Google Analytics Capstone project

by Julia Villalba

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

Introduction

The present document is the solution to a capstone project found in the Google Analytics Certificate on Coursera. All the data used here have been sourced from and are licensed by Google. This is the final assignment of the program, which includes an analysis and the formulation of solutions for a business task.

The scenario is described as follows: "A junior data analyst is 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, the team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, the team will design a new marketing strategy to convert casual riders into annual members. But first, Cyclistic executives must approve the recommendations, so they must be backed up with compelling data insights and professional data visualizations."

The dataset used for this project consists of 12 CSV files containing bike-share customer information from each month of 2022. The tools employed in this project are Excel, Tableau, SQL, and R. Excel was primarily used to explore the data sets on a monthly basis due to their size, while R was used for data cleaning, transformation, and analysis. SQL was invaluable for evaluating user differences and gaining insights. Tableau was selected for visualization, detecting outliers, and assessing data distributions due to its intuitive interface and comprehensive features.

We structured each phase of this project according to the 6 phases outlined in the Google certificate: Ask, Prepare, Process, Analyze, Share, and Act. We strived to adhere to these phases as much as possible, though there were some instances in which steps overlapped, such as process steps being included in the analysis phase for better readability. Nevertheless, we endeavored to abide by best practices at all times and ensured that analysis was never conducted before data cleaning.

The Ask phase: A clear statement of the business task

Since the company's future success depends on maximizing the number of annual memberships, the team wants to understand how casual riders and annual members use Cyclistic bikes differently, and obtain insights into why casual riders would buy a membership. Thus the first question to answer is: How do annual members and casual riders use Cyclistic bikes differently?

Business task: Identify and analyze patterns in the data to gain insights into how to attract and retain more annual members.

The Prepare phase: A description of all data sources used

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. The program is flexible in 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.

The data has been made available by Motivate International Inc. under this license

https://ride.divvybikes.com/data-license-agreement

You can find the data here

https://divvy-tripdata.s3.amazonaws.com/index.html

The tables used in this project are from January 2022 to December 2022. Next, we will give a detailed explanation about the variables and observations on these tables.

The Data

The data in question comprises historical trip data spanning the period of January 2022 to December 2022, and, upon inspection of the 13 variables and over one hundred thousand observations per table with the aid of a .txt file of associated metadata found through the provided link, we were able to determine the purpose of each variable:

1. ride_id: ID attached to each ride taken. Data type: char.
2. rideable_type: refers to the type of vehicle the rider was using, there are three options; "electric bike", "classic bike", "docked bike". Data type: char.
3. started_at: refer to the time and date when the trip started. Data type: POSIXct, which stores date and time in seconds.

4. ended_at: refer to the time and date when the trip ended. Data type: POSIXct.
5. start_station_name: refer to the name of the start station. Data type: char.
6. start_station_id: refer to the unique ID of the start station. Data type: char.
7. end_station_name: refer to the name of the end station. Data type: char.
8. end_station_id: refer to the unique ID of the end station. Data type: char.
9. start_lat: refer to the latitude coordinates for the start point of the trip. Data type: numeric.
10. start_lng: refer to the longitude coordinates for the start point of the trip. Data type: numeric.
11. end_lat: refer to the latitude coordinates for the end point of the trip. Data type: numeric.
12. end_lng: refer to the longitude coordinates for the end point of the trip. Data type: numeric.
13. member_casual: “casual" is a rider who purchased a 24-Hour Pass; "member" is a rider who purchased an Annual Membership. Data type: char.

The Process phase: Documentation of any cleaning or manipulation of the data

In this stage, we checked for outliers, verified accuracy of data, analyzed missing values, grouped and summarized data, and validated the overall data integrity. Then, reviewed the data again after cleaning to make sure all the changes made were correct, manually checking it against the original source. The following data cleaning is done in R.

Standardizing formats

To ensure the consistency of column names across all tables, we compared each column name from tables 2-12 with those in table 1, which served as the benchmark for the correct column names.

```
names_01 <- names(X2022_01) # assign column names of table 1 to the variable names_01

all(names_01 == names(X2022_02)) # do this for every table from 2 to 12
```

We identified that all tables had identical column names and thus the tables were able to be merged without issue. The tables were combined using the rbind() function.

```
full_year <- rbind(X2022_01,X2022_02, X2022_03, X2022_04, X2022_05, X2022_06,

                   X2022_07, X2022_08, X2022_09, X2022_10, X2022_12,

                   X2022_12)
```

This merged dataset allows for the comprehensive analysis of all data in one go, as well as allowing for further investigation using external programs such as SQL.

Duplicates

We look for duplicates using the function get_dupes from the library “janitor”.

```
library(janitor)
full_year %>% get_dupes(ride_id)
```

A tibble: 363,612 x 14

ride_id <chr>	dupe_count <int>	rideable_type <chr>	started_at <S3: POSIXct>
0000038F578A7278	2	electric_bike	2022-12-15 15:36:06
0000038F578A7278	2	electric_bike	2022-12-15 15:36:06
00015AF8541BB982	2	electric_bike	2022-12-02 15:28:53
00015AF8541BB982	2	electric_bike	2022-12-02 15:28:53
0001C99FAE606D71	2	classic_bike	2022-12-04 18:49:13
0001C99FAE606D71	2	classic_bike	2022-12-04 18:49:13
00023D2D08E6C0FA	2	electric_bike	2022-12-26 05:30:55
00023D2D08E6C0FA	2	electric_bike	2022-12-26 05:30:55
0002BCC09C034FC9	2	classic_bike	2022-12-16 12:12:05
0002BCC09C034FC9	2	classic_bike	2022-12-16 12:12:05

1-10 of 363,612 rows | 1-4 of 14 columns Previous 1 2 3 4 5 6 … 100 Next

We see that there are 363,612 repeated rows, that means that we must remove half of them in order to get only distinct values. We do this using the function distinct().

```
full_year <- full_year %>% distinct()
```

A tibble: 5,329,982 x 13

ride_id <chr>	rideable_type <chr>	started_at <S3: POSIXct>	ended_at <S3: POSIXct>
C2F7DD78E82EC875	electric_bike	2022-01-13 11:59:47	2022-01-13 12:02:44
A6CF8980A652D272	electric_bike	2022-01-10 08:41:56	2022-01-10 08:46:17
BD0F91DFF741C66D	classic_bike	2022-01-25 04:53:40	2022-01-25 04:58:01
CBB80ED419105406	classic_bike	2022-01-04 00:18:04	2022-01-04 00:33:00
DDC963BFDDA51EEA	classic_bike	2022-01-20 01:31:10	2022-01-20 01:37:12
A39C6F6CC0586C0B	classic_bike	2022-01-11 18:48:09	2022-01-11 18:51:31
BDC4AB637EDF981B	classic_bike	2022-01-30 18:32:52	2022-01-30 18:49:26
81751A3186E59A6B	classic_bike	2022-01-22 12:20:02	2022-01-22 12:32:06
154222B86A338ABD	electric_bike	2022-01-17 07:34:41	2022-01-17 08:00:08
72DC25B2DD467EEF	classic_bike	2022-01-28 15:27:53	2022-01-28 15:35:16

1-10 of 5,329,982 rows | 1-4 of 13 columns Previous 1 2 3 4 5 6 … 100 Next

We have 5,329,982 distinct rows. We can check that the calculations are correct:
5511788 - 5329982 = 181806
And 181806 * 2 = 363612, which were the total duplicates.

Data Validation

The maximum range of dates and times that are valid for this data set is from 2022/01/01 to 2022/12/31. We can check that with the max() and min() functions.

```
min(full_year$started_at, na.rm = TRUE)
max(full_year$started_at, na.rm = TRUE)
min(full_year$ended_at, na.rm = TRUE)
max(full_year$ended_at, na.rm = TRUE)
```

```
[1] "2022-01-01 00:00:05 UTC"
[1] "2022-12-31 23:59:26 UTC"
[1] "2022-01-01 00:01:48 UTC"
[1] "2023-01-02 04:56:45 UTC"
```

This indicates that all data pertaining to the starting time of rides is within the acceptable range. It is possible that data regarding the ending time may lie beyond this range, however, as long as the starting date is included in the range, the data can be regarded as valid. We show some examples:

```
full_year %>% filter(ended_at > "2022-12-31 23:59:59")
```

A tibble: 13 x 13

ride_id <chr>	rideable_type <chr>	started_at <S3: POSIXct>	ended_at <S3: POSIXct>
BBFAB02ADC8AE878	classic_bike	2022-12-31 13:50:56	2023-01-01 14:50:35
389D2E086B74585B	classic_bike	2022-12-31 21:13:00	2023-01-01 22:12:52
96CEA4611988B3E3	classic_bike	2022-12-31 23:41:15	2023-01-02 00:41:09
CB394CE836D40EC6	docked_bike	2022-12-28 13:23:36	2023-01-02 04:56:45
89FCD1655DF56946	docked_bike	2022-12-31 23:26:21	2023-01-02 00:26:22
6CC672888499E23E	docked_bike	2022-12-31 22:14:55	2023-01-01 15:52:26
2935ED131B8EA6CA	docked_bike	2022-12-31 23:27:02	2023-01-01 12:27:35
7BEB6DBE09F874DB	docked_bike	2022-12-29 15:29:23	2023-01-02 04:26:03
E6D8FC98D6C8CD8B	classic_bike	2022-12-31 13:14:01	2023-01-01 14:13:56
912F2F3E7CF2C93D	classic_bike	2022-12-31 23:10:34	2023-01-02 00:10:29

1-10 of 13 rows | 1-4 of 13 columns Previous 1 2 Next

Missing values

Without access to the creator of the data set, several approaches must be taken in order to address potential missing values. It is important to note that a given row does not need to be completely eliminated simply due to a missing value in one column; instead, each field should be individually analyzed before a decision is made as to how best to proceed.

Initially, we need to determine how many NAs exist and in which fields they are located. It can be seen that several NAs reside in the fields of start_station_name, start_station_id, end_station_id, end_lat and end_lng.

```
sum(is.na(full_year$start_station_name))
[1] 781107
sum(is.na(full_year$start_station_id))
[1] 781107
sum(is.na(full_year$end_station_name))
[1] 838483
sum(is.na(full_year$end_station_id))
[1] 838483
sum(is.na(full_year$end_lat))
[1] 5628
sum(is.na(full_year$end_lng))
[1] 5628
```

There are no missing values in the remaining fields; therefore, if we do not need to compare any of these fields with the NAs in their respective fields, we can disregard them. However, if we wish to compare such data points, as members vs casuals and start stations vs end destinations, we may wish to remove any rows which contain NAs, as they would not add any significant value to our analysis.

Outliers

One easy way to get the full picture of outliers or wrong data is by visualizing it. Tableau has a unique advantage over many other viz tools because it can show geographic data very easily.

We can check that this outlier corresponds to

```
X2022_01 %>% filter(round(start_lat,3) == 45.635)
```

ride_id <chr>	rideable_type <chr>	started_at <S3: POSIXct>	ended_at <S3: POSIXct>
3327172413547F64	electric_bike	2022-01-14 11:13:15	2022-01-14 11:15:50

This is likely to be an erroneous input of coordinates, given the exceedingly short distance between the "ended at" and "started at" points. Consequently, it would be prudent to eliminate this row.

```
full_year <- full_year %>% filter(ride_id != '3327172413547F64')
```

We can export this merged dataset for further in-depth analysis and investigation using the write.csv() function of R.

Exploration using Excel

Although the number of steps that can be executed in Excel are limited in this instance, individual datasets from each month may be read in their respective Excel sheets, allowing for a degree of analysis to be done one month at a time. As an example, we shall discuss the 2022/12 data set and share some steps which were taken. The "ride_length" column was created and the length of each ride was calculated through subtracting the time in the "started_at" column from that of the "ended_at" column, formatted in HH:MM:SS. A second column titled "day_of_week" was then generated, indicating the day of the week that each ride started using the WEEKDAY function. Subsequently, Pivot Tables were employed to run the following calculations:

- Calculating the mean and maximum ride lengths for members and casuals.

Row Labels	Average of ride_length	Max of ride_length
casual	0:22:17	319:17:06
member	0:10:37	24:59:56

indicating that there was a journey that spanned multiple days:

ride_id	rideable_type	started_at -	ended_at	start_station_name	start_station_id	e
F5F074FB092219CD	docked_bike	12/2/2022 21:41	12/16/2022 4:58	Larrabee St & Division St	KA1504000079	

- Calculating the average and mode of day_of_week for members and casuals.

Average of ride_length	Column Labels							
Row Labels	1	2	3	4	5	6	7	Grand Total
casual	0:18:17	0:15:19	0:24:13	0:19:42	0:29:04	0:23:18	0:25:07	0:22:17
member	0:10:29	0:10:20	0:10:23	0:10:24	0:10:54	0:11:17	0:10:51	0:10:37
Grand Total	**0:12:06**	**0:11:23**	**0:13:26**	**0:12:33**	**0:15:44**	**0:15:03**	**0:15:13**	**0:13:30**

- To calculate the number of rides per user per day of the week, the Count of trip_id was added to the Values.

	Column Labels							
	1	2	3	4	5	6	7	Grand Total
Count of ride_id	571800:00:00	672600:00:00	620160:00:00	846528:00:00	643632:00:00	589584:00:00	419040:00:00	4363344:00:00

Exploration using SQL

SQL is a powerful tool for dealing with large volumes of data; with it, we are able to do what would have been impossible with Excel. In this case, we are able to import the entire year's dataset and glean various insights from it. This has many benefits, such as the ability to work with a much larger data set, group by certain fields and conduct comparisons based on the entire year's worth of data. We make use of the PostgreSQL database management system for this segment. Prior to this, we have eliminated duplicates from a csv file containing all months merged. As a starting point, a table is created with all the relevant fields.

```
CREATE TABLE fullyear(

ride_id VARCHAR(50) NOT NULL PRIMARY KEY,

rideable_type VARCHAR(50),

started_at TIMESTAMP,

ended_at TIMESTAMP,

start_station_name VARCHAR(100),

start_station_id VARCHAR(100),

end_station_name VARCHAR(100),

end_station_id VARCHAR(100),

start_lat float(50),

start_lng float(50),

end_lat VARCHAR(100),

end_lng VARCHAR(100),

member_casual VARCHAR(50))
```

The TIMESTAMP data type with time zone was selected for started_at and ended_at fields, while the VARCHAR data type was chosen for the fields of end_lng and end_lat due to the presence of NA values which were strings. The NA strings were subsequently replaced with NULLs, allowing for these fields to be altered to have a float data type.

```
UPDATE fullyear SET end_lat = NULL WHERE end_lat = 'NA'

ALTER TABLE public.fullyear

       ALTER COLUMN end_lat TYPE float(50)
```

```
USING end_lat::double precision;
```

Now we write some queries:

```
SELECT COUNT(*) AS number_members

FROM fullyear

WHERE member_casual = 'member'
```

	number_members bigint
1	3108722

This shows that there are 41,67% of casual riders and 58,33% of members.

Comparing the average trip time between members and casual riders, exploring the preferred start and end locations for each type of rider, examining the days of the week which see the most trips taken by each type of rider and comparing the types of rideable vehicles used by members and casual riders are all activities that this investigation will cover.

```
SELECT member_casual,

AVG(ended_at - started_at) as avg_trip_duration,

MAX(ended_at - started_at) as max_trip_duration,

MIN(ended_at - started_at) as min_trip_duration

FROM fullyear

GROUP BY member_casual
```

	member_casual character varying (50)	avg_trip_duration interval	max_trip_duration interval	min_trip_duration interval
1	casual	00:29:30.047465	28 days 17:47:15	-02:17:25
2	member	00:12:49.859495	1 day 01:59:54	-7 days -04:33:21

We have noted the presence of negative values, which can be attributed to entries in the ended_at column possessing date and times prior to those in the started_at column. As this information is likely to be inaccurate, it would be best to remove these rows from the data set. Additionally, the maximum trip duration time recorded stands at 28 days, though we do not have sufficient information on the upper limit on the amount of time a user is permitted to rent the same bike.

Afterwards, we can filter out rides where ended_at is greater than started_at.

```
SELECT member_casual,

AVG(ended_at - started_at) as avg_trip_duration,

MAX(ended_at - started_at) as max_trip_duration,

MIN(ended_at - started_at) as min_trip_duration

FROM fullyear

WHERE ended_at > started_at

GROUP BY member_casual
```

	member_casual character varying (50)	avg_trip_duration interval	max_trip_duration interval	min_trip_duration interval
1	casual	00:29:30.255082	28 days 17:47:15	00:00:01
2	member	00:12:50.12807	1 day 01:59:54	00:00:01

Nevertheless, we still find several values that do not appear to be logical, like trips that only lasted one second. This could be due to trips being canceled shortly after being requested. To gain a better understanding of these fields, we shall examine their distribution in further detail, as taking extreme values into consideration may result in inaccurate readings concerning the average, maximum and minimum values. To ascertain how stations are distributed between members and casuals, the following query is executed.

```
SELECT member_casual, COUNT(DISTINCT(start_station_name))

FROM fullyear

GROUP BY member_casual
```

	member_casual character varying (50)	count bigint
1	casual	1580
2	member	1481

They are roughly evenly distributed. Next, we look for the most common stations.

```
SELECT member_casual, start_station_name,  COUNT(*)

FROM fullyear

GROUP BY member_casual, start_station_name
```

```
ORDER BY COUNT(*) DESC
```

	member_casual character varying (50)	start_station_name character varying (100)	count bigint
1	member	NA	450680
2	casual	NA	330427
3	casual	Streeter Dr & Grand Ave	56363
4	casual	DuSable Lake Shore Dr & Monroe St	30698
5	casual	Michigan Ave & Oak St	24661
6	casual	Millennium Park	24623
7	casual	DuSable Lake Shore Dr & North Blvd	23289

Unfortunately, there are several stations with missing values. We have to consider the third and fourth rows as the most popular stations. We do the same for the end stations:

	member_casual character varying (50)	end_station_name character varying (100)	count bigint
1	member	NA	449100
2	casual	NA	389383
3	casual	Streeter Dr & Grand Ave	57924
4	casual	DuSable Lake Shore Dr & Monroe St	28608
5	casual	Michigan Ave & Oak St	25749
6	casual	DuSable Lake Shore Dr & North Blvd	25737
7	casual	Millennium Park	25613

Nex, we want to check what are the most common days for rides.

```
SELECT EXTRACT(DOW FROM started_at) AS DayofWeek,

Member_casual, COUNT(*)

FROM  fullyear1

GROUP BY EXTRACT(DOW FROM started_at), member_casual

ORDER BY COUNT(*) DESC
```

	dayofweek numeric	member_casual character varying (50)	count bigint
1	4	member	493136
2	3	member	476421
3	2	member	472512
4	6	casual	461321
5	1	member	441034
6	5	member	436762
7	6	member	422841
8	0	casual	376539
9	0	member	366016
10	5	casual	320163
11	4	casual	291349
12	1	casual	267375
13	3	casual	256575
14	2	casual	247938

Total rows: 14 of 14 Query complete 00:00:02.440

This query will return the day of the week (where 0 is Sunday), the type of rider (member or casual) and the total number of rides for each day, divided into member and casual groups. This results in a total of 14 rows.

It can be seen that Saturday is the most popular day for casual riders, the second most popular is Sunday, while Tuesday is the least favored. For members, the days most frequently used for rides tend to be during weekdays, while weekends are less popular.

The Analysis phase: A summary of the analysis

Analysis using R

Due to its versatility, the programming language R is employed for both data transformation and analysis in this section. This language features a wide range of statistical functions and packages, making it an optimal choice when dealing with large datasets.

First, we check how members and casual use different types of bikes

```
full_year %>% group_by(member_casual, rideable_type) %>%
  summarize(unique(member_casual))
```

A tibble: 5 x 3 Groups: member_casual [2]

member_casual <chr>	rideable_type <chr>	unique(member_casual) <chr>
casual	classic_bike	casual
casual	docked_bike	casual
casual	electric_bike	casual
member	classic_bike	member
member	electric_bike	member

5 rows

With such a large sample size (3108722), it is highly unlikely that the sample would contain only members who have chosen classic and electric bikes due to randomness or chance alone. So it's safe to assume that members only have access to electric and classic bikes and not to docked. Therefore there might be a group of customers in the casual category who don't want to become members because of their preference of docked bikes.

We create a new column which computes the duration of each trip:

```
full_year <- full_year %>%
  mutate(ride_time = ended_at - started_at)
```

We saw before that there are some entries with negative values because there are rows where ended_at < started_at, so we remove these rows:

```
full_year <- full_year %>% filter(ended_at > started_at)
```

start_lat <dbl>	start_lng <dbl>	end_lat <dbl>	end_lng <dbl>	member_casual <chr>	ride_time <time>
41.87502	-87.63309	41.87502	-87.63309	member	1 secs
41.84720	-87.64679	41.84720	-87.64679	member	1 secs
41.88184	-87.64079	41.88224	-87.64107	member	1 secs
42.00447	-87.67235	42.00445	-87.67240	member	1 secs
41.90941	-87.67769	41.90940	-87.67769	member	1 secs
41.96649	-87.68872	41.96640	-87.68870	member	1 secs
41.86723	-87.62596	41.86723	-87.62596	member	1 secs
41.89598	-87.66781	41.90000	-87.67000	casual	1 secs
41.86481	-87.64711	41.86000	-87.65000	member	1 secs

Despite removing the negative values from ride_time, there are still a number of trips that have very short duration times. To gain an understanding of the distribution of this field, we can use the quantile() function:

```
quantile(full_year$ride_time)
```

```
Time differences in secs
     0%     25%     50%     75%    100%
      1     354     627    1126 2483235
```

The numbers provided by the quantile function represent the cut-off points for each percentile. For example, for the ride_time column, the first value (1) is the 0th percentile and represents the minimum time difference in seconds. The second value (354) is the 25th percentile and represents the time difference in seconds for the 25th percentile. The third value (627) is the 50th percentile and represents the median time difference in seconds. The fourth value (1126) is the 75th percentile and represents the time difference in seconds for the 75th percentile. Lastly, the fifth value (2483235) is the 100th percentile and represents the maximum time difference in seconds.

We can look at other percentiles that show more extreme values:

```
quantile(full_year$ride_time, probs=c(0, 0.1, 0.8, 0.9))
```

```
Time differences in secs
  0%  10%  80%  90%  95%  98%
   1  209 1308 1949 2770 4576
```

This analysis can assist in finding a more accurate representation of average ride times. The maximum ride time is far from the 98th percentile, indicating that outliers are skewing the situation. Consequently, utilizing the trimmed mean function of R can give us a more dependable result. To maintain a level of caution, we shall only exclude 5% of the data.

```
mean(full_year$ride_time, 0.05)
```

```
Time difference of 795.1731 secs
```

We can summarize this by member_casual

```
full_year2 %>%
  group_by(member_casual) %>%
  summarize(quantile(ride_time, probs=c(0, 0.1, 0.8, 0.9, 0.95, 0.98)))
```

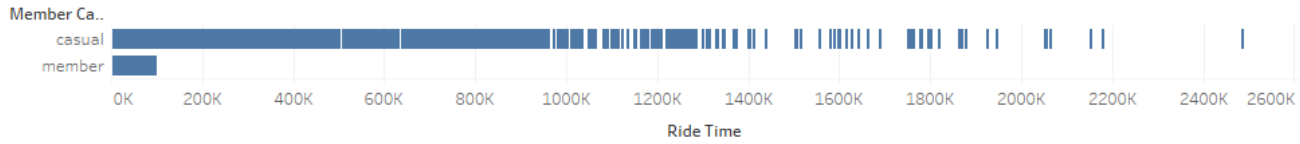
member_casual <chr>	quantile(ride_time, probs = c(0, 0.1, 0.8, 0.9, 0.95, 0.98)) <time>
casual	1 secs
casual	261 secs
casual	1722 secs
casual	2741 secs
casual	4165 secs
casual	6645 secs
member	1 secs
member	187 secs
member	1062 secs
member	1508 secs

This shows that casual riders have more variability in ride duration.
Now, we exclude the 5% of data only for casuals, since members have a more constant ride time:

```
casual <- full_year %>% filter(member_casual == 'casual')
mean(casual$ride_time, trim = 0.05)
```

```
Time difference of 1051.963 secs
```

From the graph, it can be observed that casual riders usually take longer rides, likely due to the fact that they mainly use bike share for recreational purposes. This is further supported by their most favored day for rides being on weekends. Conversely, members demonstrate a less varied ride length, likely since they often use bike share for commuting or other goal-oriented tasks (with weekdays being their most common day for rides). This theory appears to be supported. According to data from the U.S. Bureau of Labor Statistics, the majority of people in Chicago work on weekdays, thereby lending further credence to the notion that members primarily use bicycle shares for commuting purposes.
For further information please check this link https://www.bls.gov/cps/cpsaat25.htm



Next, we want to find out which stations are the most popular. We create a column with the most popular stations:

```
popular_start_stations <- full_year2 %>%

  group_by(start_station_name) %>%

  mutate(visit_count = n())
```

We create a data frame containing the stations with visits exceeding 25,000, by casual riders. This number was determined through our SQL analysis, and is an arbitrary figure that will ultimately depend on the company's goals.

```
start_above_25000 <- popular_start_stations %>%

  group_by(start_station_name) %>%
```

```
  filter(n() > 25000 & member_casual == 'casual')
```

Then we remove rows with NA values

```
start_above_25000 <-  start_above_25000 %>%

  filter(!is.na(start_station_name))

start_above_25000 %>%

  arrange(desc(visit_count))
```

This filters the 15 start stations most popular. We checked that out using the unique() function.

```
 [1] "DuSable Lake Shore Dr & North Blvd" "Millennium Park"
 [3] "Wells St & Concord Ln"              "Wilton Ave & Belmont Ave"
 [5] "Theater on the Lake"                "Indiana Ave & Roosevelt Rd"
 [7] "Wabash Ave & Grand Ave"             "DuSable Lake Shore Dr & Monroe St"
 [9] "Streeter Dr & Grand Ave"            "Clark St & Elm St"
[11] "Broadway & Barry Ave"               "Clark St & Armitage Ave"
[13] "Michigan Ave & Oak St"              "Kingsbury St & Kinzie St"
[15] "Wells St & Elm St"                  "Clark St & Lincoln Ave"
```

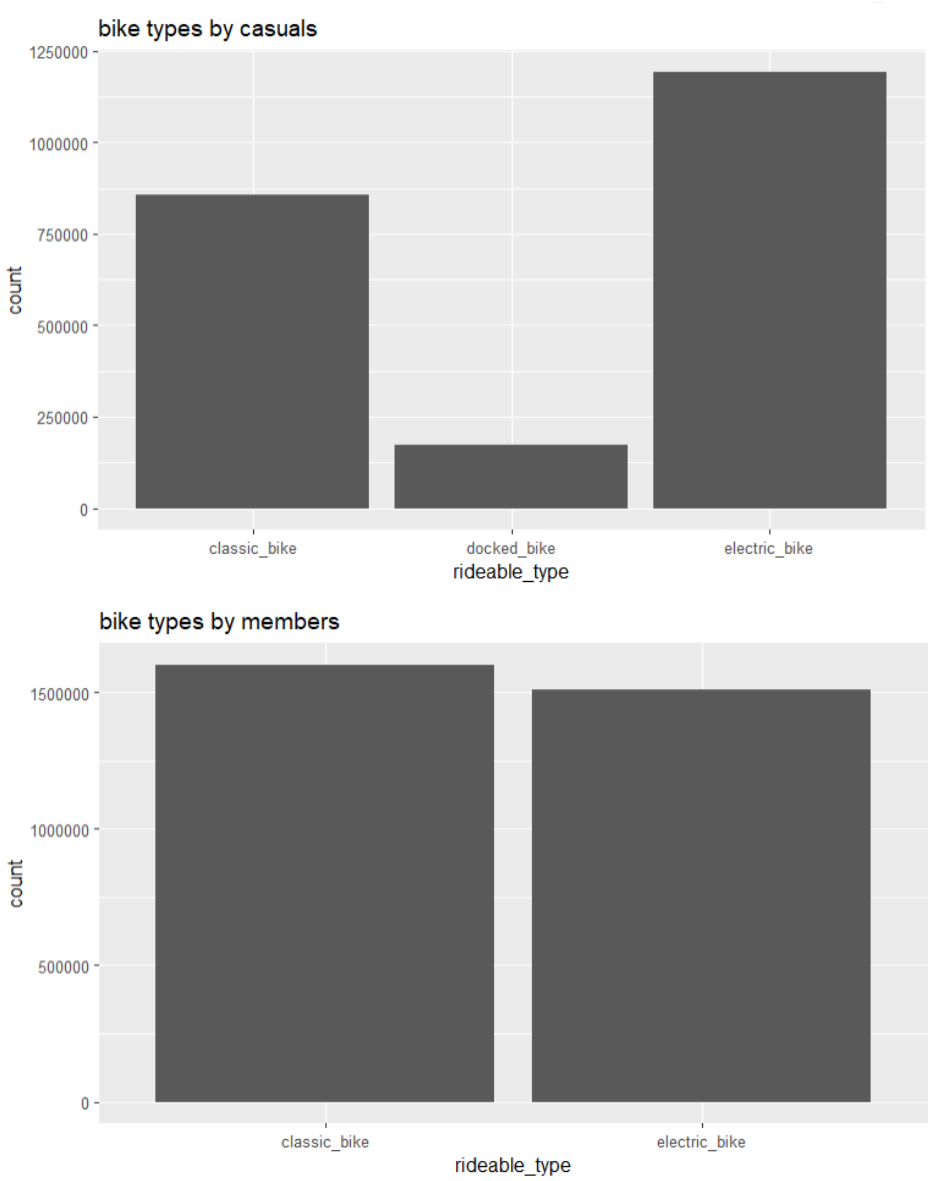
We do the exact same for the end stations. We get this most popular 15 stations:

```
 [1] "Clark St & Elm St"                "Wabash Ave & Grand Ave"
 [3] "Millennium Park"                  "Theater on the Lake"
 [5] "Wells St & Elm St"                "Streeter Dr & Grand Ave"
 [7] "DuSable Lake Shore Dr & Monroe St" "DuSable Lake Shore Dr & North Blvd"
 [9] "Wells St & Concord Ln"            "Wilton Ave & Belmont Ave"
[11] "Clark St & Lincoln Ave"           "Broadway & Barry Ave"
[13] NA                                 "Kingsbury St & Kinzie St"
[15] "Michigan Ave & Oak St"            "Clark St & Armitage Ave"
```

The Share phase: Supporting visualizations and key findings

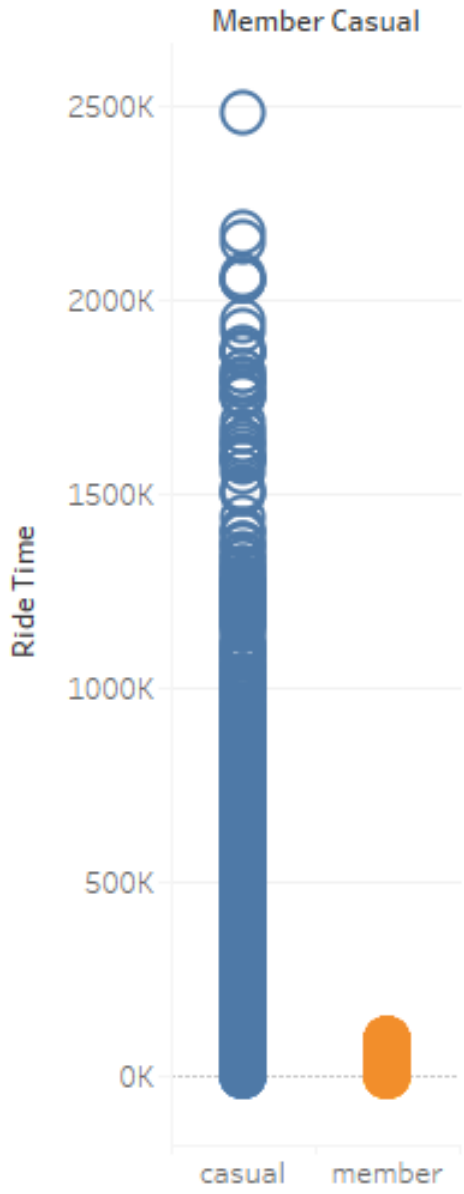
A percentage of casual riders have preference for docked bikes

An important discovery is that members likely only have access to classic and electric bicycles. A percentage of casual riders select docked bikes, and to obtain more annual members it's essential to provide them the option of having access to docked bikes through an annual membership. We can see this through these graphs:



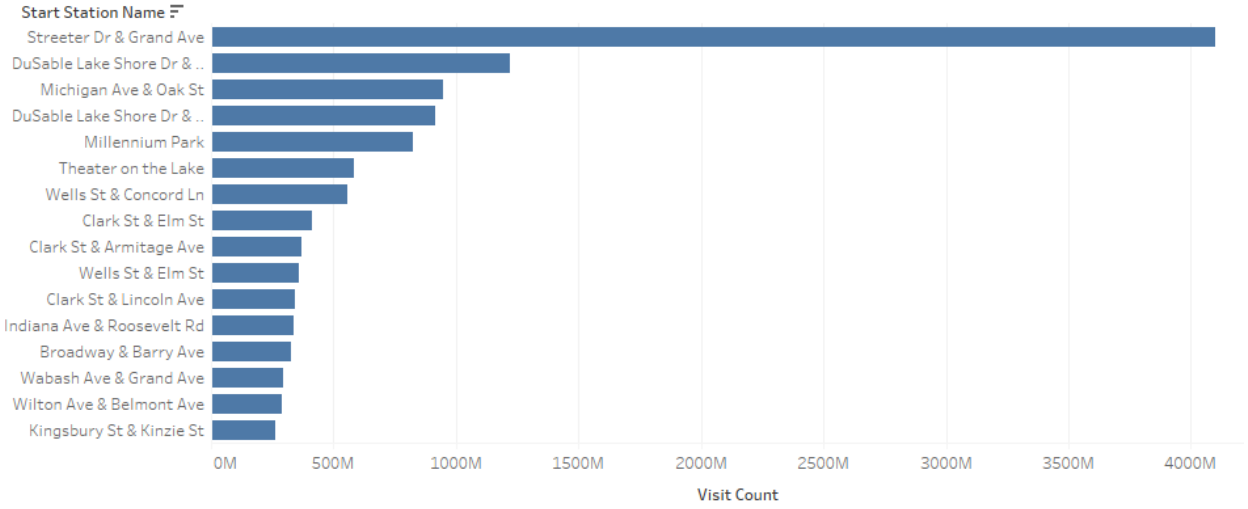
There is more variability in the rides made by casual riders

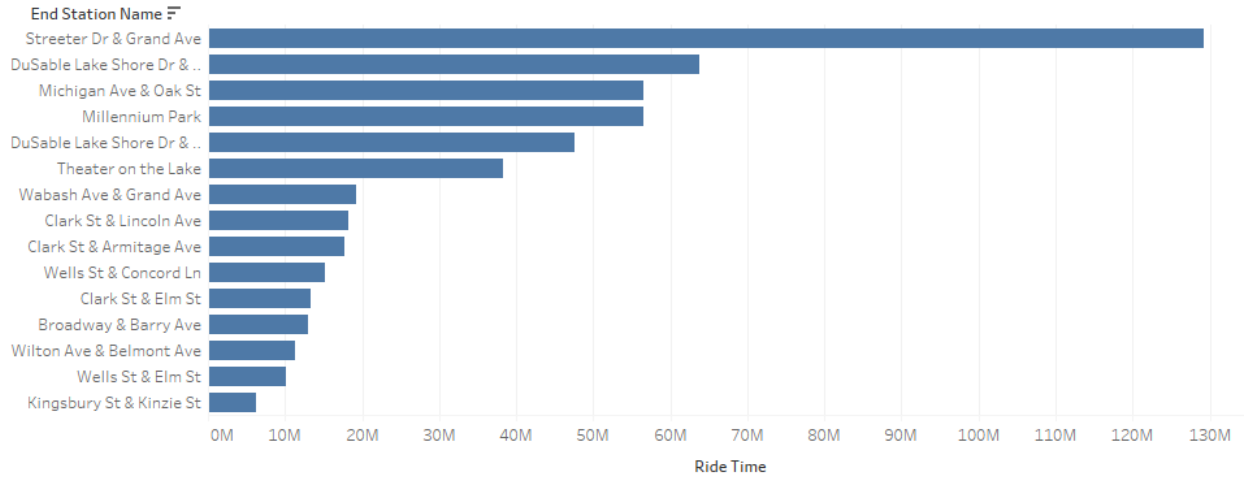
Casual riders tend to have a wider range of riding times, meaning they may take shorter or longer trips. Ultimately, this offers more flexibility in the types of rides they can undertake. By understanding their riding habits and preferences, Cyclistic can create campaigns that better meet the needs of casual riders and draw them towards becoming annual members. Furthermore, offering incentives based on different ride lengths may be an effective way to drive more conversions.



Most common start and end stations

The following charts illustrate the most common start and end stations for Cyclistic's rides. It's convenient to note that they tend to be the same, indicating that riders are likely starting and ending their trips within the same locations. This is beneficial for Cyclistic's marketing campaigns, as it allows them to target a single group of users rather than two separate groups.





The Act phase: Top three recommendations based on the analysis

1) **Offer members the choice to use docked bikes**. Memberships should include the choice to select each of the 3 different types of bikes, this way casual riders that have preference for docked bikes would have more appeal to become full-year members.

2) **Implement a publicity campaign at the most popular stations.** This can involve the placement of marketing materials at the most frequented stations, such as posters, flyers, and other interactive displays that encourage potential customers to learn more about the services offered. Additionally, we can organize promotional events at the most heavily used stations in order to drive further interest in the service.

3) **Implement strategies to target those who ride for pleasure or leisure.** This could include offering discounts for longer rides and creating special promotions for those who use the bikes for recreational activities such as sightseeing, picnics, or other leisurely activities. Additionally, the company could position itself as a fun and affordable way to explore the city and its attractions, which could attract more casual riders.

Stakeholders of a rideshare company can use this data to craft marketing campaigns that target casual riders by emphasizing the convenience and flexibility offered by an annual membership. For example, they could highlight the option of having access to both classic and electric bikes with the membership, as well as the ability to take shorter or longer rides depending on individual needs. Additionally, they could provide incentives such as discounts, free trials, or other special offers to further entice potential customers.

Conclusion

This project has demonstrated the importance of documentation during the data cleaning and transformation process. Through this project, we were able to gain valuable insights into data analysis principles and practices. Ultimately, this project served as an effective learning experience for how to handle and analyze data effectively. This project also highlighted the importance of following a structured process when dealing with data. We used the six-step approach - Ask, Prepare, Process, Analyze, Share and Act - to effectively analyze and draw meaningful conclusions from the data. Through this structured approach, we were able to gain a deeper understanding of how to effectively handle and manage data in a way that provided valuable and actionable insights.