Report

April 13, 2023

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

1.1 Introduction

This is a capstone project for the Google Data Analytics Professional Certificate. This program, offered by Coursera, serves as preparation for aspiring data analysts to start their career by learning key data analysis skills such as preparing, cleaning, and analyzing data; creating visualizations; and using tools such as SQL, Excel, Tableau, and R.

Suppose we are a junior data analyst working for Cyclistic, a bike-share company. We are assigned to work with the Cyclistic market analyst team. Our manager believes that to grow the company, we should maximize the number of annual members by converting casual riders into members. The task is to understand how casual riders and annual members use Cyclistic bikes differently. We want to develop a marketing strategy to convert casual riders into annual members. In order to do this, we need to analyze the Cyclistic historical bike trip data to identify trends. This dataset is offered by the Coursera course.

To answer the business questions, we will follow the steps of the data analysis process: ask, prepare, process, analyze, share, and act.

1.2 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.

1.3 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?

The problem we are trying to solve is trying to maximize the number of annual members. We will be analyzing trends to learn about how casual riders and annual members use Cyclistic bikes differently. Our analysis will be shared with our manager and Cyclistic marketing analytics team, which will be passed on to the Cyclistic executive team to decide whether to approve the recommended marketing program.

1.4 Prepare

We will use Cyclistic's historical trip data to analyze and identify trends. This dataset has been made available by Motivate International Inc. under this license https://ride.divvybikes.com/data-license-agreement.

[5]:
```
import pandas as pd
import datetime
```

We'll first read the data we obtained from the website.

[6]:
```
december21 = pd.read_csv("202112-divvy-tripdata.csv")
```

[7]:
```
january22 = pd.read_csv("202201-divvy-tripdata.csv")
```

[8]:
```
february22 = pd.read_csv("202202-divvy-tripdata.csv")
```

[9]:
```
march22 = pd.read_csv("202203-divvy-tripdata.csv")
```

[10]:
```
april22 = pd.read_csv("202204-divvy-tripdata.csv")
```

[11]:
```
may22 = pd.read_csv("202205-divvy-tripdata.csv")
```

[12]:
```
june22 = pd.read_csv("202206-divvy-tripdata.csv")
```

[13]:
```
july22 = pd.read_csv("202207-divvy-tripdata.csv")
```

[14]:
```
august22 = pd.read_csv("202208-divvy-tripdata.csv")
```

[15]:
```
september22 = pd.read_csv("202209-divvy-publictripdata.csv")
```

[16]:
```
october22 = pd.read_csv("202210-divvy-tripdata.csv")
```

[17]:
```
november22 = pd.read_csv("202211-divvy-tripdata.csv")
```

[18]:
```
december21.head()
```

[18]:

	ride_id	rideable_type	started_at	ended_at	\
0	46F8167220E4431F	electric_bike	2021-12-07 15:06:07	2021-12-07 15:13:42	
1	73A77762838B32FD	electric_bike	2021-12-11 03:43:29	2021-12-11 04:10:23	
2	4CF42452054F59C5	electric_bike	2021-12-15 23:10:28	2021-12-15 23:23:14	
3	3278BA87BF698339	classic_bike	2021-12-26 16:16:10	2021-12-26 16:30:53	
4	6FF54232576A3B73	electric_bike	2021-12-30 11:31:05	2021-12-30 11:51:21	

```
              start_station_name start_station_id            end_station_name  \
0      Laflin St & Cullerton St            13307         Morgan St & Polk St
1         LaSalle Dr & Huron St     KP1705001026  Clarendon Ave & Leland Ave
2  Halsted St & North Branch St     KA1504000117        Broadway & Barry Ave
3  Halsted St & North Branch St     KA1504000117       LaSalle Dr & Huron St
4      Leavitt St & Chicago Ave            18058      Clark St & Drummond Pl

  end_station_id  start_lat  start_lng    end_lat    end_lng member_casual
0   TA1307000130  41.854833 -87.663660  41.871969 -87.650965        member
1   TA1307000119  41.894405 -87.632331  41.967968 -87.650001        casual
2          13137  41.899357 -87.648522  41.937582 -87.644098        member
3   KP1705001026  41.899390 -87.648545  41.894877 -87.632326        member
4   TA1307000142  41.895579 -87.682024  41.931248 -87.644336        member
```

We'll start out by combining all the datasets into one big data frame.

[19]:
```
bike_data = pd.concat(objs = [december21, january22, february22, march22,␣
↪april22, may22, june22, july22, august22, september22,
                              october22, november22])
```

[20]:
```
bike_data.head()
```

[20]:
```
             ride_id  rideable_type           started_at             ended_at  \
0  46F8167220E4431F  electric_bike  2021-12-07 15:06:07  2021-12-07 15:13:42
1  73A77762838B32FD  electric_bike  2021-12-11 03:43:29  2021-12-11 04:10:23
2  4CF42452054F59C5  electric_bike  2021-12-15 23:10:28  2021-12-15 23:23:14
3  3278BA87BF698339   classic_bike  2021-12-26 16:16:10  2021-12-26 16:30:53
4  6FF54232576A3B73  electric_bike  2021-12-30 11:31:05  2021-12-30 11:51:21

              start_station_name start_station_id            end_station_name  \
0      Laflin St & Cullerton St            13307         Morgan St & Polk St
1         LaSalle Dr & Huron St     KP1705001026  Clarendon Ave & Leland Ave
2  Halsted St & North Branch St     KA1504000117        Broadway & Barry Ave
3  Halsted St & North Branch St     KA1504000117       LaSalle Dr & Huron St
4      Leavitt St & Chicago Ave            18058      Clark St & Drummond Pl

  end_station_id  start_lat  start_lng    end_lat    end_lng member_casual
0   TA1307000130  41.854833 -87.663660  41.871969 -87.650965        member
1   TA1307000119  41.894405 -87.632331  41.967968 -87.650001        casual
2          13137  41.899357 -87.648522  41.937582 -87.644098        member
3   KP1705001026  41.899390 -87.648545  41.894877 -87.632326        member
4   TA1307000142  41.895579 -87.682024  41.931248 -87.644336        member
```

The data contains the following columns:

[21]:
```
bike_data.columns
```

[21]:
```
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'],
      dtype='object')
```

We want to make sure that we combined the data correctly by checking if the combined data set has the correct number of rows.

[22]:
```
len(december21) + len(january22) + len(february22) + len(march22) +␣
↪len(april22) + len(may22) + len(june22) + len(july22) + len(august22) +␣
↪len(september22) + len(october22) + len(november22)
```

[22]:
```
5733451
```

[23]:
```
len(bike_data)
```

[23]:
```
5733451
```

[24]:
```
rows = len(bike_data)
print(f"Notice that our dataset has {rows} rows.")
```

```
Notice that our dataset has 5733451 rows.
```

The data isn't sorted, so we'll sort by the column "started_at" in ascending order, meaning the time and date in which a trip has started.

[25]:
```
bike_data = bike_data.sort_values(by = "started_at")
```

[26]:
```
bike_data
```

[26]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id
242975	937DC7C2109D635C	electric_bike	2021-12-01 00:00:01	2021-12-01 00:23:16	NaN	NaN
182646	0CD83C3FE35E69A0	classic_bike	2021-12-01 00:00:03	2021-12-01 00:07:34	State St & Kinzie St	13050
116503	FC2D02B730EBC33D	electric_bike	2021-12-01 00:00:15	2021-12-01 00:02:40	Wells St & Huron St	TA1306000012
229332	227558BB46C7DE48	electric_bike	2021-12-01 00:01:00	2021-12-01 00:05:29	NaN	NaN
73536	5CB387082B4310B2	classic_bike	2021-12-01 00:03:44			
…	…	…	…			
180458	CB5ECCA9D1567A45	electric_bike	2022-11-30 23:54:40			
252640	DD58003A0A0C9FE4	electric_bike	2022-11-30 23:54:46			
177330	0A7F3F32FCC736D1	classic_bike	2022-11-30 23:55:02			
16514	C32D3E0585F38636	electric_bike	2022-11-30 23:55:28			
337271	F6BE77C9DC9AB3BC	electric_bike	2022-11-30 23:56:11			

```
73536   2021-12-01 00:07:44      Ellis Ave & 60th St     KA1503000014
…                     …                       …              …
180458  2022-11-30 23:58:59  State St & Van Buren St     TA1305000035
252640  2022-11-30 23:59:23                      NaN              NaN
177330  2022-12-01 00:21:10     McClurg Ct & Ohio St     TA1306000029
16514   2022-11-30 23:59:46  Broadway & Cornelia Ave            13278
337271  2022-12-01 00:00:52  Clark St & Winnemac Ave     TA1309000035

                    end_station_name end_station_id  start_lat  start_lng  \
242975                           NaN            NaN  41.930000 -87.730000
182646         St. Clair St & Erie St          13016  41.889187 -87.627754
116503                           NaN            NaN  41.894763 -87.634595
229332                           NaN            NaN  41.790000 -87.600000
73536        University Ave & 57th St   KA1503000071  41.785097 -87.601073
…                                …              …        …        …
180458         Michigan Ave & 14th St   TA1307000124  41.877238 -87.627823
252640                           NaN            NaN  41.980000 -87.660000
177330  Sheffield Ave & Fullerton Ave   TA1306000016  41.892592 -87.617289
16514      Clarendon Ave & Gordon Ter          13379  41.945466 -87.646343
337271           Broadway & Ridge Ave          15578  41.973391 -87.667762

          end_lat    end_lng member_casual
242975  41.880000 -87.710000        casual
182646  41.894345 -87.622798        casual
116503  41.900000 -87.640000        member
229332  41.800000 -87.600000        member
73536   41.791478 -87.599861        member
…             …         …           …
180458  41.864059 -87.623727        member
252640  41.970000 -87.650000        member
177330  41.925602 -87.653708        member
16514   41.957867 -87.649505        member
337271  41.984045 -87.660274        casual

[5733451 rows x 13 columns]
```

Let's inspect the datatype of each column.

[27]:
```
bike_data.dtypes
```

[27]:
```
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
```

We need to convert the "started_at" and "ended_at" columns into date and time values.

[28]:
```
bike_data[["started_at", "ended_at"]] = bike_data[["started_at", "ended_at"]].
↪astype("datetime64")
```

[29]:
```
bike_data.dtypes
```

[29]:
```
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
```

[30]:
```
bike_data.sort_values(by = "started_at")
```

[30]:
```
                ride_id  rideable_type          started_at  \
242975  937DC7C2109D635C  electric_bike 2021-12-01 00:00:01
182646  0CD83C3FE35E69A0   classic_bike 2021-12-01 00:00:03
116503  FC2D02B730EBC33D  electric_bike 2021-12-01 00:00:15
229332  227558BB46C7DE48  electric_bike 2021-12-01 00:01:00
73536   5CB387082B4310B2   classic_bike 2021-12-01 00:03:44
…                  …            …                   …
180458  CB5ECCA9D1567A45  electric_bike 2022-11-30 23:54:40
252640  DD58003A0A0C9FE4  electric_bike 2022-11-30 23:54:46
177330  0A7F3F32FCC736D1   classic_bike 2022-11-30 23:55:02
16514   C32D3E0585F38636  electric_bike 2022-11-30 23:55:28
337271  F6BE77C9DC9AB3BC  electric_bike 2022-11-30 23:56:11

                   ended_at       start_station_name start_station_id  \
242975 2021-12-01 00:23:16                      NaN              NaN
182646 2021-12-01 00:07:34     State St & Kinzie St            13050
116503 2021-12-01 00:02:40      Wells St & Huron St     TA1306000012
```

```
229332 2021-12-01 00:05:29                      NaN             NaN
73536  2021-12-01 00:07:44      Ellis Ave & 60th St     KA1503000014
…                    …                       …               …
180458 2022-11-30 23:58:59  State St & Van Buren St     TA1305000035
252640 2022-11-30 23:59:23                      NaN             NaN
177330 2022-12-01 00:21:10     McClurg Ct & Ohio St     TA1306000029
16514  2022-11-30 23:59:46  Broadway & Cornelia Ave            13278
337271 2022-12-01 00:00:52  Clark St & Winnemac Ave     TA1309000035

                    end_station_name end_station_id  start_lat  start_lng  \
242975                           NaN            NaN  41.930000 -87.730000
182646        St. Clair St & Erie St          13016  41.889187 -87.627754
116503                           NaN            NaN  41.894763 -87.634595
229332                           NaN            NaN  41.790000 -87.600000
73536       University Ave & 57th St   KA1503000071  41.785097 -87.601073
…                                 …              …         …          …
180458        Michigan Ave & 14th St   TA1307000124  41.877238 -87.627823
252640                           NaN            NaN  41.980000 -87.660000
177330  Sheffield Ave & Fullerton Ave   TA1306000016  41.892592 -87.617289
16514     Clarendon Ave & Gordon Ter          13379  41.945466 -87.646343
337271          Broadway & Ridge Ave          15578  41.973391 -87.667762

          end_lat    end_lng member_casual
242975  41.880000 -87.710000        casual
182646  41.894345 -87.622798        casual
116503  41.900000 -87.640000        member
229332  41.800000 -87.600000        member
73536   41.791478 -87.599861        member
…             …         …           …
180458  41.864059 -87.623727        member
252640  41.970000 -87.650000        member
177330  41.925602 -87.653708        member
16514   41.957867 -87.649505        member
337271  41.984045 -87.660274        casual

[5733451 rows x 13 columns]
```

1.5 Process

1.5.1 Cleaning the data

In the started_at column, we need to check for any NaN values, or incomplete values.

[31]:
```
bike_data[bike_data["started_at"].isnull()]
```

[31]:
```
Empty DataFrame
Columns: [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]
Index: []
```

There are no null values.

1.5.2 Find duplicated data

Now we need to check to see if any rows are duplicates.

[32]:
```
bike_data[bike_data["ride_id"].duplicated()].value_counts().count()
```

[32]:
```
0
```

There are no rows with a duplicate ride id. Now we check for any duplicate rows.

[33]:
```
bike_data[bike_data.duplicated()]
```

[33]:
```
Empty DataFrame
Columns: [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]
Index: []
```

There are no duplicate rows.

1.5.3 Transforming the data

Create a column called "ride_length" which calculates the ride length. To do this, we take the column "ended_at" and subtract it by the column "ended_at". Our first step in creating this column is to convert the "started_at" and "ended_at" columns to a time format so we can perform this calculation.

[34]:
```
bike_data['started_at'] = pd.to_datetime(bike_data['started_at'], format="%Y/%m/
↪%d, %H:%M:%S")
```

[35]:
```
bike_data['ended_at'] = pd.to_datetime(bike_data["ended_at"], format = "%Y/%m/
↪%d, %H:%M:%S")
```

Now, perform the calculation.

[36]:
```
bike_data["ended_at"] - bike_data["started_at"]
```

[36]:
```
242975   0 days 00:23:15
182646   0 days 00:07:31
116503   0 days 00:02:25
229332   0 days 00:04:29
73536    0 days 00:04:00
              ...
180458   0 days 00:04:19
252640   0 days 00:04:37
```

```
177330   0 days 00:26:08
16514    0 days 00:04:18
337271   0 days 00:04:41
Length: 5733451, dtype: timedelta64[ns]
```

This is what we want, so we can now create a new column called "ride_length".

[37]:
```
bike_data["ride_length"] = bike_data["ended_at"] - bike_data["started_at"]
```

Create a column called "day_of_week" to calculate the day of the week that each ride started.

[38]:
```
bike_data["ride_length"].sort_values(ascending = True)
```

[38]:
```
417795   -8 days +19:26:39
154251   -1 days +21:11:18
585672   -1 days +21:42:35
126109   -1 days +21:49:58
301277   -1 days +21:50:55
                ...
626805    23 days 18:08:49
67464     23 days 20:34:04
610068    24 days 21:00:38
590027    25 days 04:17:48
405002    28 days 17:47:15
Name: ride_length, Length: 5733451, dtype: timedelta64[ns]
```

Convert the "started_at" column to a column which gives the day of the week.

[39]:
```
bike_data['started_at'].dt.day_name()
```

[39]:
```
242975    Wednesday
182646    Wednesday
116503    Wednesday
229332    Wednesday
73536     Wednesday
             ...
180458    Wednesday
252640    Wednesday
177330    Wednesday
16514     Wednesday
337271    Wednesday
Name: started_at, Length: 5733451, dtype: object
```

Let's name this column "day_of_week".

[40]:
```
bike_data["day_of_week"] = bike_data['started_at'].dt.day_name()
```

Here's the updated data frame.

[41]:
```
bike_data
```

[41]:

```
                ride_id  rideable_type          started_at  \
242975  937DC7C2109D635C  electric_bike 2021-12-01 00:00:01
182646  0CD83C3FE35E69A0   classic_bike 2021-12-01 00:00:03
116503  FC2D02B730EBC33D  electric_bike 2021-12-01 00:00:15
229332  227558BB46C7DE48  electric_bike 2021-12-01 00:01:00
73536   5CB387082B4310B2   classic_bike 2021-12-01 00:03:44
…                    …           …                   …
180458  CB5ECCA9D1567A45  electric_bike 2022-11-30 23:54:40
252640  DD58003A0A0C9FE4  electric_bike 2022-11-30 23:54:46
177330  0A7F3F32FCC736D1   classic_bike 2022-11-30 23:55:02
16514   C32D3E0585F38636  electric_bike 2022-11-30 23:55:28
337271  F6BE77C9DC9AB3BC  electric_bike 2022-11-30 23:56:11

                   ended_at       start_station_name start_station_id  \
242975 2021-12-01 00:23:16                      NaN              NaN
182646 2021-12-01 00:07:34     State St & Kinzie St            13050
116503 2021-12-01 00:02:40      Wells St & Huron St     TA1306000012
229332 2021-12-01 00:05:29                      NaN              NaN
73536  2021-12-01 00:07:44      Ellis Ave & 60th St     KA1503000014
…                      …                      …                …
180458 2022-11-30 23:58:59  State St & Van Buren St     TA1305000035
252640 2022-11-30 23:59:23                      NaN              NaN
177330 2022-12-01 00:21:10     McClurg Ct & Ohio St     TA1306000029
16514  2022-11-30 23:59:46  Broadway & Cornelia Ave            13278
337271 2022-12-01 00:00:52  Clark St & Winnemac Ave     TA1309000035

                     end_station_name end_station_id  start_lat  start_lng  \
242975                            NaN            NaN  41.930000 -87.730000
182646         St. Clair St & Erie St          13016  41.889187 -87.627754
116503                            NaN            NaN  41.894763 -87.634595
229332                            NaN            NaN  41.790000 -87.600000
73536        University Ave & 57th St   KA1503000071  41.785097 -87.601073
…                                 …              …         …         …
180458         Michigan Ave & 14th St   TA1307000124  41.877238 -87.627823
252640                            NaN            NaN  41.980000 -87.660000
177330  Sheffield Ave & Fullerton Ave   TA1306000016  41.892592 -87.617289
16514      Clarendon Ave & Gordon Ter          13379  41.945466 -87.646343
337271           Broadway & Ridge Ave          15578  41.973391 -87.667762

          end_lat    end_lng member_casual     ride_length day_of_week
242975  41.880000 -87.710000        casual 0 days 00:23:15   Wednesday
182646  41.894345 -87.622798        casual 0 days 00:07:31   Wednesday
116503  41.900000 -87.640000        member 0 days 00:02:25   Wednesday
229332  41.800000 -87.600000        member 0 days 00:04:29   Wednesday
73536   41.791478 -87.599861        member 0 days 00:04:00   Wednesday
…            …         …           …                 …           …
180458  41.864059 -87.623727        member 0 days 00:04:19   Wednesday
```

```
252640  41.970000 -87.650000        member 0 days 00:04:37   Wednesday
177330  41.925602 -87.653708        member 0 days 00:26:08   Wednesday
16514   41.957867 -87.649505        member 0 days 00:04:18   Wednesday
337271  41.984045 -87.660274        casual 0 days 00:04:41   Wednesday

[5733451 rows x 15 columns]
```

Now let's clean the data more. We need to remove any rows in which the ride length is less than or equal to zero seconds because a ride length less than zero does not make any sense in this context, and a ride length of zero seconds can't happen. The following command let's us drop these rows.

[48]:
```
bike_data = bike_data.drop(bike_data[~(bike_data["ride_length"] > "0 days")].
↪index)
```

[49]:
```
bike_data
```

[49]:
```
                ride_id  rideable_type          started_at  \
242975  937DC7C2109D635C  electric_bike 2021-12-01 00:00:01
182646  0CD83C3FE35E69A0   classic_bike 2021-12-01 00:00:03
116503  FC2D02B730EBC33D  electric_bike 2021-12-01 00:00:15
229332  227558BB46C7DE48  electric_bike 2021-12-01 00:01:00
73536   5CB387082B4310B2   classic_bike 2021-12-01 00:03:44
…                    …             …                   …
180458  CB5ECCA9D1567A45  electric_bike 2022-11-30 23:54:40
252640  DD58003A0A0C9FE4  electric_bike 2022-11-30 23:54:46
177330  0A7F3F32FCC736D1   classic_bike 2022-11-30 23:55:02
16514   C32D3E0585F38636  electric_bike 2022-11-30 23:55:28
337271  F6BE77C9DC9AB3BC  electric_bike 2022-11-30 23:56:11

                  ended_at       start_station_name start_station_id  \
242975 2021-12-01 00:23:16                      NaN              NaN
182646 2021-12-01 00:07:34     State St & Kinzie St            13050
116503 2021-12-01 00:02:40      Wells St & Huron St     TA1306000012
229332 2021-12-01 00:05:29                      NaN              NaN
73536  2021-12-01 00:07:44      Ellis Ave & 60th St     KA1503000014
…                      …                      …                …
180458 2022-11-30 23:58:59  State St & Van Buren St     TA1305000035
252640 2022-11-30 23:59:23                      NaN              NaN
177330 2022-12-01 00:21:10     McClurg Ct & Ohio St     TA1306000029
16514  2022-11-30 23:59:46  Broadway & Cornelia Ave            13278
337271 2022-12-01 00:00:52  Clark St & Winnemac Ave     TA1309000035

                    end_station_name end_station_id  start_lat  start_lng  \
242975                           NaN            NaN  41.930000 -87.730000
182646         St. Clair St & Erie St          13016  41.889187 -87.627754
116503                           NaN            NaN  41.894763 -87.634595
229332                           NaN            NaN  41.790000 -87.600000
73536        University Ave & 57th St   KA1503000071  41.785097 -87.601073
```

```
…                                 …            …         …         …
180458         Michigan Ave & 14th St   TA1307000124  41.877238 -87.627823
252640                            NaN            NaN  41.980000 -87.660000
177330  Sheffield Ave & Fullerton Ave   TA1306000016  41.892592 -87.617289
16514      Clarendon Ave & Gordon Ter          13379  41.945466 -87.646343
337271           Broadway & Ridge Ave          15578  41.973391 -87.667762

          end_lat    end_lng member_casual     ride_length day_of_week
242975  41.880000 -87.710000        casual 0 days 00:23:15   Wednesday
182646  41.894345 -87.622798        casual 0 days 00:07:31   Wednesday
116503  41.900000 -87.640000        member 0 days 00:02:25   Wednesday
229332  41.800000 -87.600000        member 0 days 00:04:29   Wednesday
73536   41.791478 -87.599861        member 0 days 00:04:00   Wednesday
…             …         …            …                …           …
180458  41.864059 -87.623727        member 0 days 00:04:19   Wednesday
252640  41.970000 -87.650000        member 0 days 00:04:37   Wednesday
177330  41.925602 -87.653708        member 0 days 00:26:08   Wednesday
16514   41.957867 -87.649505        member 0 days 00:04:18   Wednesday
337271  41.984045 -87.660274        casual 0 days 00:04:41   Wednesday

[5729103 rows x 15 columns]
```

[50]:
```
bike_data.day_of_week.value_counts()
```

[50]:
```
Saturday     921356
Thursday     853429
Friday       816810
Wednesday    816518
Tuesday      782121
Sunday       782045
Monday       756824
Name: day_of_week, dtype: int64
```

Our cleaned data set contains 5,729,103 rows.

1.6 Analyze

Export the data set into a CSV file, so that we can analyze it using SQL.

[51]:
```
bike_data.to_csv('bike_data.csv')
```

My SQL code along with its outputs can be found here.

To summarize our findings, we found that the average ride length is 19.43 minutes. Casual riders have an average ride length of 29.12 minutes while annual riders have an average ride length of 12.71 minutes. It seems that casual riders ride significantly longer than annual riders. Furthermore, the average ride lengths are the longest on weekends: 23.66 minutes on Saturday and 24.09 minutes on Sunday. The average ride length is the lowest on Wednesdays in which it is 16.39 minutes. The most number of rides occur on Saturday and the lowest number of rides happens on Mondays. The

total number of rides from Saturdays is 921,356 rides and for Mondays is 756,824.

1.7 Share

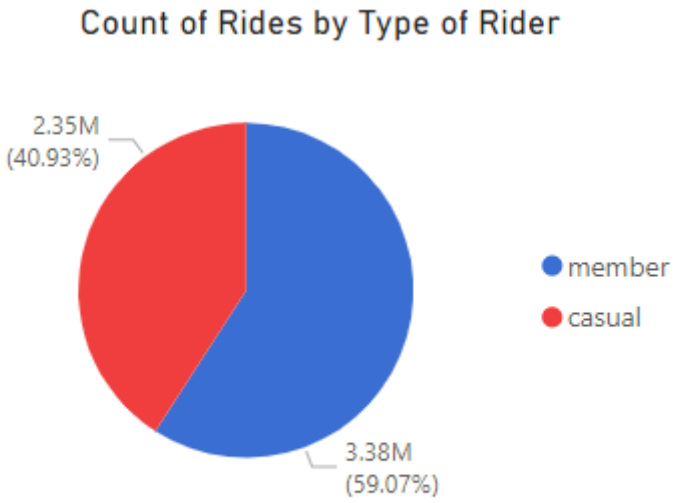
To learn more about our data and present it in an aesthetically pleasing way, we need to create visualizations. I used PowerBI to accomplish this task and my visualizations can be found here.

1.8 Act

We'll take a look at the visualizations to learn more about our data and find any trends. Note that the blue lines/bars depict the annual members while the red lines/bars depict the casual riders.

[5]:
```
from IPython.display import Image
Image(filename="Graph2.png", width = 400)
```

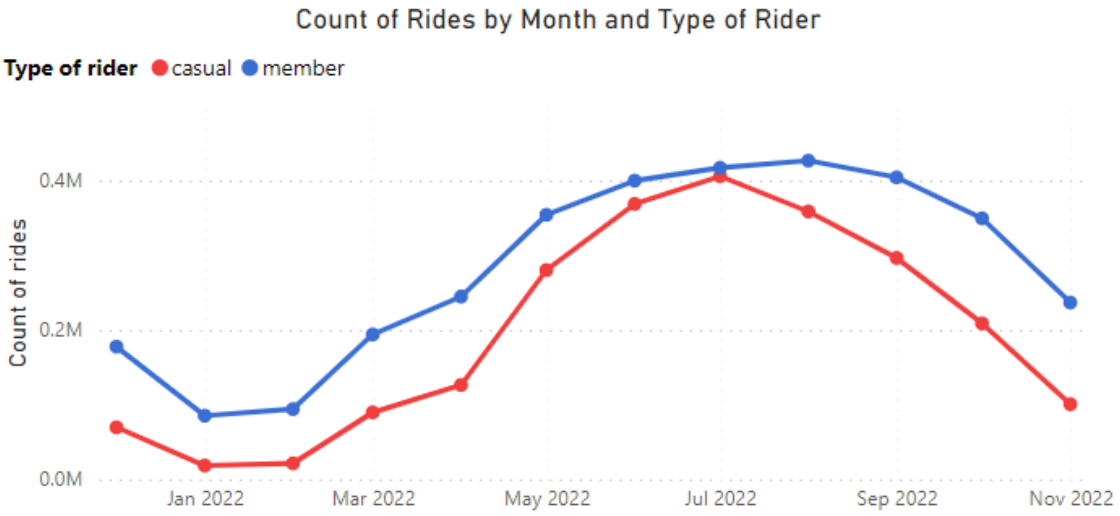
[5]:



We can see that 59.07% (3.38 million) of the riders have the annual membership, however, only 40.93% (2.35 million) of the riders are casual riders. This is important because there are still a lot of riders that we want to convert to annual memberships.

[1]:
```
from IPython.display import Image
Image(filename="Graph1.png", width = 800)
```

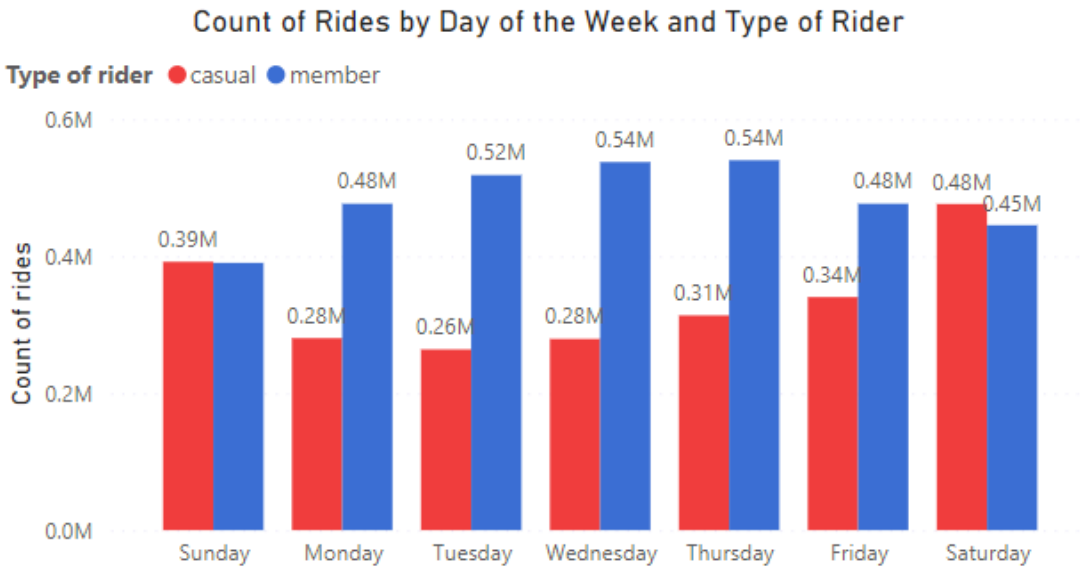
[1]:



The graph above shows the monthly count of rides for casual riders and members. The trends are similar for both type of riders; the count of rides start to increase during the spring and it peaks in the summer. After the summer, the count of rides drop. The count of rides is the lowest during the winter months, specifically during January and February. This makes sense because riders are more likely to prefer using bikes as opposed to walking when it is hot outside. We can also see that members ride more than casual riders.

[2]:
```
from IPython.display import Image
Image(filename="Graph3.png", width = 600)
```

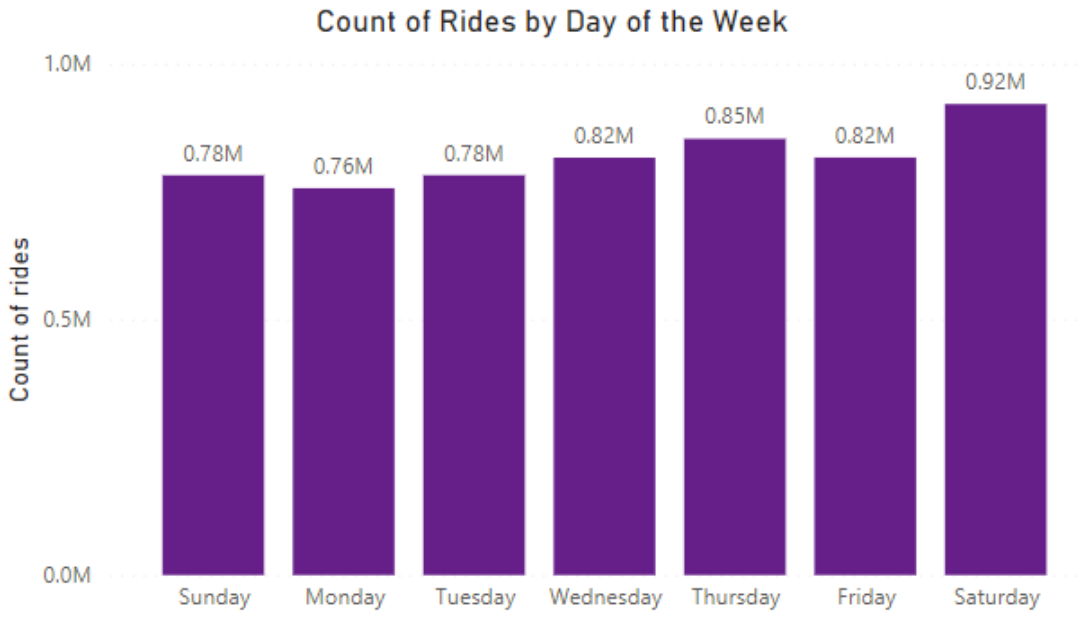
[2]:



This bar chart above shows the weekly count of rides for casual riders versus members. We see that members use the bikes more often during the weekday than weekends. The count of rides for members are the highest on Wednesday and Thursdays with 0.54 million rides. For casual riders, the count of rides is highest on weekends where there are 0.48 million rides on Saturday and 0.39 million on Sunday. The bar chart below shows the weekly count rides for all riders. The count of rides is highest on Saturday with 0.92 million total rides and the lowest count of rides is on Monday with 0.76 million total rides.

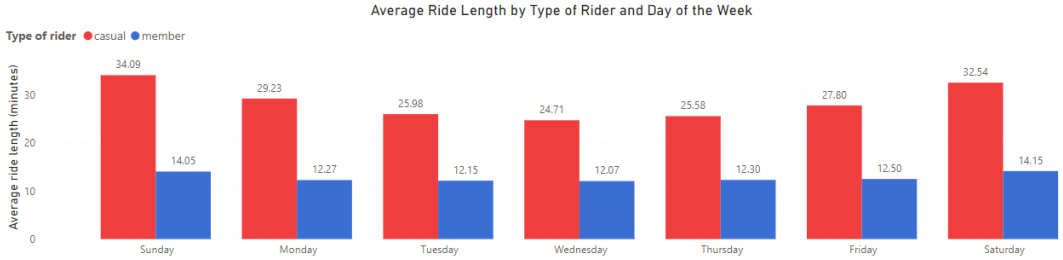
[1]:
```
from IPython.display import Image
Image(filename="Graph4.png", width = 600)
```

[1]:



[6]:
```
from IPython.display import Image
Image(filename="Graph5.png", width = 1000)
```

[6]:

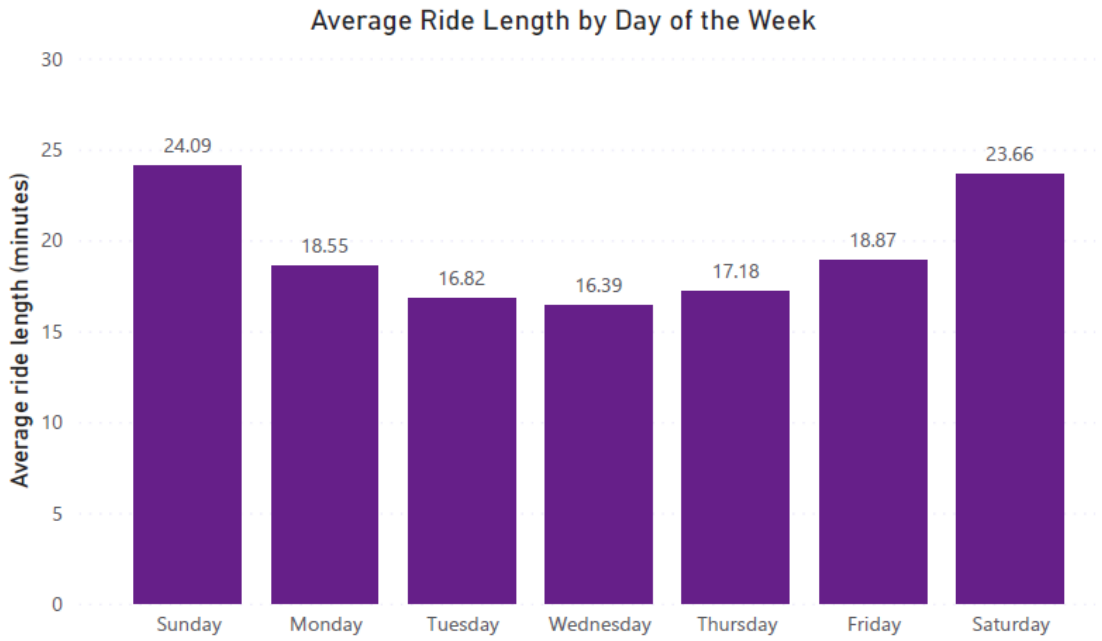


The above bar graph shows the weekly average ride length for both casual riders and members.

The average ride length for casual riders are approximately double the length as members. The trends for both casual riders and members are similar; the average ride length is high on weekends compared to weekdays. For casual riders, the longest average ride length happens on Sunday in which it is 34.09 minutes, while the longest average ride length for members is on Saturday in which it is 14.15 minutes. The day with the lowest average ride length is on Wednesday and this is the case for both type of riders. The bar graph below depicts the weekly average ride length for all members. It is not surprising that the ride length is highest on weekends. The average ride length decreases by 5.54 from Sunday to Monday, and the trend is the average ride length decreases until Wednesday, where it then increases until Sunday.

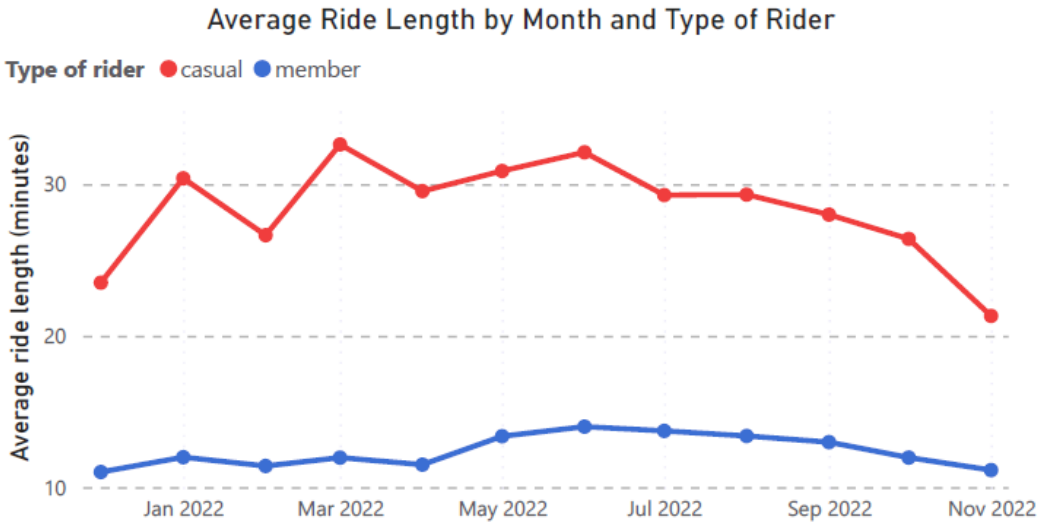
[4]:
```
from IPython.display import Image
Image(filename="Graph7.png", width = 700)
```

[4]:



[5]:
```
from IPython.display import Image
Image(filename="Graph6.png", width = 800)
```

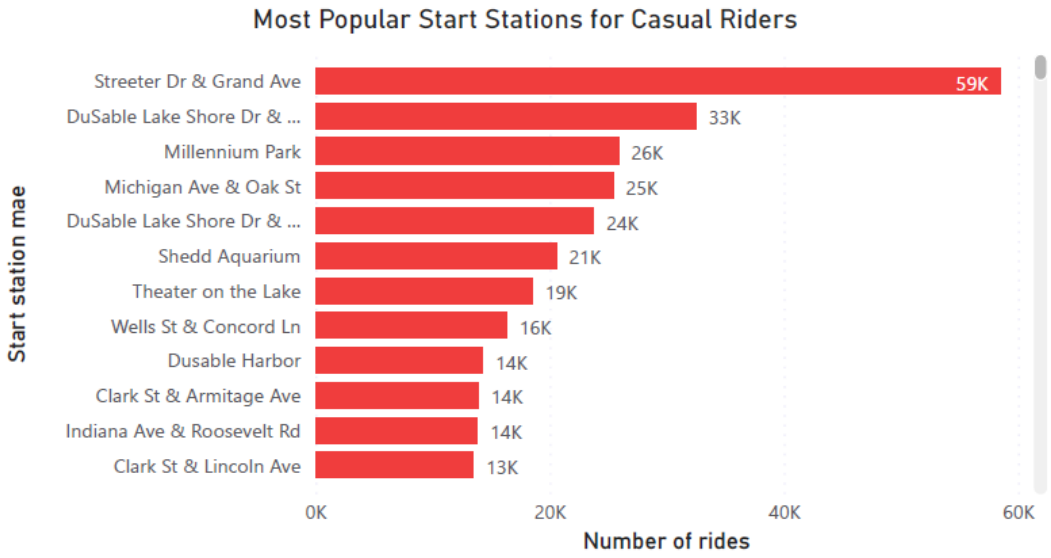
[5]:



The above line chart shows the monthly ride length for casual riders and members. The average ride length for casual riders is much higher than members. The average ride length for members doesn't change much throughout the year. There is a slight increase in ride length of 1.88 minutes from April 2022 to May 2022. After May 2022, the ride length peaks at June 2022, and it continues to slightly decrease. For casual riders, the average ride length is 23.50 minutes on December 2022 and increases to 30.39 minutes on January 2022. The ride length decreases in February but increases and peaks in March 2022 at 32.63 minutes. After March 2022, the ride length decreases to 29.54 minutes in April 2022 but increases to 32.11 minutes in June 2022. After June 2022, the average ride length drops and continues to decrease into the colder months.

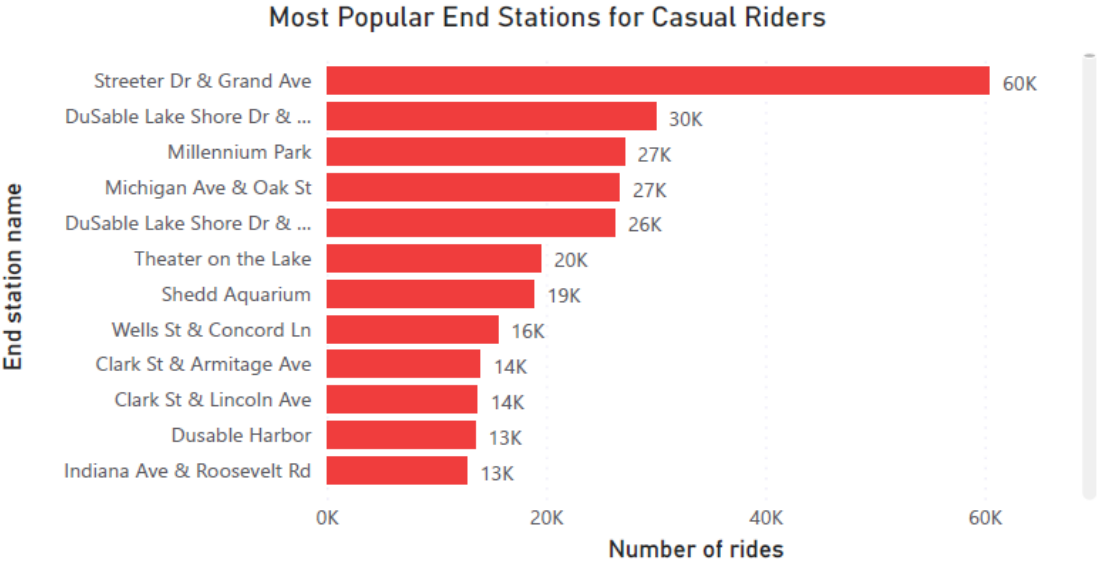
[34]:
```
from IPython.display import Image
Image(filename="Graph8.png", width = 600)
```

[34]:



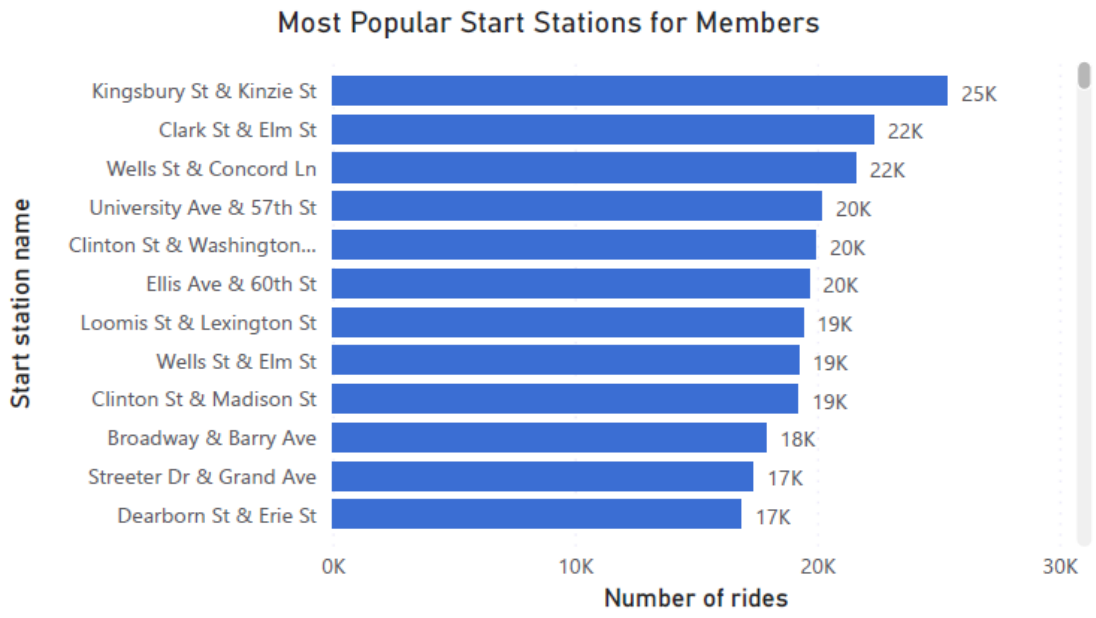
[35]:
```
from IPython.display import Image
Image(filename="Graph9.png", width = 600)
```

[35]:



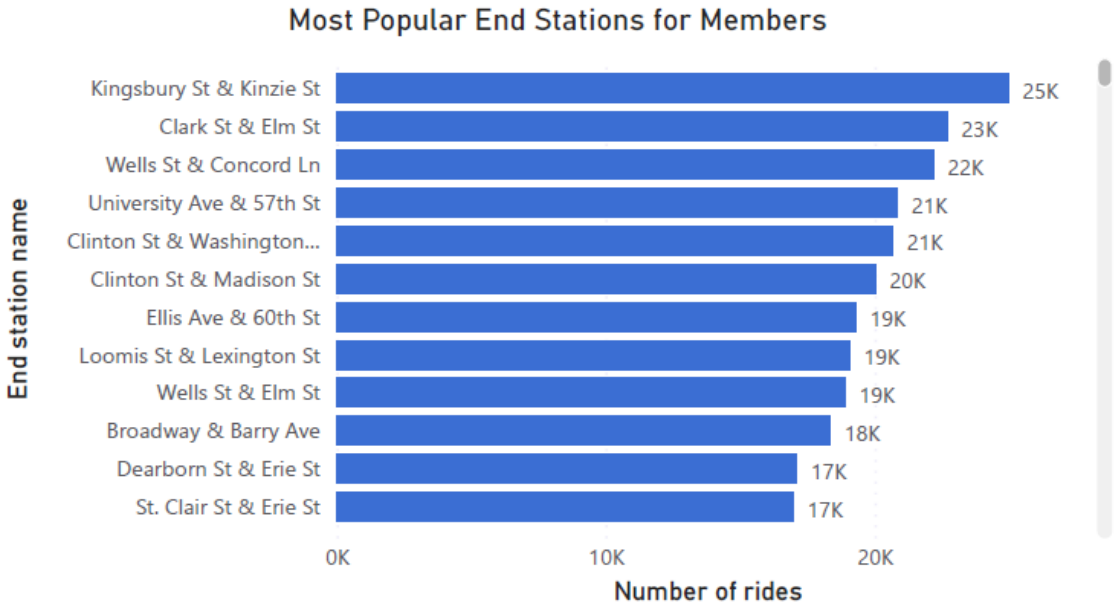
[36]:
```
from IPython.display import Image
Image(filename="Graph10.png", width = 600)
```

[36]:



[39]:
```
from IPython.display import Image
Image(filename="Graph11.png", width = 600)
```

[39]:

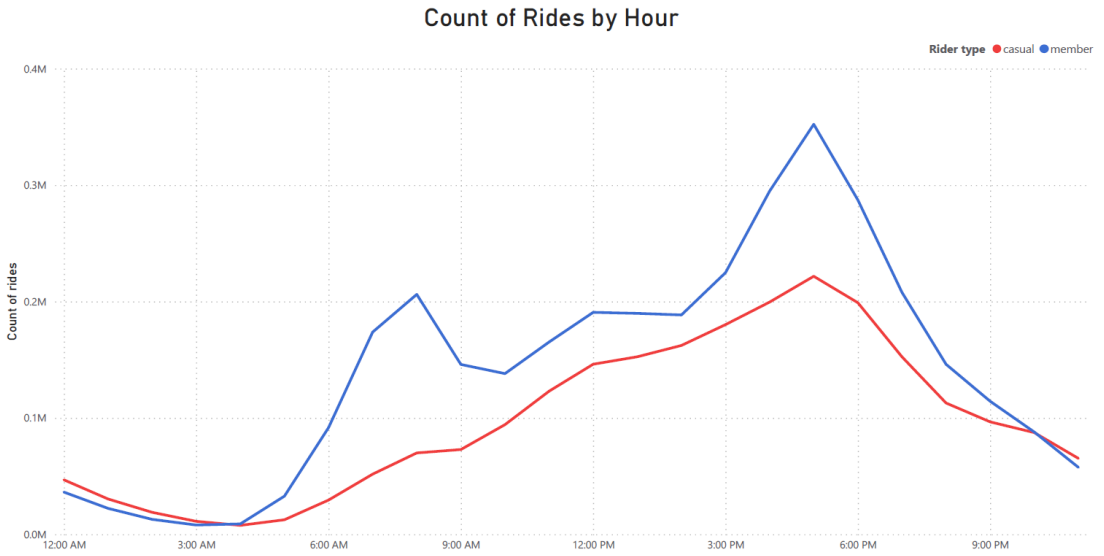


The charts above show the most popular stations for casual riders and members. We see that for casual riders, the most popular stations in which their ride starts is at Streeter Dr & Grand Avenue which accounted for about 58,544 rides. This is significantly higher than the other popular start

stations for casual riders, with the second most popular station being DuSable Lake Shore Dr & Monroe St with a ride count of 32,511. For members, the most popular start stations occur at Kingsbury St & Kinzie St in which it accounts for 25,377 rides.

[38]:
```
from IPython.display import Image
Image(filename="Graph12.png", width = 1000)
```

[38]:



The graph above shows us the hourly count of rides for both rider types. For members, the amount of rides peaks at 8 AM and has another peak at 5 PM. The trend for casual riders is a little different; the count of rides start to increase at 4 AM and continues to slowly increase until its peak hour at 5 PM. Both these trends make sense because the ride counts correspond to the start and end of the work day.

1.9 Conclusion and Recommendations

Now that we've gain some insight into our findings, we need to guide our team by creating a marketing strategy to maximize the amount of riders that purchase annual memberships.

1.9.1 1. Offer casual riders a special discount for purchasing membership before the summer months. Let customers choose the place their membership on hold.

The count of rides peaks during the summer months so offering a special promotion for the membership will encourage riders to purchase the membership. It would be a good time to offer this promotion in April because the count of rides for casual riders shows a huge increase after April. During the winter months, not many riders use the bikes so let memberships place their account on freeze so they don't unsubscribe.

1.9.2 2. By showing data, inform casual riders about how much money they can save if they convert to an annual membership plan.

The average ride times for casual riders are a lot higher compared to the average ride times for members. We should account for this fact in our advertisements and demonstrate how much money riders can save by purchasing an annual membership.

1.9.3 3. Partner with businesses, retailers, restaurants, or hold special events near the popular start/end stations.

Most Cyclistic bike rides start and end at the station located on Streeter Dr. & Grand Avenue. This area has heavy traffic so it partnering with businesses near this location can help boost sales for memberships since casual riders will likely visit these businesses and see the campaigns or advertisements.