name  300362 non-null  object
 5   start_station_id    300362 non-null  object
 6   end_station_name    295961 non-null  object
 7   end_station_id      295961 non-null  object
 8   start_lat           371249 non-null  float64
 9   start_lng           371249 non-null  float64
 10  end_lat             370932 non-null  float64
 11  end_lng             370932 non-null  float64
 12  member_casual       371249 non-null  object
dtypes: float64(4), object(9)
memory usage: 36.8+ MB
```

'df_2022_05'

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 634858 entries, 0 to 634857
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   ride_id             634858 non-null  object 
 1   rideable_type       634858 non-null  object 
 2   started_at          634858 non-null  object 
 3   ended_at            634858 non-null  object 
 4   start_station_name  548154 non-null  object 
 5   start_station_id    548154 non-null  object 
 6   end_station_name    541687 non-null  object 
 7   end_station_id      541687 non-null  object 
 8   start_lat           634858 non-null  float64
 9   start_lng           634858 non-null  float64
 10  end_lat             634136 non-null  float64
 11  end_lng             634136 non-null  float64
 12  member_casual       634858 non-null  object 
dtypes: float64(4), object(9)
memory usage: 63.0+ MB
```

‘df_2022_06’

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 769204 entries, 0 to 769203
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   ride_id             769204 non-null  object 
 1   rideable_type       769204 non-null  object 
 2   started_at          769204 non-null  object 
 3   ended_at            769204 non-null  object 
 4   start_station_name  676260 non-null  object 
 5   start_station_id    676260 non-null  object 
 6   end_station_name    669052 non-null  object 
 7   end_station_id      669052 non-null  object 
 8   start_lat           769204 non-null  float64
 9   start_lng           769204 non-null  float64
 10  end_lat             768149 non-null  float64
 11  end_lng             768149 non-null  float64
 12  member_casual       769204 non-null  object 
dtypes: float64(4), object(9)
memory usage: 76.3+ MB
```

‘df_2022_07’

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 823488 entries, 0 to 823487
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   ride_id             823488 non-null  object 
 1   rideable_type       823488 non-null  object 
 2   started_at          823488 non-null  object 
 3   ended_at            823488 non-null  object 
 4   start_station_name  711457 non-null  object 
 5   start_station_id    711457 non-null  object 
 6   end_station_name    702537 non-null  object 
 7   end_station_id      702537 non-null  object 
 8   start_lat           823488 non-null  float64
 9   start_lng           823488 non-null  float64
 10  end_lat             822541 non-null  float64
 11  end_lng             822541 non-null  float64
 12  member_casual       823488 non-null  object 
dtypes: float64(4), object(9)
memory usage: 81.7+ MB
```

‘df_2022_08’

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 785932 entries, 0 to 785931
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   ride_id             785932 non-null  object 
 1   rideable_type       785932 non-null  object 
 2   started_at          785932 non-null  object 
 3   ended_at            785932 non-null  object 
 4   start_station_name  673895 non-null  object 
 5   start_station_id    673895 non-null  object 
 6   end_station_name    665410 non-null  object 
 7   end_station_id      665410 non-null  object 
 8   start_lat           785932 non-null  float64
 9   start_lng           785932 non-null  float64
 10  end_lat             785089 non-null  float64
 11  end_lng             785089 non-null  float64
 12  member_casual       785932 non-null  object 
dtypes: float64(4), object(9)
memory usage: 78.0+ MB
```

‘df_2022_09’

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 701339 entries, 0 to 701338
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   ride_id             701339 non-null  object 
 1   rideable_type       701339 non-null  object 
 2   started_at          701339 non-null  object 
 3   ended_at            701339 non-null  object 
 4   start_station_name  597559 non-null  object 
 5   start_station_id    597559 non-null  object 
 6   end_station_name    590154 non-null  object 
 7   end_station_id      590154 non-null  object 
 8   start_lat           701339 non-null  float64
 9   start_lng           701339 non-null  float64
 10  end_lat             700627 non-null  float64
 11  end_lng             700627 non-null  float64
 12  member_casual       701339 non-null  object 
dtypes: float64(4), object(9)
memory usage: 69.6+ MB
```

'df_2022_10'

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 558685 entries, 0 to 558684
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   ride_id             558685 non-null  object 
 1   rideable_type       558685 non-null  object 
 2   started_at          558685 non-null  object 
 3   ended_at            558685 non-null  object 
 4   start_station_name  467330 non-null  object 
 5   start_station_id    467330 non-null  object 
 6   end_station_name    462068 non-null  object 
 7   end_station_id      462068 non-null  object 
 8   start_lat           558685 non-null  float64
 9   start_lng           558685 non-null  float64
 10  end_lat             558210 non-null  float64
 11  end_lng             558210 non-null  float64
 12  member_casual       558685 non-null  object 
dtypes: float64(4), object(9)
memory usage: 55.4+ MB
```

'df_2022_11'

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 337735 entries, 0 to 337734
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   ride_id             337735 non-null  object 
 1   rideable_type       337735 non-null  object 
 2   started_at          337735 non-null  object 
 3   ended_at            337735 non-null  object 
 4   start_station_name  285778 non-null  object 
 5   start_station_id    285778 non-null  object 
 6   end_station_name    283476 non-null  object 
 7   end_station_id      283476 non-null  object 
 8   start_lat           337735 non-null  float64
 9   start_lng           337735 non-null  float64
 10  end_lat             337505 non-null  float64
 11  end_lng             337505 non-null  float64
 12  member_casual       337735 non-null  object 
dtypes: float64(4), object(9)
memory usage: 33.5+ MB
```

‘df_2022_12’

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181806 entries, 0 to 181805
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   ride_id             181806 non-null  object 
 1   rideable_type       181806 non-null  object 
 2   started_at          181806 non-null  object 
 3   ended_at            181806 non-null  object 
 4   start_station_name  152523 non-null  object 
 5   start_station_id    152523 non-null  object 
 6   end_station_name    150648 non-null  object 
 7   end_station_id      150648 non-null  object 
 8   start_lat           181806 non-null  float64
 9   start_lng           181806 non-null  float64
 10  end_lat             181678 non-null  float64
 11  end_lng             181678 non-null  float64
 12  member_casual       181806 non-null  object 
dtypes: float64(4), object(9)
memory usage: 18.0+ MB
```

‘df_2023_01’

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 190301 entries, 0 to 190300
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   ride_id             190301 non-null  object
 1   rideable_type       190301 non-null  object
 2   started_at          190301 non-null  object
 3   ended_at            190301 non-null  object
 4   start_station_name  163580 non-null  object
 5   start_station_id    163580 non-null  object
 6   end_station_name    162461 non-null  object
 7   end_station_id      162461 non-null  object
 8   start_lat           190301 non-null  float64
 9   start_lng           190301 non-null  float64
 10  end_lat             190174 non-null  float64
 11  end_lng             190174 non-null  float64
 12  member_casual       190301 non-null  object
dtypes: float64(4), object(9)
memory usage: 18.9+ MB
```

All tables have the same structure and formatting standards. The inconsistencies found in the "df_2022_02" data frame are repeated in the others.

As the structure is the same, we can proceed by consolidating the tables into a single data frame.

Consolidating tables into a single data frame

```
df_tripdata = pd.concat([df_2022_02, df_2022_03,df_2022_04,df_2022_05,
                         df_2022_06, df_2022_07, df_2022_08, df_2022_09, df_2022_10,
                         df_2022_11, df_2022_12, df_2023_01], axis= 0)
```

Adjust the date/time attributes:

```
2    started_at          object
3    ended_at            object
```

```
df_tripdata['started_at']= pd.to_datetime(df_tripdata['started_at'], format = '%Y-%m-%d %H:%M:%S')
df_tripdata['ended_at']= pd.to_datetime(df_tripdata['ended_at'], format = '%Y-%m-%d %H:%M:%S')
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5754248 entries, 0 to 190300
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: 614.6+ MB
```

calculating the duration of each trip:

```
# Add column 'duration'
df_tripdata['duration']= df_tripdata['ended_at']- df_tripdata['started_at']
```

Adding a day of the week corresponding to the trip date column:

```
# Add column 'day_of_week'
df_tripdata['day_of_week']= df_tripdata['started_at'].dt.day_name()
```

Searching for duplicate values in the "ride_id" key

Input-

```
duplicates = df_tripdata['ride_id'].duplicated()
if duplicates.any():
    print('There are Duplicate values')
else:
    print('There are no duplicate values')
```

Output-

```
There are no duplicate values
```

There are no Duplicate values

Anomalies in the "duration" attribute:

Input -

```
# Anomalies in 'duration' column
df_tripdata['duration'].describe()
```

Output-

```
count                          5754248
mean      0 days 00:19:18.334898148
std       0 days 02:55:19.871287872
min                 -8 days +19:26:39
25%                   0 days 00:05:46
50%                   0 days 00:10:12
75%                   0 days 00:18:20
max                  28 days 17:47:15
Name: duration, dtype: object
```

Negative time values are anomalies
Counting negative or zero values to determine whether the absence of such information may impact future analysis results:
Input -

```
# Counting Negative Duration Values
count_neg_values = (df_tripdata['duration']<= pd.Timedelta(0)).sum()
```

```
print(f'{(count_neg_values/len(df_tripdata) )*100 }%')
```

Output-

```
0.009280100544849647%
```

There are 534 invalid values out of 5754248 records i.e. 0.009280100544849647%.
Eliminating rows with negative or zero values in the 'duration column'

Eliminating these records:
Input-

```
# Eliminating Negative Duration Values
df_tripdata = df_tripdata.loc[df_tripdata['duration'] >= pd.Timedelta(0)]
```

Output-

```
count                        5754148
mean      0 days 00:19:18.492221611
std       0 days 02:55:16.751992415
min                  0 days 00:00:00
25%                  0 days 00:05:46
50%                  0 days 00:10:12
75%                  0 days 00:18:20
max                 28 days 17:47:15
Name: duration, dtype: object
```

Values with duration above normal
Amount of trips equal to or greater than a day:
Input-

```
# Counting data where duration is more than one day
count_dur_greater_than_one = (df_tripdata['duration']>= pd.Timedelta(days= 1)).sum()
printDD(f'There are {count_dur_greater_than_one} values that are greater than 1 i.e {(count_dur_greater_than_one/len(df_tripdata)
```

Output-

```
There are 5390 values that are greater than 1 i.e 0.09367155658839502%
```

Eliminating these records for further future analysis
Input-

```
#Eliminating values where duration is more than one day
df_tripdata= df_tripdata.loc[df_tripdata['duration']<pd.Timedelta(days= 1)]
```

Output-

```
count                        5748758
mean      0 days 00:16:09.102077353
std       0 days 00:29:10.903454616
min                  0 days 00:00:00
25%                  0 days 00:05:46
50%                  0 days 00:10:12
75%                  0 days 00:18:18
max                  0 days 23:59:56
Name: duration, dtype: object
```

Searching null values

```python
for attributes in df_tripdata:
    nulls = df_tripdata[attributes].isnull().sum()
    print(f'There are {nulls} null values in {attributes}')
```

```
There are 0 null values in ride_id
There are 0 null values in rideable_type
There are 0 null values in started_at
There are 0 null values in ended_at
There are 843502 null values in start_station_name
There are 843502 null values in start_station_id
There are 897393 null values in end_station_name
There are 897393 null values in end_station_id
There are 0 null values in start_lat
There are 0 null values in start_lng
There are 703 null values in end_lat
There are 703 null values in end_lng
There are 0 null values in member_casual
There are 0 null values in duration
There are 0 null values in day_of_week
```

The Attributes start_station_id and end_station_id have null values. Deleting these records will result in approx 15.61% of the data frame.
We will choose to do the first analyses with the full DataFrame and clean up the null values for the identifications of the stations at the time this information is for analysis
With everything organized, we are ready to start the analysis.

Phase 4 - Analyze

We have analyzed the data from four major plots

1. Proportion of Users
2. Trip Count by Day of Week and Member Type
3. Count of trips started per month
4. Mean Trip Duration by Month and User Type
5. Top Stations by Casual Users

For data visualization, import the "matplotlib" library:

```
# For Data Visualization we will use 'matplotlib' library
import matplotlib.pyplot as plt
```

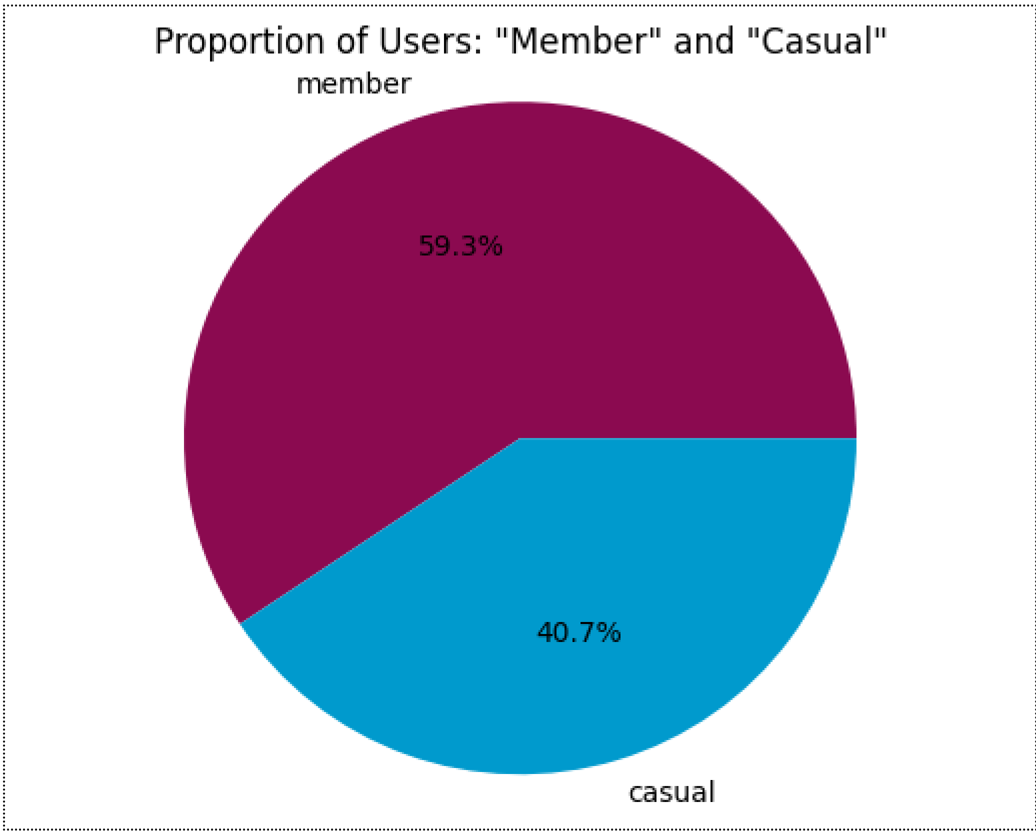
1. The proportion of Users

The ratio of casual users to members in the year gives an overview of the amount of potential new members. (pie chart)

```
# count of users in each category
count_users = df_tripdata['member_casual'].value_counts()
```

```
# Creating pie Chart
colors = ['#8B0A50', '#009ACD']
count_users.plot(kind='pie', autopct='%1.1f%%', colors=colors)
# set title and display chart
plt.title('Proportion of Users: "Member" and "Casual"')
plt.axis('equal')
plt.axis('off')
plt.show()
```

Plot-



- 59.3% of users are already members. 40.7% of users remain for a possible conversion. We need to understand the behavior of these 40.7% users so that we can convert them to Members.\

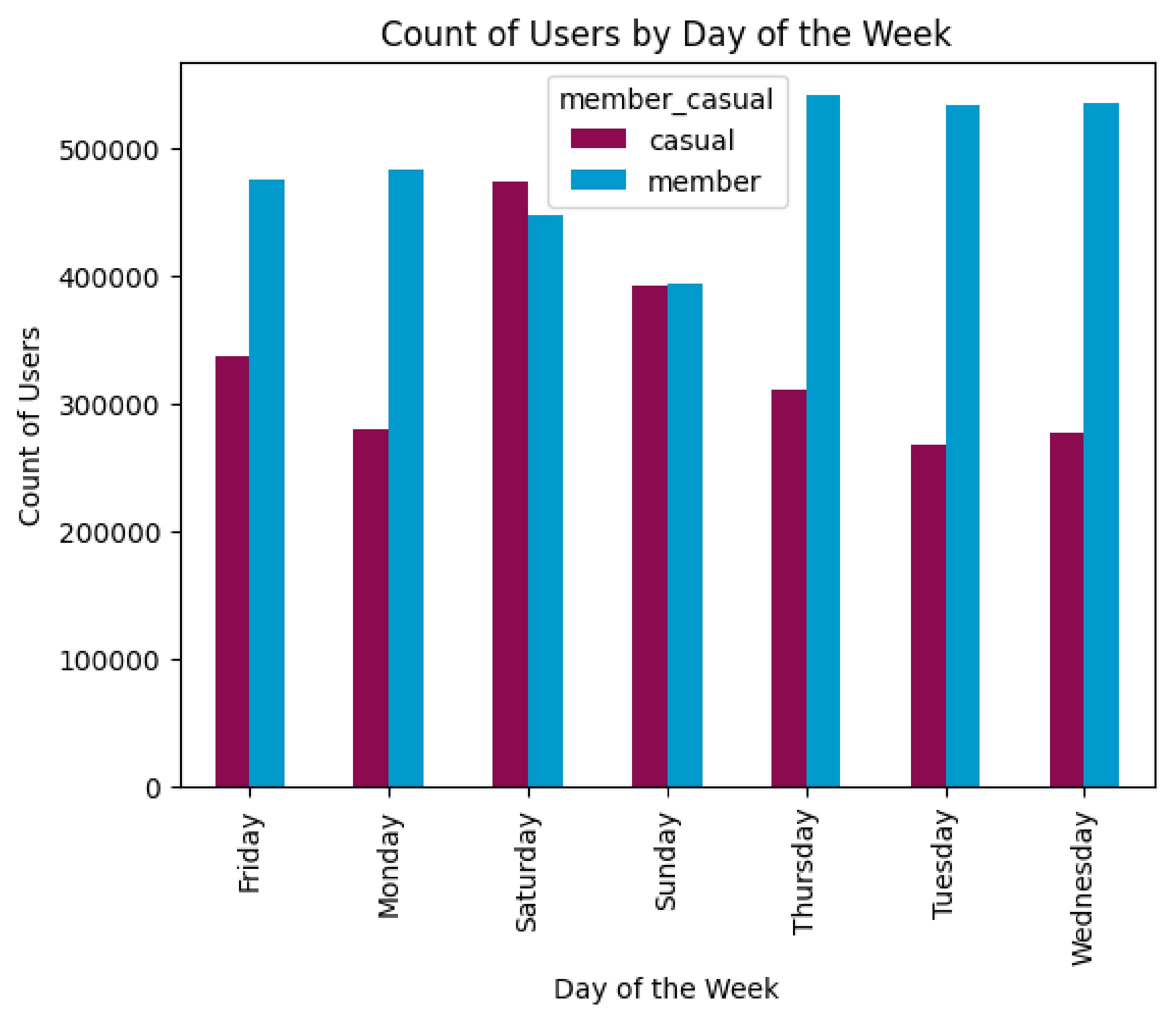
2. Trip Count by Day of Week and Member Type

```
# count of users in each day of week
count_users = df_tripdata.groupby([df_tripdata['day_of_week'],'member_casual'])['day_of_week'].count()
```

```
# Bar Chart

colors = ['#8B0A50', '#009ACD']
count_users.unstack().plot(kind='bar', color=colors)
plt.title('Count of Users by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Count of Users')
```

Plot -

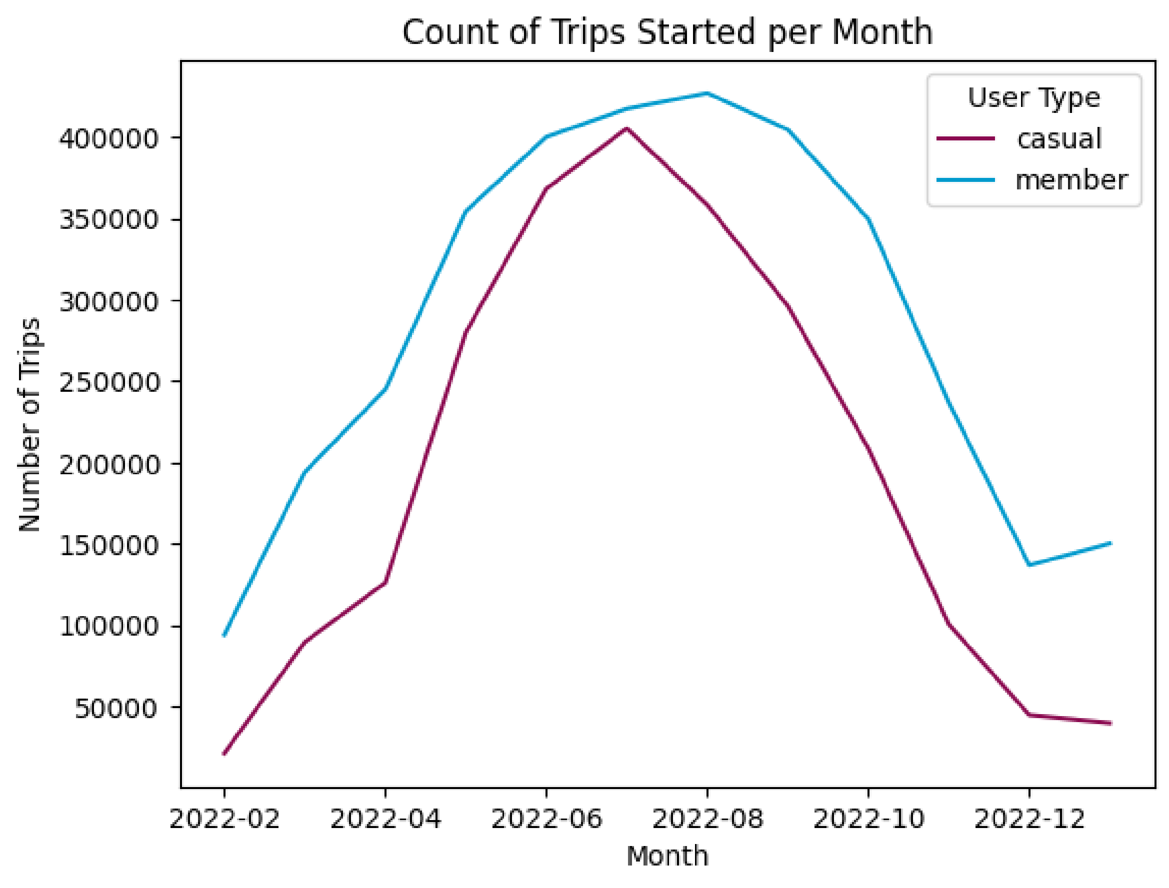


- Weekends are the best days to reach casual users.

3. Count of trips started per month

The variations in use among the year (seasonality) and the correlations between casual users and members can bring relevant information regarding the behavior of users at different times of the year.

```
# create the line chart
# group by month and count occurrences for each user category
count_tspm = df_tripdata.groupby([df_tripdata['started_at'].dt.strftime('%Y-%m'),'member_casual'])['ride_id'].count()
colors = ['#8B0A50', '#009ACD']
count_tspm.unstack().plot(kind='line', color=colors)
#Title and labels
plt.title('Count of Trips Started per Month')
plt.xlabel('Month')
plt.ylabel('Number of Trips')
plt.legend(title='User Type')
plt.show()
```

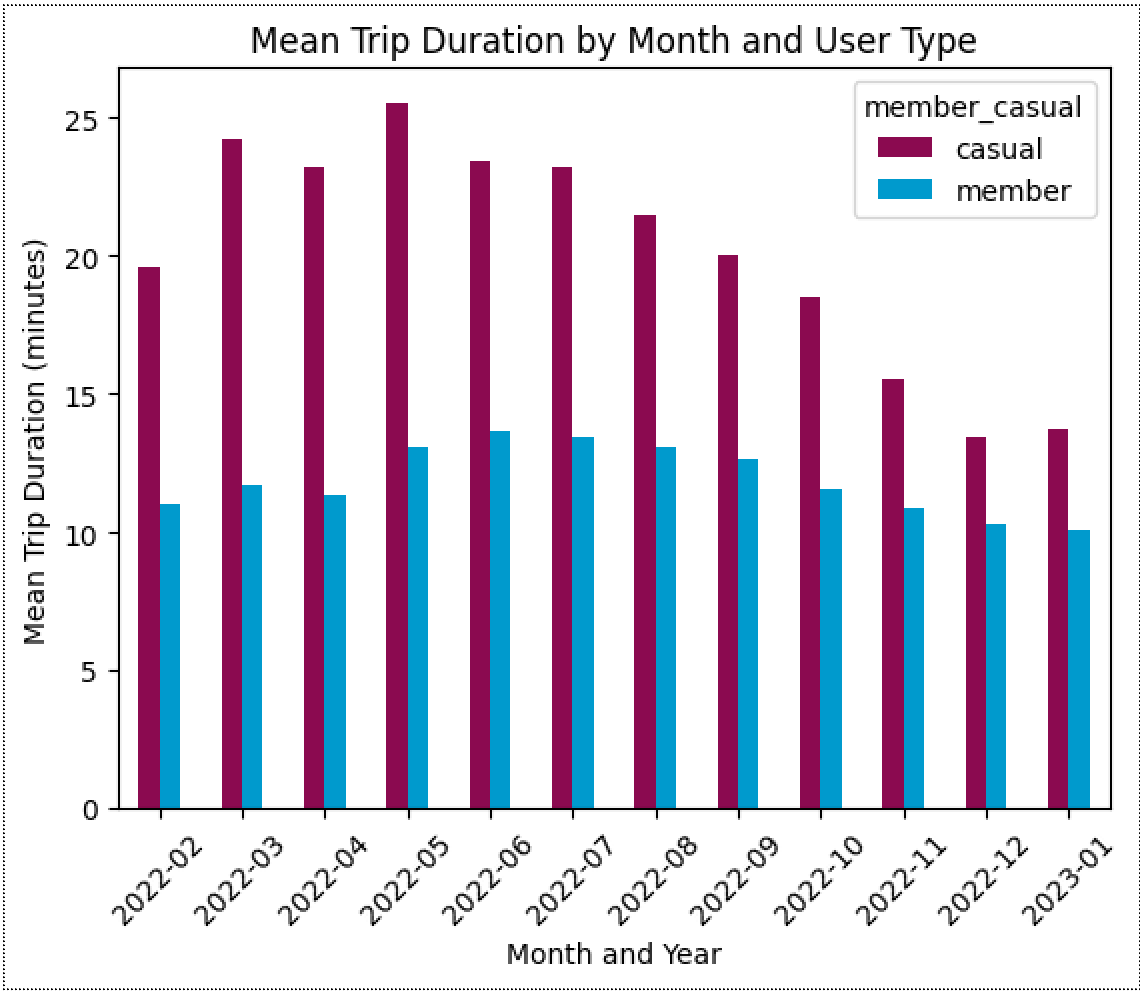


- The number of users varies significantly during the year. The service is used significantly more by subscribers and casual users in the summer. While the total number of members stayed higher throughout the year, there was a notable increase in the number of infrequent users nearly equaling the number of active users. This may indicate a user profile that includes visitors or those on vacation. Summertime is a better time to run a marketing campaign to turn non-members into members, especially in June and July.

4. Mean Trip Duration by Month and User Type

```
# Mean Trip Duration by Month and User Type
mean_tspm =df_tripdata.groupby([df_tripdata['started_at'].dt.strftime('%Y-%m'),'member_casual'])['duration'].mean().dt.total_secc
```

```
colors = ['#8B0A50', '#009ACD']
mean_dur_user = mean_tspm.unstack().plot(kind='bar', rot=45,
color=colors)
plt.xlabel('Month and Year')
plt.ylabel('Mean Trip Duration (minutes)')
plt.title('Mean Trip Duration by Month and User Type')
```



- It is evident that casual riders often ride longer. Members ride the bikes for shorter amounts of time.
  This may indicate that casual users prefer to ride around and enjoy the day with the service, whereas members may utilize it more frequently as a means of transportation (e.g., traveling to work from home).
  However, that's only a supposition that has to be verified by surveys. We will continue analyzing the data we currently have for the time being.

5. Top Stations by Casual Users

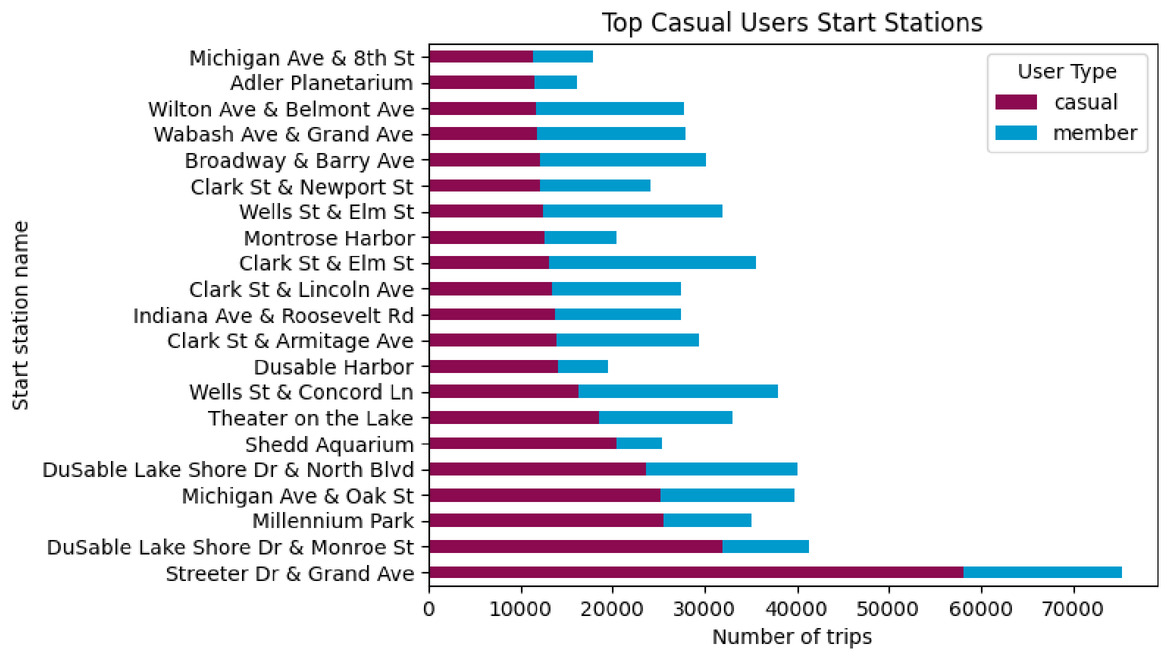
```
# eliminate rows with null values in column 'member_casual'
df_clean_member_casual = df_tripdata.dropna(subset=['member_casual'])
```

Top Start Start Stations

```
df_grouped_start_station = df_clean_member_casual.groupby(['start_station_name','member_casual']).size().unstack()
```

```
df_grouped_start_station = df_grouped_start_station.sort_values(by='casual',
ascending=False).head(21)
```

```
colors = ['#8B0A50', '#009ACD']
df_grouped_start_station.plot(kind='barh', stacked=True, color=colors)
plt.title('Top Casual Users Start Stations')
plt.xlabel('Number of trips')
plt.ylabel('Start station name')
plt.legend(title='User Type')
plt.show()
```



- The most popular beginning location for casual users is "Streeter Dr & Grand Ave" station. It's fascinating to observe the ratios between the various user types in addition to the amount. Certain stations are utilized less frequently overall, yet they have a very sizable segment of casual users. To improve comprehension of the target audience and marketing strategies, all of these stations ought to be taken into account.

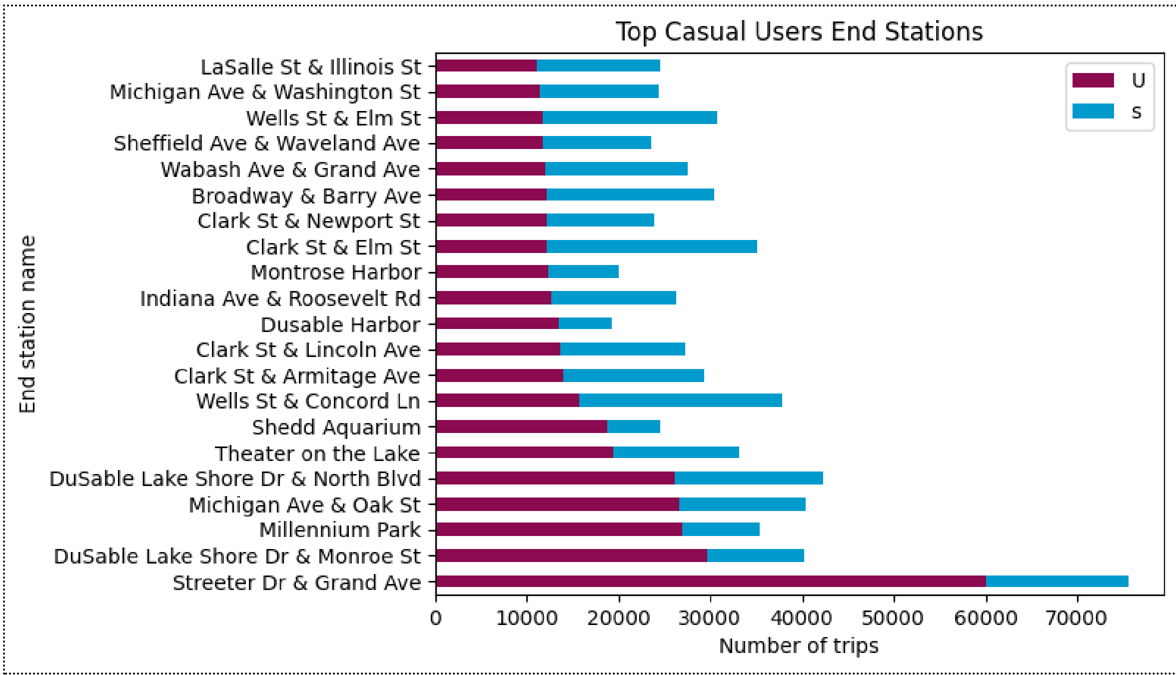
Top End Stations

```
df_grouped_end_station = df_clean_member_casual.groupby(['end_station_name','member_casual']).size().unstack()
```

```
# select only the top 21 casual type counts
df_grouped_start_station = df_grouped_start_station.sort_values(by='casual',
ascending=False).head(21)
```

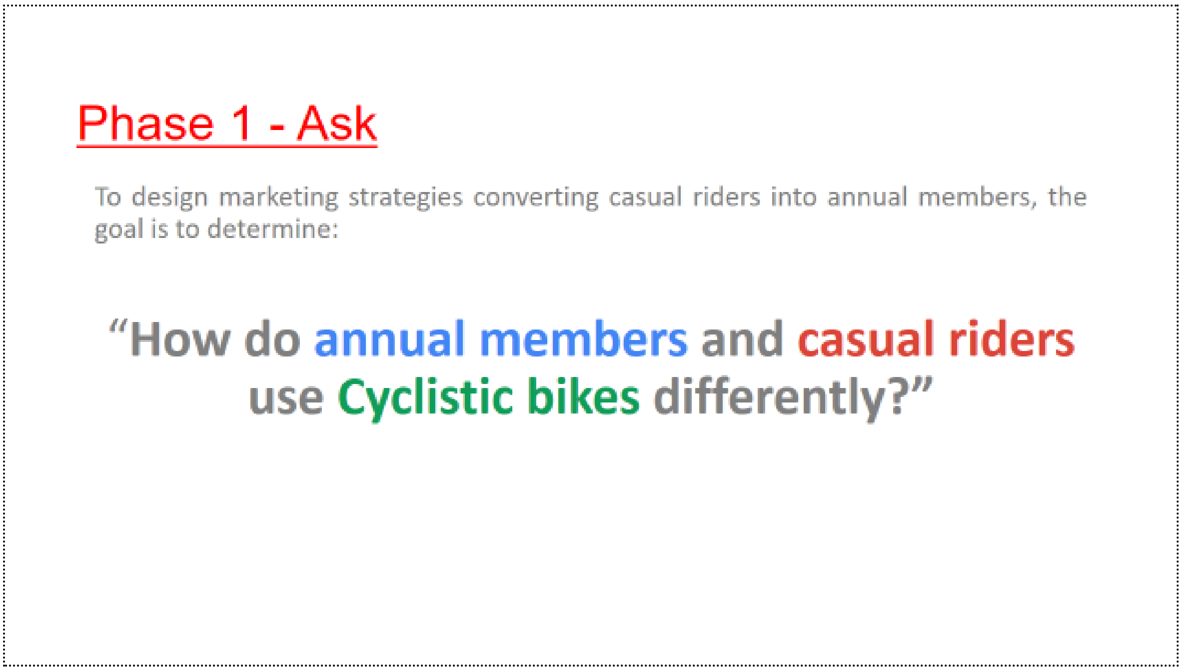
```
df_grouped_end_station = df_grouped_end_station.sort_values(by='casual',ascending=False).head(21)
```

```
colors = ['#8B0A50', '#009ACD']
df_grouped_end_station.plot(kind='barh', stacked=True, color=colors)
plt.title('Top Casual Users End Stations')
plt.xlabel('Number of trips')
plt.ylabel('End station name')
plt.legend('User Type')
plt.show()
```


Top Casual Users End Stations
LaSalle St & Illinois St
Michigan Ave & Washington St
Wells St & Elm St
Sheffield Ave & Waveland Ave
Wabash Ave & Grand Ave
Broadway & Barry Ave
Clark St & Newport St
Clark St & Elm St
Montrose Harbor
Indiana Ave & Roosevelt Rd
Dusable Harbor
Clark St & Lincoln Ave
Clark St & Armitage Ave
Wells St & Concord Ln
Shedd Aquarium
Theater on the Lake
DuSable Lake Shore Dr & North Blvd
Michigan Ave & Oak St
Millennium Park
DuSable Lake Shore Dr & Monroe St
Streeter Dr & Grand Ave
End station name
0
10000
20000
30000
40000
50000
60000
70000
Number of trips
U
s

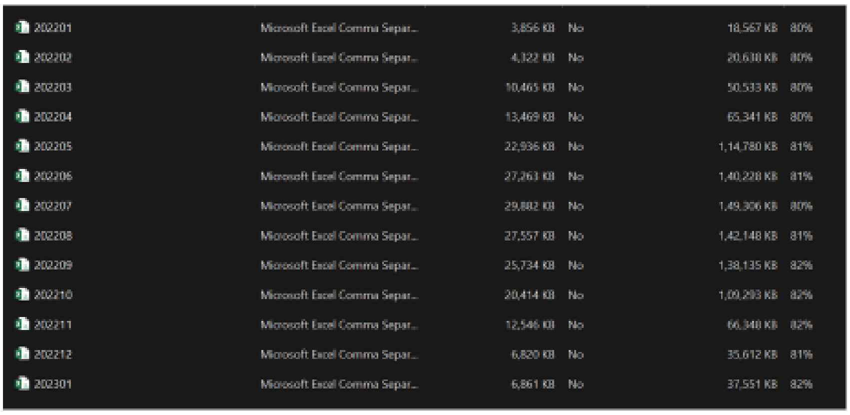
Phase 5 - Share

For the presentation, let's insert the charts into PowerPoint and point out the most important information and analysis.



Phase 2 - Prepare

Using Cyclistic's historical trip data to analyze and identify trends:



202201	Microsoft Excel Comma Separ...	3,856 KB	No	18,567 KB	80%
202202	Microsoft Excel Comma Separ...	4,322 KB	No	20,638 KB	80%
202203	Microsoft Excel Comma Separ...	10,465 KB	No	50,533 KB	80%
202204	Microsoft Excel Comma Separ...	13,469 KB	No	65,341 KB	80%
202205	Microsoft Excel Comma Separ...	22,936 KB	No	1,14,780 KB	81%
202206	Microsoft Excel Comma Separ...	27,263 KB	No	1,40,228 KB	81%
202207	Microsoft Excel Comma Separ...	29,882 KB	No	1,49,306 KB	80%
202208	Microsoft Excel Comma Separ...	27,557 KB	No	1,42,148 KB	81%
202209	Microsoft Excel Comma Separ...	25,734 KB	No	1,38,135 KB	82%
202210	Microsoft Excel Comma Separ...	20,414 KB	No	1,09,293 KB	82%
202211	Microsoft Excel Comma Separ...	12,546 KB	No	66,348 KB	82%
202212	Microsoft Excel Comma Separ...	6,820 KB	No	35,612 KB	81%
202301	Microsoft Excel Comma Separ...	6,861 KB	No	37,551 KB	82%

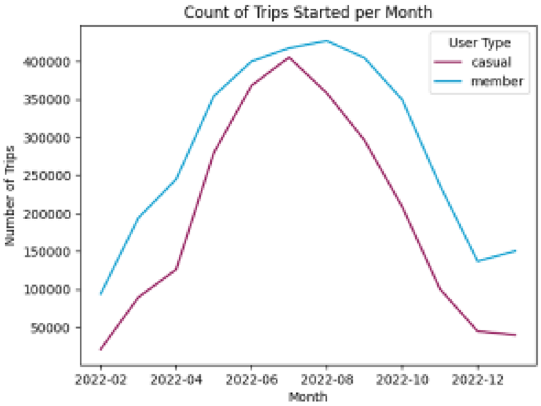
Phase 3 - Process

- Checked the data for errors using Python.

- Transformed the data into a unique Data Frame to work with it effectively.

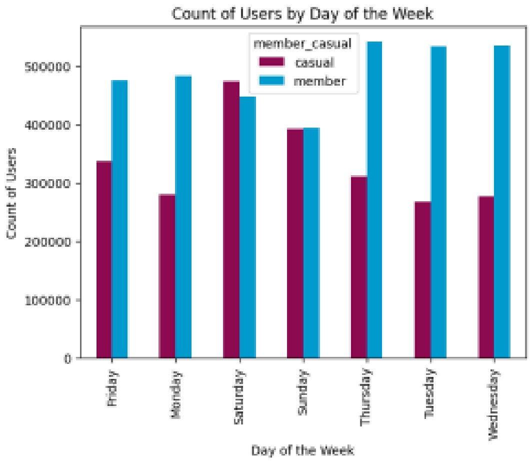
- Documented the cleaning process.

Phase 5 - Share



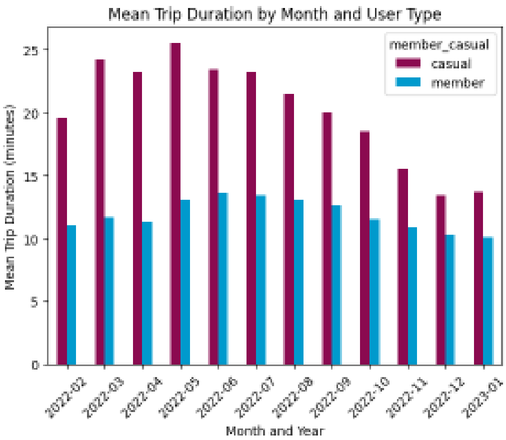
A marketing action to convert casual users into members should be more effective in the summer, especially in June and July.

Phase 5 - Share



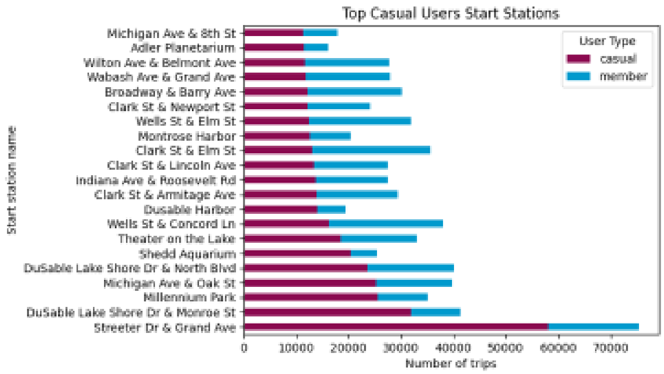
Weekends seem like the best days to reach casual users.

Phase 5 - Share



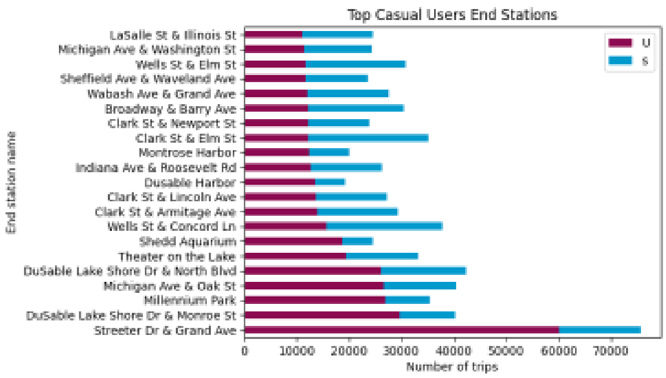
Casual users clearly tend to ride longer. Members uses the bikes for shorter periods.

Phase 5 - Share



In addition to the quantity, it is interesting to note the proportions between user types.

Phase 5 - Share



Stations with a big quantity and proportion of casual users may be the best ones to focus the marketing strategies.

Phase 6 - Act

- 40,7% are casual users.
- Usage increases a lot in the summer.
- Weekends seem like the best days to reach casual users.
- Casual users clearly tend to ride longer.
- Marketing strategies should focus in the top casual user stations.

Phase 6 - Act

This study helped us better understand "How do annual members and casual riders use Cyclistic bikes differently".

Now we know when and where to focus marketing actions.

It is suggested to carry out a survey at the main stations to understand the casual user's profile and the reasons for not becoming a member.

Google

Data Analytics Professional Certificate Capstone

Thank You

Phase 6 – Act

- 40,7% are casual users.
- Usage increases a lot in the summer.
- Weekends seem like the best days to reach casual users.
- Casual users tend to ride longer.
- Marketing strategies should focus on the top casual user stations.

This study helped us better understand “How do annual members and casual riders use Cyclistic bikes differently”.

Now we know when and where to focus marketing actions.

Actions should prioritize the summer period, especially on weekends, in stations with a higher concentration of casual users.

It is suggested to survey the main stations to understand the casual user's profile and the reasons for not becoming a member.

Google

Data Analytics Capstone – Ask, Prepare, Process, Analyze, Share, and Act