How Does a Bike-Share Navigate Speedy Success?

Prepared by

Burak Mustafa Karakaya

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

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?

Prepare
I will use Cyclistic's historical trip data to analyze and identify trends from January 2023 to March 2022(First Quarter) which can be downloaded from divvy_tripdata. The data has been made available by Motivate International Inc. under this license.

This is public data that can be used to explore how different customer types are using Cyclistic bikes. But note that data-privacy issues prohibit from using riders' personally identifiable information. This means that we won't be able to connect pass purchases to credit card numbers to determine if casual riders live in the Cyclistic service area or if they have purchased multiple single passes.

Three files follow the naming convention YYYYMM-divvy-tripdata, each containing data for one month. The information in these files includes details such as ride ID, bike type, start time, end time, start station, end station, start location, end location, and membership status. The corresponding column names are ride_id, rideable_type, started_at, ended_at, start_station_name, start_station_id, end_station_name, end_station_id, start_lat, start_lng, end_lat, end_lng, and member_casual.

Upon examination, all three files exhibit consistency in terms of the number and names of columns. The data types are uniform across all files. The source of the data is Cyclist's first-party data, ensuring a low likelihood of bias. The credibility of the data is exceptionally high since it originates from the company itself.

The data also adheres to the ROCCC criteria: it is Reliable, Original, Comprehensive, Current, and Cited. This further reinforces the trustworthiness of the dataset.

Process

I will use Python to merge, clean, and analyze three different datasets. Python, with its various libraries (e.g., pandas, numpy, datetime, statistics), makes working with data more convenient.

First, we download the necessary libraries.

```
# Libraries
import pandas as pd
import numpy as np
from statistics import mode
from datetime import datetime
```

Then, we read the downloaded CSV files.

```
# Q1 csv files (2023-01, 2023-02, 2023-03)
df1 = pd.read_csv("C:/Users/burak/OneDrive/Masaüstü/Google Data Analytics/csv.data/202301-divvy-tripdata.csv")
df2 = pd.read_csv("C:/Users/burak/OneDrive/Masaüstü/Google Data Analytics/csv.data/202302-divvy-tripdata.csv")
df3 = pd.read_csv("C:/Users/burak/OneDrive/Masaüstü/Google Data Analytics/csv.data/202303-divvy-tripdata.csv")
```

By writing a function, we merge these three different datasets into a single dataset named df0.

```
# Merge function
def merge_df(*dfs):
    merged_df = pd.concat(dfs, ignore_index = True)
    return merged_df

df0 = merge_df(df1, df2, df3)
```

When examining the first 5 rows of our dataset, we can see the columns and the data within them. However, this is not sufficient for understanding our dataset.

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	start_lat	start_lng	end_lat	end_lng	member_casual
0	F96D5A74A3E41399	electric_bike	2023-01-21 20:05:42	2023-01-21 20:16:33	Lincoln Ave & Fullerton Ave	TA1309000058	Hampden Ct & Diversey Ave	202480.0	41.924074	-87.646278	41.930000	-87.640000	member
1	13CB7EB698CEDB88	classic_bike	2023-01-10 15:37:36	2023-01-10 15:46:05	Kimbark Ave & 53rd St	TA1309000037	Greenwood Ave & 47th St	TA1308000002	41.799568	-87.594747	41.809835	-87.599383	member
2	BD88A2E670661CE5	electric_bike	2023-01-02 07:51:57	2023-01-02 08:05:11	Western Ave & Lunt Ave	RP-005	Valli Produce - Evanston Plaza	599	42.008571	-87.690483	42.039742	-87.699413	casual
3	C90792D034FED968	classic_bike	2023-01-22 10:52:58	2023-01-22 11:01:44	Kimbark Ave & 53rd St	TA1309000037	Greenwood Ave & 47th St	TA1308000002	41.799568	-87.594747	41.809835	-87.599383	member
4	3397017529188E8A	classic_bike	2023-01-12 13:58:01	2023-01-12 14:13:20	Kimbark Ave & 53rd St	TA1309000037	Greenwood Ave & 47th St	TA1308000002	41.799568	-87.594747	41.809835	-87.599383	member

```
1  df0.describe(include = ["object"])
```

The describe() function provides total counts, unique value counts, and most frequently occurring values for all features. Let's conduct a brief review of the 'rideable_type' column.

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	member_casual
count	639424	639424	639424	639424	551320	551188	546408	546267	639424
unique	639424	3	594512	595322	1102	1066	1120	1082	2
top	F96D5A74A3E41399	electric_bike	2023-03-21 17:27:42	2023-03-22 17:30:49	University Ave & 57th St	KA1503000071	University Ave & 57th St	KA1503000071	member
freq	1	345206	5	11	5908	5908	5908	5908	494199

We can observe that there are 3 different rideable types.

Electric bikes being the most preferred type at 345,206 rides out of a total of 639,424 rides.

In the following code block, we see the total number of missing values in our columns. It is evident that there are many missing values in the start_station_name, start_station_id, end_station_name, and end_station_id columns.

```
df0.isnull().sum()
```

```
ride_id                   0
rideable_type             0
started_at                0
ended_at                  0
start_station_name    88104
start_station_id      88236
end_station_name      93016
end_station_id        93157
start_lat                 0
start_lng                 0
end_lat                 426
end_lng                 426
member_casual             0
dtype: int64
```

```
df0.dtypes
```

```
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
```

After learning about our data types, we proceed to acquaint ourselves with our dataset. We do not want the started_at and ended_at columns to be objects because they represent date values.

Now, we can clean and organize our data. First, we convert the started_at and ended_at columns from object to datetime.

```
# Converting dtypes
df0["started_at"] = pd.to_datetime(df0["started_at"], errors="coerce")
df0["ended_at"] = pd.to_datetime(df0["ended_at"], errors="coerce")
```

```
df0.dtypes
```

✓ 0.0s

```
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
```

We add a column called ride_length, which calculates the difference between the end and start dates. This allows us to see the duration of rides.

```
df0["ride_length"] = df0["ended_at"] - df0["started_at"]

df0["ride_length"]
```

```
0         0 days 00:10:51
1         0 days 00:08:29
2         0 days 00:13:14
3         0 days 00:08:46
4         0 days 00:15:19
                ...
639419    0 days 00:06:23
639420    0 days 00:26:03
639421    0 days 00:02:35
639422    0 days 00:07:06
639423    0 days 00:05:44
Name: ride_length, Length: 639424, dtype: timedelta64[ns]
```

By adding the day_of_week column, we create a new column for the days of the week in the starting dates. The +1 in the apply() function ensures that the days are from 1 to 7 instead of 0 to 6.

```python
# sunday = 1, saturday = 7
df0["day_of_week"] = df0["started_at"].apply(lambda x: (x.dayofweek + 1) % 7 + 1)
```

When looking at the statistical values of the newly added columns, I see that the average value of the ride_length column is 13 minutes. However, I notice anomalies, such as a ride duration of 23 days and 8 hours, which is not realistic. There is also an anomaly in the minimum value, as the started_at date cannot be after the ended_at date. Finally, I observe that the 3rd day of the week has the highest number of rides.

```python
print(df0["ride_length"].mean())
print("-"*40)
print(df0["ride_length"].max()) # anomaly detected
print("-"*40)
print(df0["day_of_week"].mode())
print("-"*40)
print(df0["ride_length"].min()) # anomaly detected
```

```
0 days 00:13:11.478791850
----------------------------------------
23 days 08:03:44
----------------------------------------
0    3
Name: day_of_week, dtype: int64
----------------------------------------
-1 days +23:56:44
```

To verify these anomalies, I first retrieve the minimum and maximum ride_length values. It is evident that there are incorrect date entries. Unfortunately, these anomalies will have a negative impact on our analysis. Therefore, we need to find all row values with this anomaly.

```python
ridelength_min = df0[df0["ride_length"] == df0["ride_length"].min()]

print(ridelength_min)
```

```
                ride_id  rideable_type          started_at  \
379648  4EFC95304E050AA1  electric_bike 2023-02-04 13:08:08

                  ended_at start_station_name start_station_id  \
379648 2023-02-04 13:04:52                NaN              NaN

            end_station_name end_station_id  start_lat  start_lng  \
379648  Dearborn St & Monroe St   TA1305000006      41.88     -87.63

         end_lat    end_lng member_casual          ride_length day_of_week
379648  41.88132 -87.629521        member -1 days +23:56:44           7
```

```python
ridelength_max = df0[df0["ride_length"] == df0["ride_length"].max()]

print(ridelength_max)
```

```
              ride_id rideable_type          started_at            ended_at  \
72088  307CA01BAE3CC7E3   docked_bike 2023-01-08 11:08:52 2023-01-31 19:12:36

      start_station_name start_station_id end_station_name end_station_id  \
72088  Michigan Ave & 8th St              623              NaN            NaN

       start_lat  start_lng  end_lat  end_lng member_casual       ride_length \
72088  41.872773 -87.623981      NaN      NaN        casual 23 days 08:03:44

       day_of_week
72088            1
```

I want to know the count of other rows with anomaly_period_negative and anomaly_period_positive columns.

```python
df0["anomaly_period_negative"] = df0["ended_at"] < df0["started_at"]
df0["anomaly_period_positive"] = df0["ride_length"] > pd.Timedelta("1 days")
```

While there is a negative 1 value, there are 387 rides lasting more than 1 day.

```python
df0[df0["anomaly_period_negative"]].count()
```

```
ride_id                    1
rideable_type              1
started_at                 1
ended_at                   1
start_station_name         0
start_station_id           0
end_station_name           1
end_station_id             1
start_lat                  1
start_lng                  1
end_lat                    1
end_lng                    1
member_casual              1
ride_length                1
day_of_week                1
anomaly_period_negative    1
anomaly_period_positive    1
dtype: int64
```

```python
df0[df0["anomaly_period_positive"]].count()
```

```
ride_id                    387
rideable_type              387
started_at                 387
ended_at                   387
start_station_name         387
start_station_id           387
end_station_name            12
end_station_id              12
start_lat                  387
start_lng                  387
end_lat                     17
end_lng                     17
member_casual              387
ride_length                387
day_of_week                387
anomaly_period_negative    387
anomaly_period_positive    387
dtype: int64
```

Since removing these anomaly values will result in a more meaningful analysis, I delete them from the dataset.

```python
df0.drop(df0[df0["anomaly_period_negative"] | df0["anomaly_period_positive"]].index, inplace=True)
```

I also decide to check if there are rides with 0 duration for control purposes and find 35 rides with 0 duration. I remove them from our dataset as well.

```python
zero_length_rides = df0[df0["ride_length"] == pd.Timedelta(0)]
print(zero_length_rides.count())
```

```
ride_id                    35
rideable_type              35
started_at                 35
ended_at                   35
start_station_name         35
start_station_id           35
end_station_name           26
end_station_id             26
start_lat                  35
start_lng                  35
end_lat                    35
end_lng                    35
member_casual              35
ride_length                35
day_of_week                35
anomaly_period_negative    35
anomaly_period_positive    35
dtype: int64
```

```python
df0["anomaly_period_zero"] = df0["ride_length"] == pd.Timedelta("0 seconds")
```

```python
df0 = df0[df0["ride_length"] != pd.Timedelta("0 seconds")]
```

Now, our dataset is cleaned, and a total of 423 entries have been removed. When checking the data types, I see that all of them are correct. Additionally, there are now unnecessary columns that we will remove later; but before that, I want the ride_length value to be an integer instead of timedelta64.

```
df0.count()
```

```
ride_id                    639001
rideable_type              639001
started_at                 639001
ended_at                   639001
start_station_name         550898
start_station_id           550766
end_station_name           546369
end_station_id             546228
start_lat                  639001
start_lng                  639001
end_lat                    638945
end_lng                    638945
member_casual              639001
ride_length                639001
day_of_week                639001
anomaly_period_negative    639001
anomaly_period_positive    639001
anomaly_period_zero        639001
dtype: int64
```

```
df0.dtypes
```

```
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]
day_of_week                           int64
anomaly_period_negative                bool
anomaly_period_positive                bool
anomaly_period_zero                    bool
dtype: object
```

First, I convert from timedelta to str, then from str to int, and during this process, I assign it to a new column named ride_duration_int.

```
# Converting ride_length timedelta64 to integer
df0['ride_duration_str'] = df0['ride_length'].astype(str).str.extract(r'(\d+:\d+:\d+|\d+:\d+)')[0]
df0['ride_duration_str'] = df0['ride_duration_str'].str.replace(":", ".")
df0["ride_duration_str"] = df0["ride_duration_str"].str[-8:]
```

```
df0['hour'] = df0['ride_duration_str'].str.split('.').str[0].astype(float)
df0['minute'] = df0['ride_duration_str'].str.split('.').str[1].astype(float)
df0['second'] = df0['ride_duration_str'].str.split('.').str[2].astype(float)

df0['ride_duration_int'] = df0['hour'] * 60 + df0['minute'] + df0['second'] / 60

df0['ride_duration_int'] = df0['ride_duration_int'].round(2)
```

	ride_duration_int
0	10.85
1	8.48
2	13.23
3	8.77
4	15.32
...	...
639419	6.38
639420	26.05
639421	2.58
639422	7.10
639423	5.73

639001 rows × 1 columns

Next, it's time to remove the unnecessary columns from our dataset. I have removed all unnecessary columns.

```
df0 = df0.drop(["second", "minute", "hour", "ride_duration_str", "anomaly_period_zero", "anomaly_period_negative", "anomaly_period_positive"], axis = 1)
```

```
df0.columns
```

```
Index(['ride_id', 'rideable_type', 'started_at', 'ended_at',
       'start_station_name', 'start_station_id', 'end_station_name',
       'end_station_id', 'start_lat', 'start_lng', 'end_lat', 'end_lng',
       'member_casual', 'ride_length', 'day_of_week', 'ride_duration_int'],
      dtype='object')
```

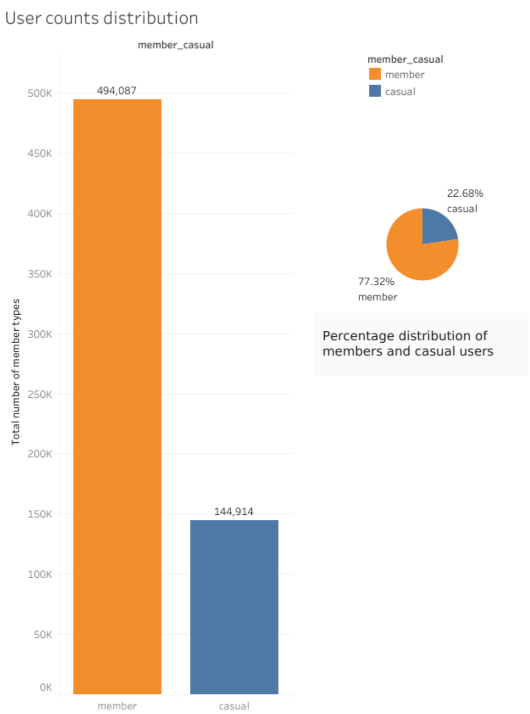
```
df0.to_csv(r"C:/Users/burak/OneDrive/Masaüstü/Google Data Analytics/updated.csv", index = False)
```

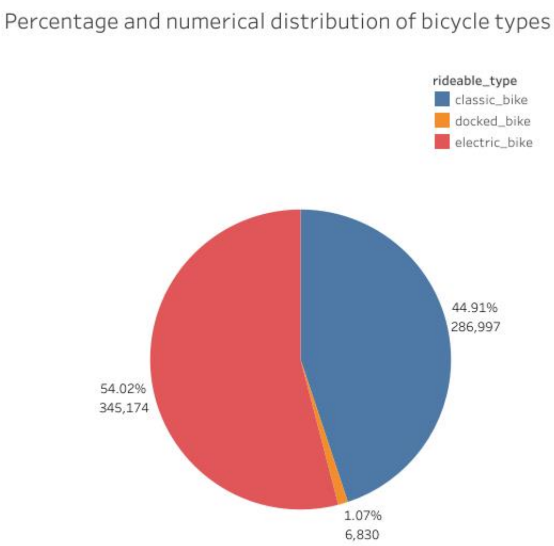
Data Cleaning:

- Data types have been adjusted ('started_at', 'ended_at').
- Necessary columns have been added ('ride_length', 'day_of_week', 'ride_duration_int').
- Data anomalies have been detected and removed. A total of 423 entries were deleted.

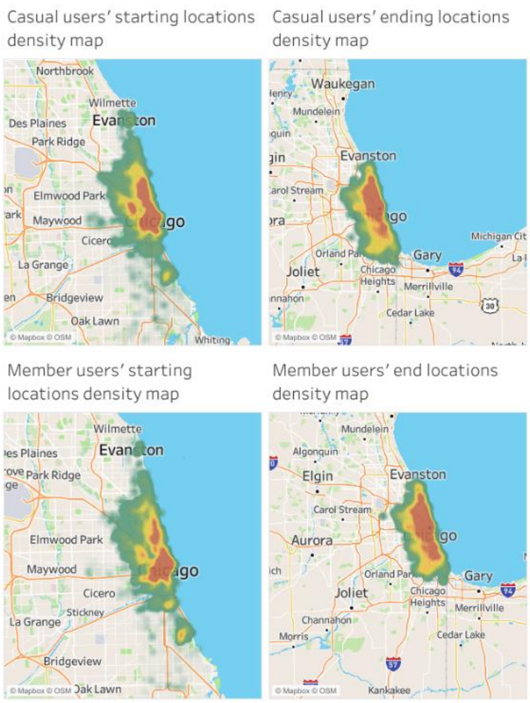
Analyze and Sharing:

When we look at the total number of user types, we have 494,087 members and 144,914 casual users. In terms of proportional distribution, 77.32% of users are members, while 22.68% are in the casual status.



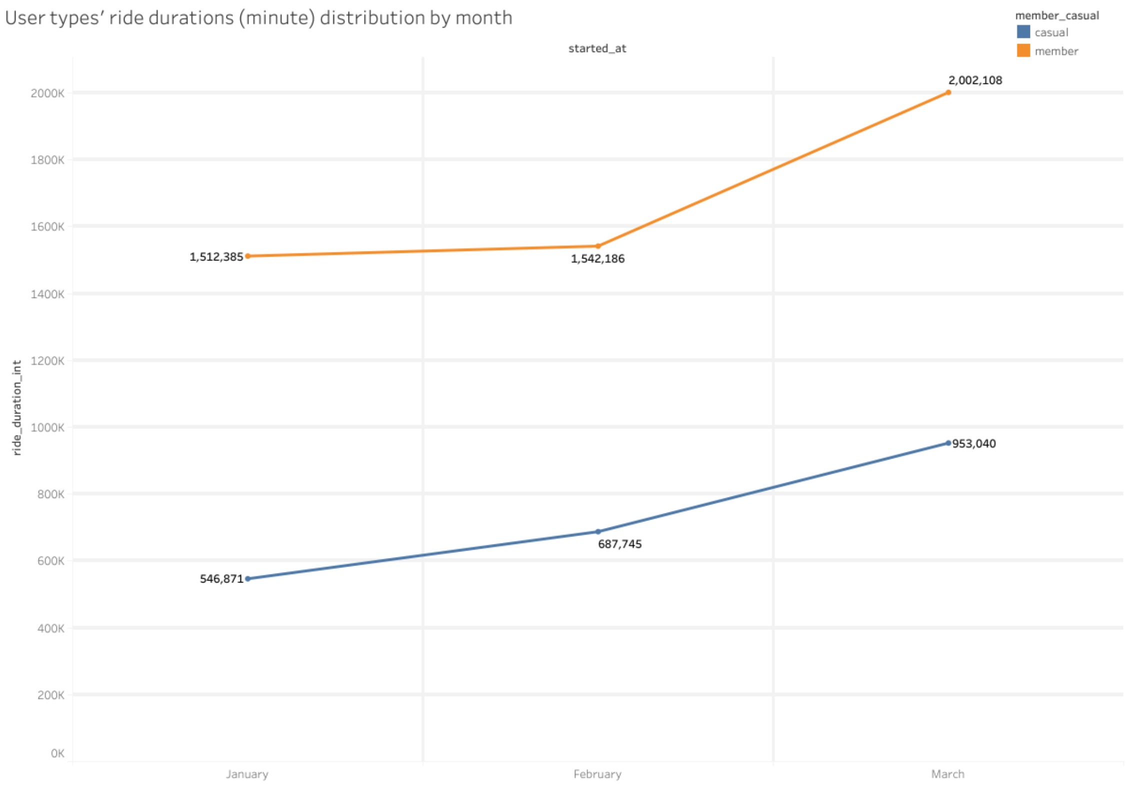


In terms of rideable bike types, our users prefer electric bikes the most (54.02%), while docked bikes are the least preferred (1.07%).



We observe the density maps of starting and ending locations for member and casual user rides. While starting points are close for both user types, when looking at the ending points, we see that member users tend to finish their rides more towards the east compared to casual users.

When looking at the total ride duration by month, we see that March is the month with the most ride duration. The main reason for this is that March has milder weather conditions compared to January and February.



When we look at the most preferred starting and ending stations for our member users, we see locations such as universities, workplaces, and residential areas. It can be roughly stated that member users aim for commuting to and from work or school.

```
# The most preferred starting point for member users
df0[df0["member_casual"] == "member"].groupby("start_station_name")["start_station_name"].count().sort_values(ascending=False).head(10)
```

✓ 0.2s

```
start_station_name
University Ave & 57th St        4976
Ellis Ave & 60th St             4836
Clinton St & Washington Blvd    4499
Kingsbury St & Kinzie St        4095
Clark St & Elm St               3674
Canal St & Adams St             3487
Clinton St & Madison St         3386
State St & Chicago Ave          3319
Ellis Ave & 55th St             3306
Loomis St & Lexington St        3157
Name: start_station_name, dtype: int64
```

```
# The most preferred ending point for member users
df0[df0["member_casual"] == "member"].groupby("end_station_name")["end_station_name"].count().sort_values(ascending=False).head(10)
```

✓ 0.2s

```
end_station_name
University Ave & 57th St        4945
Clinton St & Washington Blvd    4921
Ellis Ave & 60th St             4758
Kingsbury St & Kinzie St        4189
Clinton St & Madison St         3662
Clark St & Elm St               3578
Canal St & Adams St             3552
Ellis Ave & 55th St             3384
State St & Chicago Ave          3353
Loomis St & Lexington St        3293
Name: end_station_name, dtype: int64
```

When we examine the most preferred starting and ending stations for our casual users, we observe parks, aquariums, and tourist attractions. It can be roughly said that casual users engage in cycling more for leisure purposes.

```
# The most preferred starting point for casual users
df0[df0["member_casual"] == "casual"].groupby("start_station_name")["start_station_name"].count().sort_values(ascending=False).head(10)
```

✓ 0.1s

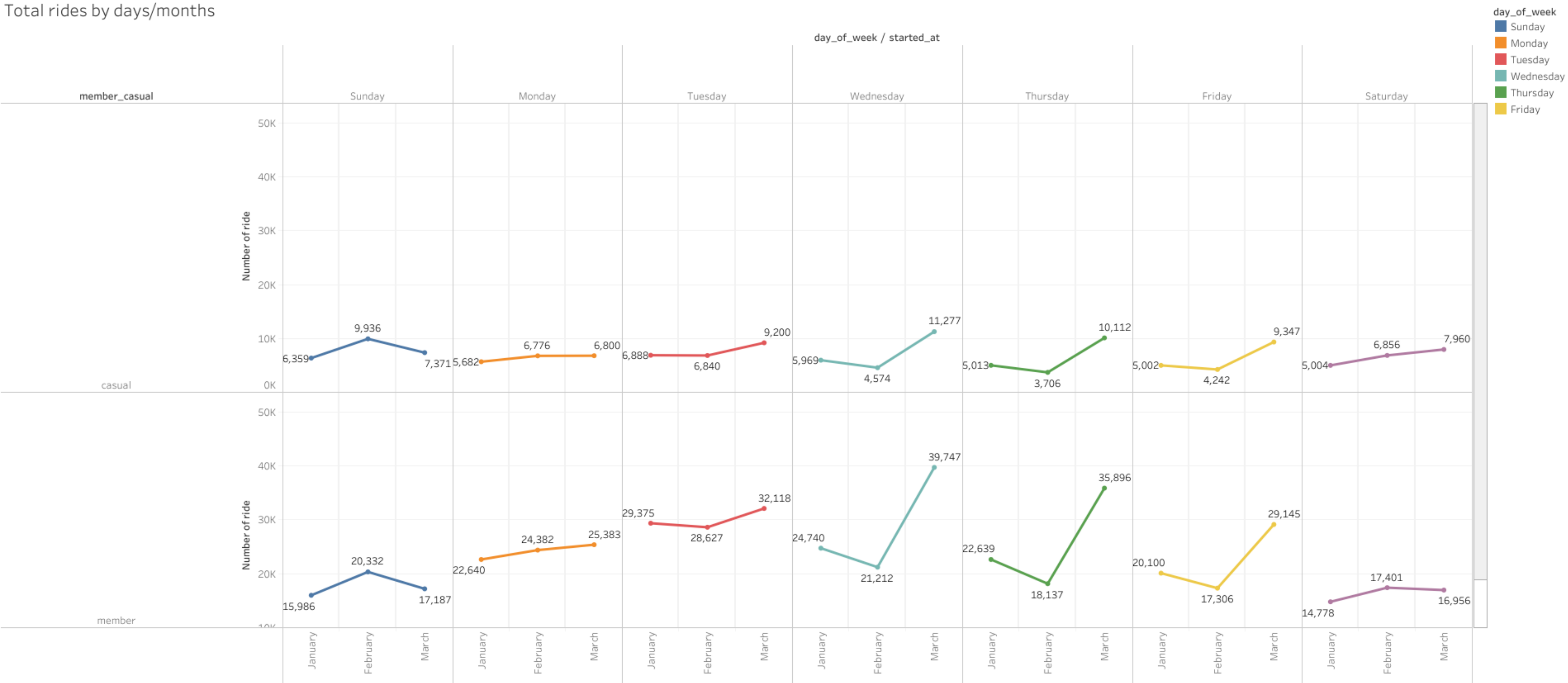
```
start_station_name
Streeter Dr & Grand Ave            1626
Shedd Aquarium                     1295
DuSable Lake Shore Dr & Monroe St  1281
Millennium Park                    1067
University Ave & 57th St            929
Ellis Ave & 60th St                 906
Wells St & Concord Ln               887
Kingsbury St & Kinzie St            836
Sheffield Ave & Fullerton Ave       826
LaSalle St & Illinois St            773
Name: start_station_name, dtype: int64
```

```
# The most preferred ending point for casual users
df0[df0["member_casual"] == "casual"].groupby("end_station_name")["end_station_name"].count().sort_values(ascending=False).head(10)
```

✓ 0.1s

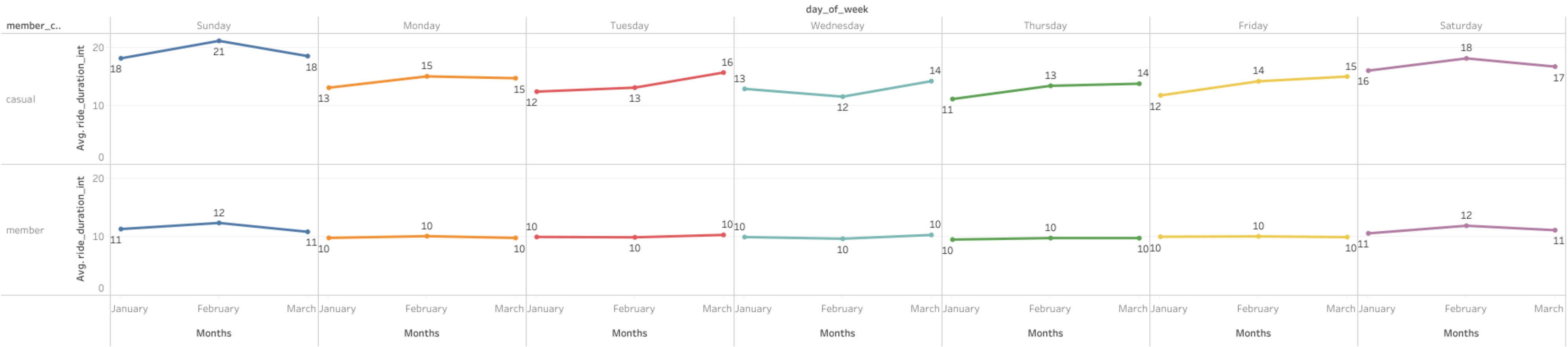
```
end_station_name
Streeter Dr & Grand Ave              1872
Millennium Park                      1230
Shedd Aquarium                       1021
University Ave & 57th St              962
DuSable Lake Shore Dr & Monroe St     924
Wells St & Concord Ln                 894
Ellis Ave & 60th St                   870
Michigan Ave & Washington St          868
LaSalle St & Illinois St              824
Broadway & Barry Ave                  809
Name: end_station_name, dtype: int64
```

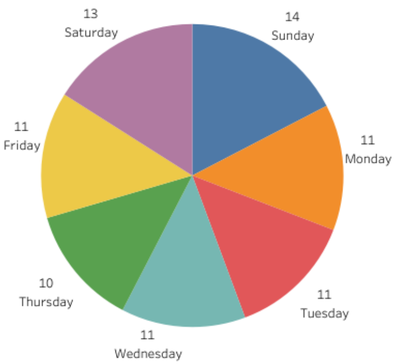
On a daily basis, we see that on Sundays in February, both types of members ride more compared to other months. Wednesdays and Thursdays in March also show an almost 2 to 3 times increase in rides for both user types compared to other months. Additionally, the better weather conditions of March contribute positively to ride numbers on the remaining days.



When it comes to the distribution of average ride durations by day and month, we see that Sunday is the day with the longest ride duration. The days with the most rides, Wednesday, and Thursday, have average durations shorter than weekends. Especially for casual users, this difference is more pronounced. This suggests that casual users enjoy longer bike rides on weekends, possibly for recreational purposes such as city tours or sightseeing. On the other hand, member users show a more consistent distribution, indicating they use bikes for commuting purposes, such as going to work or school, regardless of the day of the week.

Average ride duration(minute) by day/month distribution

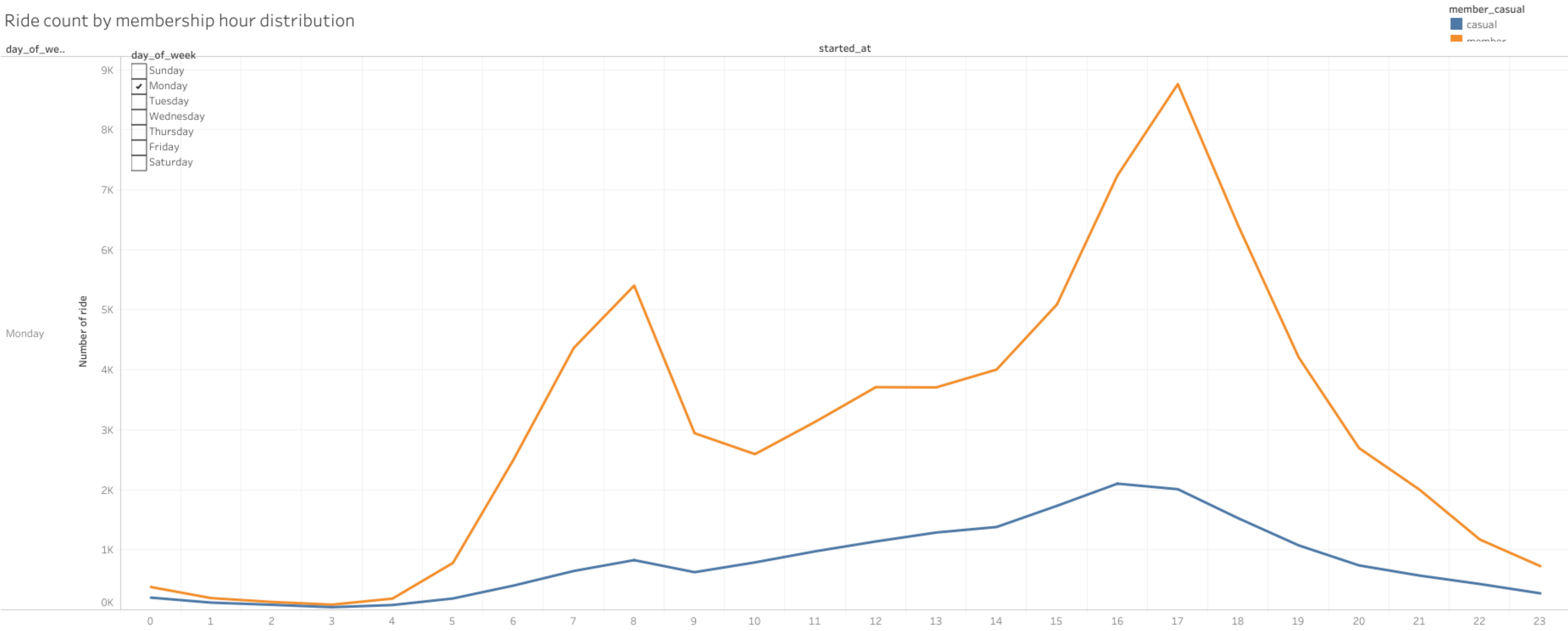




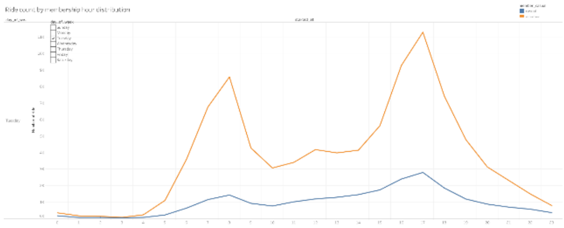
Average ride duration (minute)

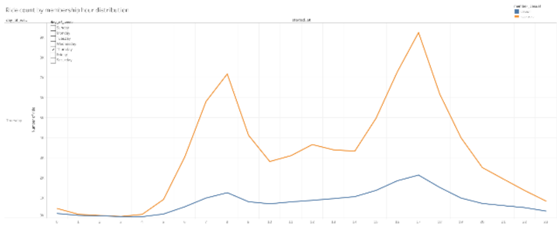
Longest day of cycling: ***Sunday***
Shortest day of cycling: ***Thursday***

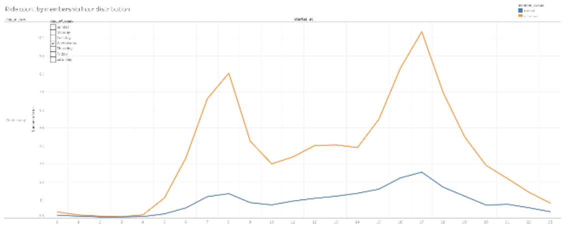
To better understand, we examine the ride counts by hour/day. On Monday, member users prefer the hours between 8-17. This suggests the usage for school and work purposes. It could also be used for lunch breaks around 12:00, as evidenced by the number of rides, where users ride their bikes to restaurants or cafes during those hours. Casual users, on the other hand, prefer the hours around the end of school or work during the day. However, they don't seem to prefer riding bikes in the morning when going to school or work.

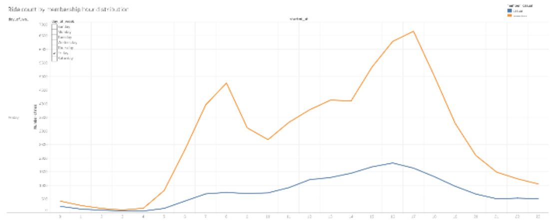


This pattern is valid for Tuesday, Wednesday, Thursday, and Friday as well.

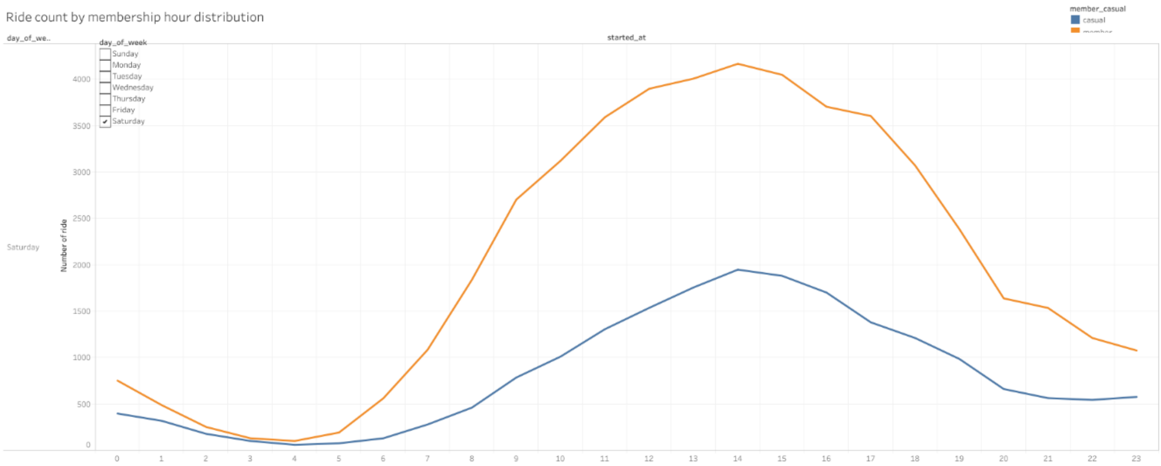


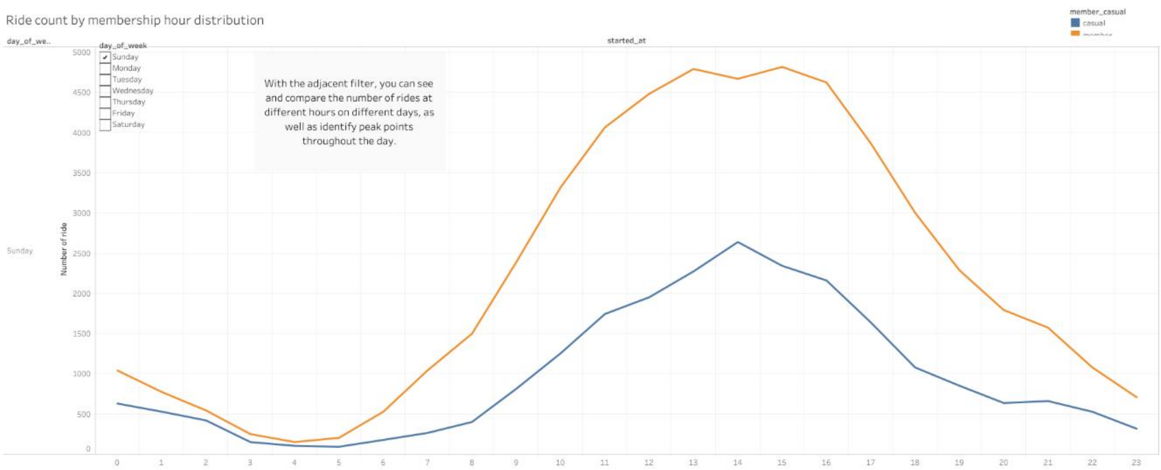




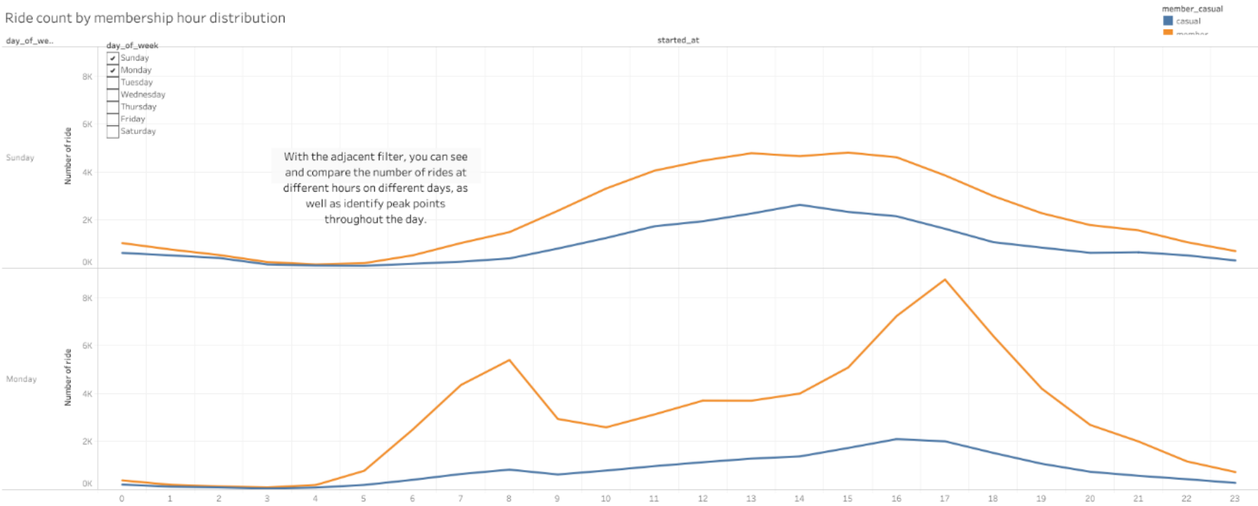


On Saturday, both casual and member users have peak hours at 14:00. This is because Saturday rides are likely for leisure purposes, such as sports, activities with friends or family, city tours, and tourist trips. The same applies to Sunday.





When comparing Sunday and Monday, we can understand the difference in the purpose of rides. Looking at the hours, during the weekdays, the rides are for work or school purposes, while during the weekends, they are for leisure. Although people use it more during the weekdays, the ride durations show that rides on weekends are longer.



How do annual members and casual riders use Cyclistic bikes differently?

Member	**Casual**
Members typically ride bicycles during weekdays, including the start and end times of work or school, as well as during lunch breaks.	Casual users enjoy cycling after work or school on weekdays.
They generally maintain a consistent duration of cycling throughout the weekdays.	On weekends, their riding durations are significantly longer compared to weekdays.
The popular starting and ending points are located near schools, workplaces, and commercial buildings.	The popular starting and ending points are close to parks, museums, aquarium and tourist attractions.

Action Plan:

- Evaluate the winter compatibility of our bikes in challenging conditions, such as January and February. Assessing features like wheels and the power of electric bikes, we can encourage the use of winter-specific bikes during the winter months, potentially increasing our user base.

- Increase annual membership by offering special discounts and introducing winter-specific bikes through marketing campaigns during the winter months.

- Since casual users prefer riding on weekends, create special campaigns and advantageous memberships for them during weekends.

- Highlight stations in tourist locations (museums, historical sites, natural attractions, cinemas, popular places) in social media advertisements. People are more likely to choose us when they know we have stations in such locations.

- Review membership and single-use pricing. Conduct A/B testing with Decoy Effect research and revise pricing strategies.