Final Report

This Final Data Analysis report addresses the three fundamental questions asked by leadership. The report will give the top three recommendations and these key deliverables:

1. Statement of the business task
2. Description of all data sources used
3. Documentation of cleaning and manipulation of data
4. Summary of your analysis
5. Supporting visualizations and key findings
6. Top three recommendations based on your analysis

This report is addressed to:

- Cyclistic executive team
- Lily Moreno: Director of Marketing
- Cyclistic marketing analytics team

The company will use the information from this report and its recommendations to decide if they will pursue this initiative and proceed to an implementation plan, defining resources, time and funding.

Deliverable 01. Statement of the business task

Conduct Data Analysis with the information provided by the company to understand the differences between casual riders and member riders and determine the feasibility and strategy to convert casual riders into members.
Answer these fundamental questions:

1. How do annual members and casual riders use Cyclistic bikes differently?
On average Casual Riders Trips last 51.83% longer
On weekends, casual rides are on average 20% less than members but the difference grows on weekdays, where the difference oscillates between 53% and 66%

2. Why would casual riders buy Cyclistic annual memberships?
For the opportunity to have additional trips during the week for a healthier life style and money savings from a year membership

3. How can Cyclistic use digital media to influence casual riders to become members?
Targeted marketing campaign to casual riders, advertising the savings in cost from becoming a member and the health benefits of using a bicycle to commute to work, or running other frequent errands or events in their schedule

Deliverable 2. Description of all data sources used

The original data is located in Amazon AWS and is available for download, files containing records for the past 12 months From April 2020 to September 2024 were downloaded. The data is organized in files by month of the year:

Format of the files: YYYYMM-divvy-tripdata.zip

From 2013 to March 2020 – the data is organized in files by quarter of the year
Format of the files: Divvy_Trips_YYYY_QXQX.zip or Divvy_Trips_YYYY_QX.zip

Bias and Credibility:
The data complies with the recommended rules ROCCC:

- Reliable: files passed integrity check and content check
- Original: original source of data provided
- Comprehensive: twelve months of previous rides which is sufficient for the scope of the analysis
- Current: 12 previous months
- Cited: data has been made available by Motivate International Inc. under license

Potential Sources of Bias: Data has been collected electronically there is no human bias involved in data collection

Data Inconsistencies: Out of 5,854,527 records, a total of 24 outlier records with values that drastically affected the standard deviation were found in latitude and longitude values for initial and end trip coordinates. Analysis of the outliers is out of the scope of this study. The outlier values will not be included in the Analysis and they represent less that 0.0003%

Statistical analysis a statistical analysis was made on each of the files to obtain Min, Max, Mean, Median, Mode and Standard Deviation of the initial and final coordinates of the trip. Once the few outlier values were removed, the analysis revealed that the values of the mean and the median are very similar, revealing a symmetrical distribution. The standard deviation value for the trip coordinates is les than 0.5, which means that the data is clustered tightly around the mean.

Sample Size and Representation: The data being analyzed is the total rides for the last 12 month, this adds confidence in the analysis and the findings. The only values excluded were the outliers.

Cross-Reference with Other Sources: The maximum and minimum values for the coordinates in the data set were compared against coordinates from google maps, to confirm geographical accuracy of the data.

Relevance of the data: The analysis will consider the last 12 months of rides, which makes the information recent and relevant for analysis of trends

Licensing, privacy, security, and accessibility: The data has been made available by Motivate International Inc. under license the data in the files does not include personally identifiable information.

Data Integrity Verification: The data was checked for integrity by reviewing each of the 12 csv files to: check for Structural Issues, Validate Data Types and Ranges and Count of Rows and Columns. An integrity Check summary table is added below

Importance of Data for the objective of the analysis: conducting analysis on trip information will reveal the volume of trips per type of users

Issues with the data: Plotting of minimum and maximum values for the coordinates of start location and end location in google maps revealed an area wider than the city of Chicago.

Data Validation: Value Range Check

File Structure Check

No	File	Delimiters	line breaks	missing Start Station ID	% Missing	Missing End Station ID	% Missing	data types	range check	Total Rows	Total Columns
1	202310-divvy-tripdata.csv	PASS	PASS	84,412	15.72	89,253	16.62	PASS	PASS	537,113	13
2	202311-divvy-tripdata.csv	PASS	PASS	55,003	15.17	57,899	15.97	PASS	PASS	362,518	13
3	202312-divvy-tripdata.csv	PASS	PASS	35,710	15.94	37,924	16.92	PASS	PASS	224,073	13
4	202401-divvy-tripdata.csv	PASS	PASS	19,165	13.23	20,749	14.32	PASS	PASS	144,873	13
5	202402-divvy-tripdata.csv	PASS	PASS	23,418	10.49	25,151	11.27	PASS	PASS	223,164	13
6	202403-divvy-tripdata.csv	PASS	PASS	44,255	14.67	47,140	15.63	PASS	PASS	301,687	13
7	202404-divvy-tripdata.csv	PASS	PASS	74,793	18.02	78,519	18.92	PASS	PASS	415,025	13
8	202405-divvy-tripdata.csv	PASS	PASS	109,048	17.89	112,731	18.50	PASS	PASS	609,493	13
9	202406-divvy-tripdata.csv	PASS	PASS	144,025	20.26	148,027	20.83	PASS	PASS	710,721	13
10	202407-divvy-tripdata.csv	PASS	PASS	136,965	18.29	139,224	18.59	PASS	PASS	748,961	13
11	202408-divvy-tripdata.csv	PASS	PASS	140,591	18.61	142,225	18.82	PASS	PASS	755,621	13
12	202409-divvy-tripdata.csv	PASS	PASS	189,150	23.03	192,950	23.49	PASS	PASS	821,271	13
									Total Records	5,854,520	

Excel Table

item	Criteria
delimeters	visual inspection of columns
Line breaks	visual inspection of columns
Missing values	applied filter to detect missing values
data Types	conditional formatting
Range check	sort fileter
Total rows	checked from end of table
Total Columns	Always need to be 13

removed 1 record with outlier data
removed 18 records with outlier data
removed 5 record with outlier data

Data Validation: Statistical Analysis

No	File	Min start_lat	Max start_lat	Min start_long	Max start_long	Min end_lat	Max end_lat	Min end_long	Max end_long
1	202310-divvy-tripdata.csv	41.64	42.07	-87.86	-87.52	41.64	42.07	-87.99	-87.52
2	202311-divvy-tripdata.csv	41.648556	42.07	-87.94	-87.52823174	41.64850076	42.07	-87.94	-87.52823174
3	202312-divvy-tripdata.csv	41.69	42.064313	-87.83670282	-87.54658233	41.69	42.064313	-87.84	-87.5485146
4	202401-divvy-tripdata.csv	41.6485881	42.07	-87.84408414	-87.52843106	41.63	42.07	-87.86	-87.46
5	202402-divvy-tripdata.csv	41.6485008	42.07	-87.84410954	-87.52823174	41.64850076	42.08	-87.87	-87.53
6	202403-divvy-tripdata.csv	41.6485008	42.07	-87.84111	-87.52823174	41.63	42.08	-87.87	-87.52823174
7	202404-divvy-tripdata.csv	41.6485008	42.07	-87.89	-87.52823174	41.64850076	42.15	-87.96	-87.52823174
8	202405-divvy-tripdata.csv	41.6485008	42.07	-87.86	-87.52823174	41.63	42.17	-87.89	-87.49
9	202406-divvy-tripdata.csv	41.6485008	42.07	-87.89	-87.52823174	41.6	42.19	-88.12	-87.46
10	202407-divvy-tripdata.csv	41.64	42.07	-87.86	-87.52	41.48	42.11	-87.99	-87.5
11	202408-divvy-tripdata.csv	41.6485008	42.07	-87.9	-87.52	41.19	42.9	-89.27	-85.52
12	202409-divvy-tripdata.csv	41.64	42.07	-87.91	-87.52	41.64	42.73	-89.79	-85.89
	Min	**41.64**	**42.064313**	**-87.94**	**-87.54658233**	**41.19**	**42.064313**	**-89.79**	**-87.5485146**
	Max	**41.69**	**42.07**	**-87.83670282**	**-87.52**	**41.69**	**42.9**	**-87.84**	**-85.52**
	Mean	**41.6498457**	**42.06952608**	**-87.87300054**	**-87.52703365**	**41.58962519**	**42.223693**	**-88.19916667**	**-87.20860082**
	Median	**41.6485008**	**42.07**	**-87.86**	**-87.52823174**	**41.635**	**42.095**	**-87.95**	**-87.51**
	Mode	**41.6485008**	**42.07**	**-87.86**	**-87.52823174**	**41.64850076**	**42.07**	**-87.99**	**-87.52823174**
	Standard Deviation	**0.0132022**	**0.001641695**	**0.032581705**	**0.00732744**	**0.135601466**	**0.2818244**	**0.636174052**	**0.707292081**

Check for Structural Issues: Open the CSV file in a text editor or spreadsheet software to look for formatting errors like inconsistent delimiters, incorrect line breaks, or missing values.

Validate Data Types and Ranges: Ensure that the data in each column matches the expected data type (e.g., numbers, dates, text) and falls within a reasonable range.

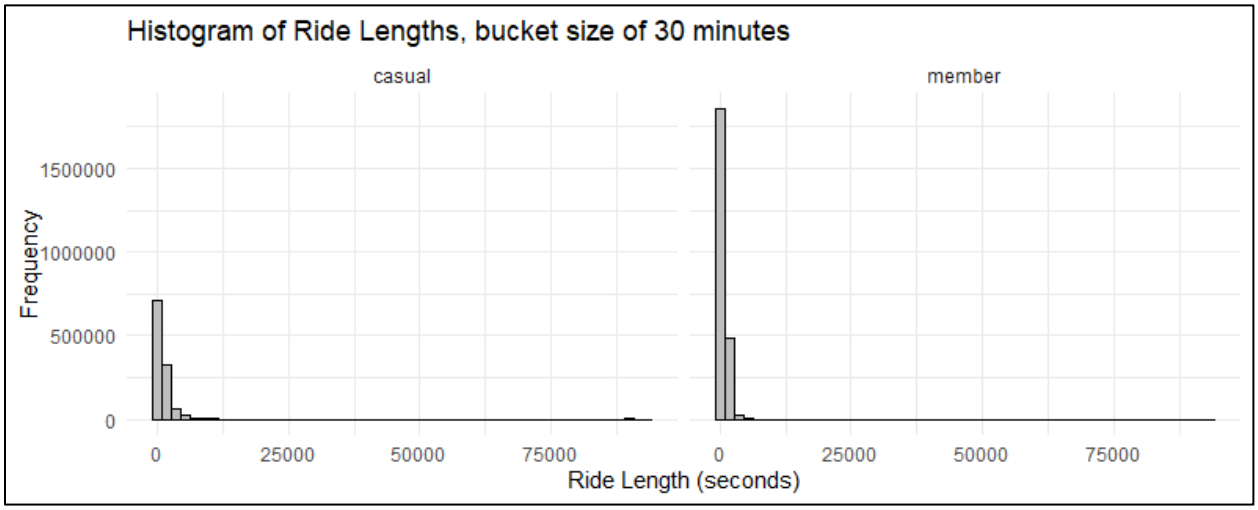
Count Rows and Columns: Verify that the total number of rows and columns aligns with your expectations based on the data source.

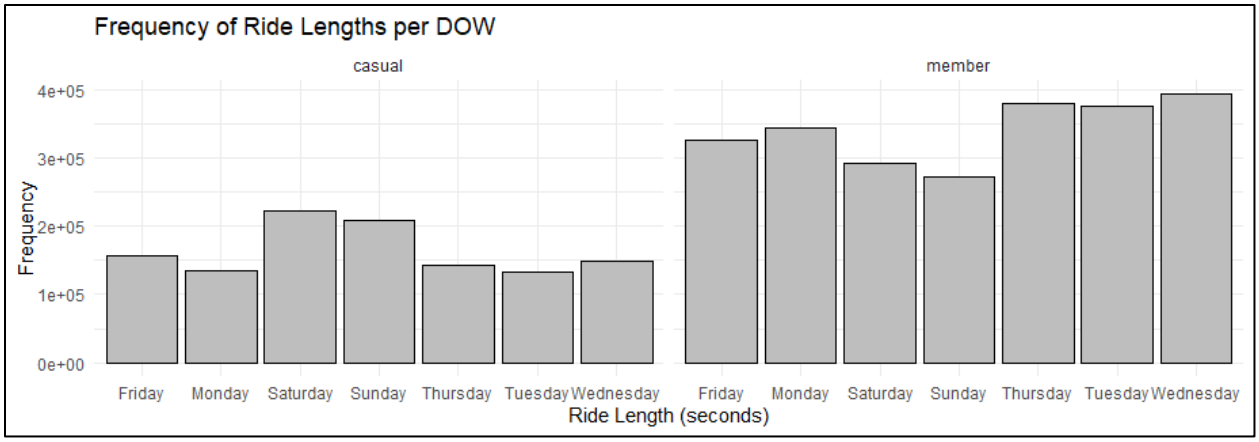
Deliverable 03 Data Processing, Documentation of cleaning and manipulation of data

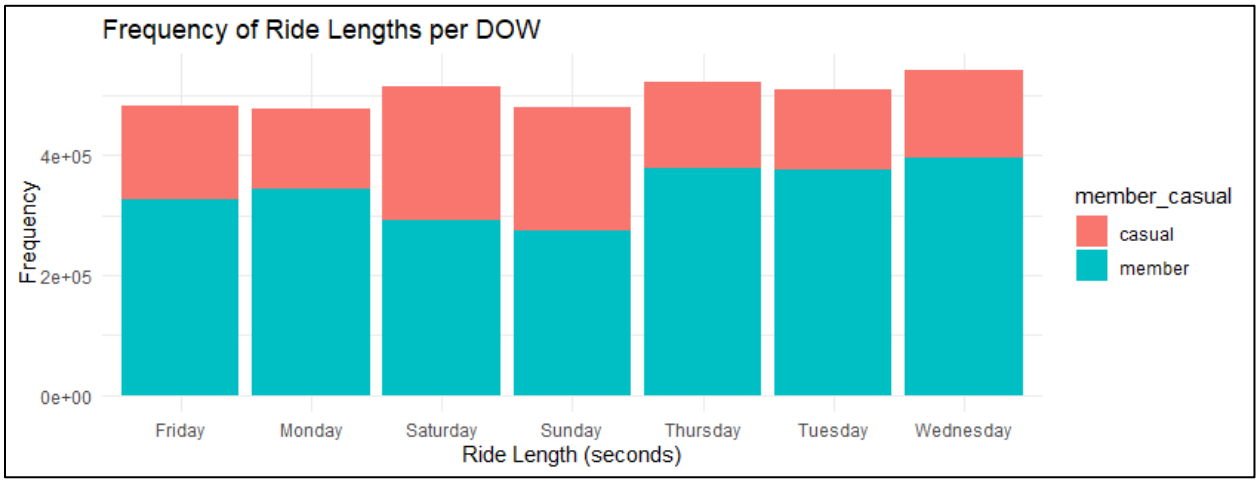
Data Cleaning: Even with validation, errors can occur. Data was Thoroughly clean before analysis. This included handling missing values, removing duplicates, and addressing outliers.
Addressing Outliers: Out of 5,854,527 records, a total of 24 outlier records with values that drastically affected the standard deviation were found in latitude and longitude values for initial and end trip coordinates. Analysis of the outliers is out of the scope of this study. The outlier values will not be included in the Analysis and they represent less that 0.0003%

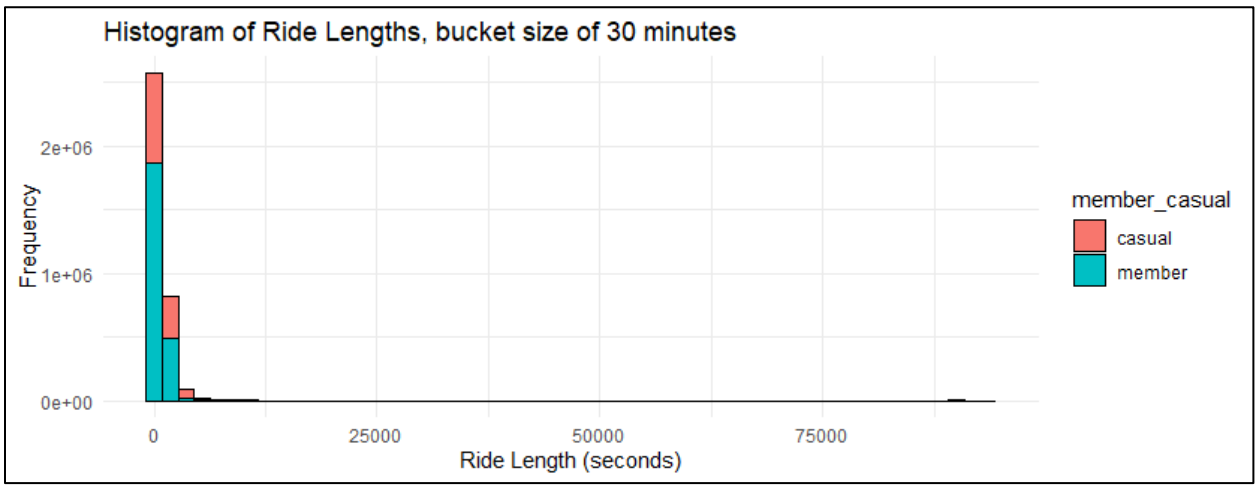
Removing Duplicates: There were no duplicate records found in the dataset
Missing Values: There were missing values for ride start and end date, those values were discarded since it was not possible to obtain ride length or day of the ride. The size of the dataset went from 5,854,520 to 3,528,667, 60.27% of the total records. The size of the sample is 60% and the error appeared randomly through out the files. Sampling was random and unbiased.

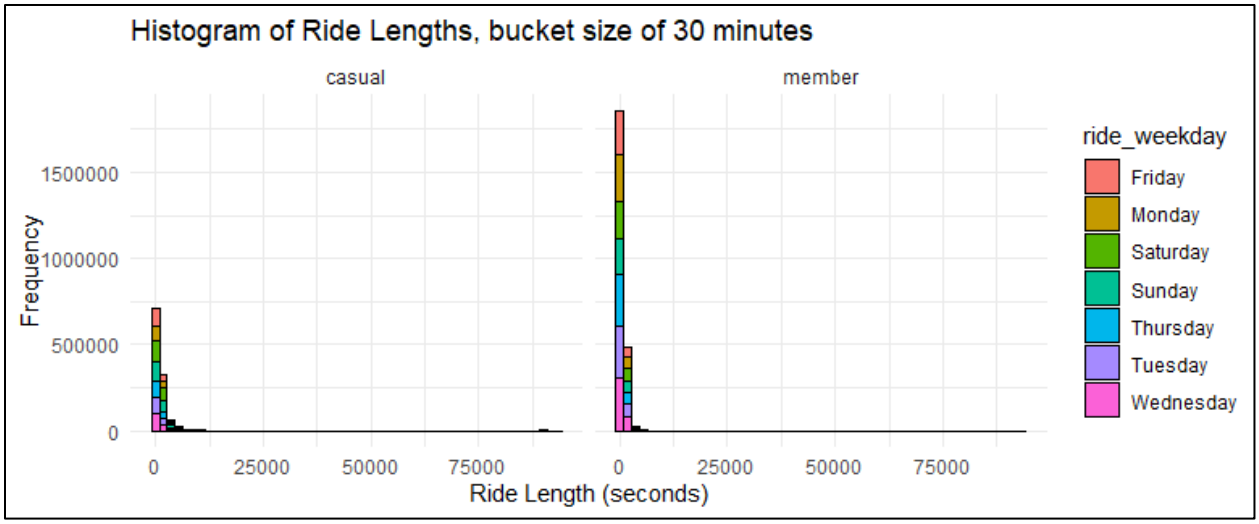
Data Transformation: Transforming data in a consistent and documented manner is key. This might involve standardizing formats, converting units, or creating new variables. Be sure to document these transformations for clarity and reproducibility.
All the transformations were documented and saved as R scripts and are part of the deliverables
Data Processing - Visuals generated from R code:




Frequency of Ride Lengths per DOW
casual
member
Frequency
4e+05
3e+05
2e+05
1e+05
0e+00
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday
Ride Length (seconds)


Frequency of Ride Lengths per DOW
Frequency
4e+05
2e+05
0e+00
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday
Ride Length (seconds)
member_casual
casual
member


Histogram of Ride Lengths, bucket size of 30 minutes
Frequency
2e+06
1e+06
0e+00
0
25000
50000
75000
Ride Length (seconds)
member_casual
casual
member


Histogram of Ride Lengths, bucket size of 30 minutes
casual
member
Frequency
1500000
1000000
500000
0
0
25000
50000
75000
Ride Length (seconds)
ride_weekday
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday

Deliverable 04A Analysis Summary

Below is summary of steps and values obtained, the complete scripts and results are included in appendix 1 – Data Analysis with R **Delivery 04 B Detailed Analysis**

1) Combining the Data into a single file

Total Rows	Total Columns
3,528,667	20

2) Determine total of observations by casual and member

Rides/Percentage	Total Observations Casual	Total Observations Member	Total
Rides	1,145,263	2,383,404	3,528,667.00
Percentage	**32%**	**68%**	**100%**

3) Add Columns that list the date, month, day and year of each ride

4) Add a "ride_length" calculation to all trips (in seconds)

5) Removing Bad Data and redundant columns

6) Determine total of observations by casual and member

Rides/Percentage	Total Observations Casual	Total Observations Member	Total
Rides	820,689	1,765,365	2,586,054.00
Percentage	**32%**	**68%**	**100%**

7) Data Calculations

Calculation	Value In Seconds	Value In Minutes	Value In Hours
Mean value for ride length	952.88	15.88	0.26
Maximum ride length	90,600.00	1,510.00	25.17
Mode of Day_of_Week	WEDNESDAYS	WEDNESDAYS	WEDNESDAYS

Mean ride length for members and casuals

Member / Casual	Mean Value In Seconds	Mean Value In Minutes
Casual	1420.00	23.67
Member	736.00	12.27

Mean Ride Length for Casual Riders is 51.83% longer than Casual Ride Length
Casual Riders Trips last 51.83% longer

Calculate the average ride_length for members and casuals by day_of_week in minutes

Day	Members AVG Ride Length	Casuals avg Ride Length	% Casual is longer than member
Friday	11.8	22.4	89.83
Monday	11.8	22.9	94.07
Saturday	13.7	26.9	96.35
Sunday	13.9	27.9	100.72
Thursday	11.6	19.8	70.69
Tuesday	11.8	20.3	72.03
Wednesday	12.0	21.2	76.67
Average Ride Length for Casuals is at least 70% longer than Members on any given day, Sunday is the highest value of 100.72%			

Calculate the number of rides for users by day_of_week

Day	Member Total Rides	Casual Total Rides	% Casual perform less trips than members
Friday	237,995	110,606	53.53
Monday	261,675	98,368	62.41
Saturday	207,807	161,568	22.25
Sunday	198,359	152,451	23.14
Thursday	282,269	99,216	64.85
Tuesday	283,112	94,566	66.60
Wednesday	294,148	103,914	64.67
Total Rides	**1,765,365**	**820,689**	**53.51**

On the weekend, the casual rides are 20% less than members but the difference grows on weekdays, where the difference oscillates between 53% and 66%

Mean trip length value in minutes	15.88136 min.
Median trip length value in minutes	10 min.
Max trip length value in hours	25.16667 hrs.
Min trip length value in seconds	60 secs.

```
summary(merged_data$ride_length_sec) RESULT IN SECONDS
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    60.0   360.0   600.0   952.9  1020.0 90600.0

summary(merged_data$ride_length_sec/60) RESULT IN MINUTES
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    6.00   10.00   15.88   17.00 1510.00

summary(merged_data$ride_length_sec/3600) RESULT IN HOURS
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.01667  0.10000  0.16667  0.26469  0.28333 25.16667
```

Compare members and casual members

```
aggregate(merged_data$ride_length ~ merged_data$member_casual, FUN = mean)
```

MEAN VALUE FOR RIDE LENGTH	
Rider Type	**Mean Value Ride Length Seconds**
Casual	1,420.15
Member	735.66
MEAN Value of Ride Length for Members is 48.19% shorter than Casual riders	

```
aggregate(merged_data$ride_length ~ merged_data$member_casual, FUN = median)
```

MEDIAN VALUE FOR RIDE LENGTH	
Rider Type	**Median Value Ride Length Seconds**
Casual	780.00
Member	540.00
MEDIAN Value of Ride Length for Members is 30.76% shorter than Casual riders	

```
aggregate(merged_data$ride_length ~ merged_data$member_casual, FUN = max)
```

MAXIMUM VALUE FOR RIDE LENGTH	
Rider Type	**Maximum Value Ride Length Seconds**
Casual	90,600.00
Member	89,880.00
MAXIMUM Value of Ride Length for Members and Casual Riders is virtually the same	

```
aggregate(merged_data$ride_length ~ merged_data$member_casual, FUN = min)
```

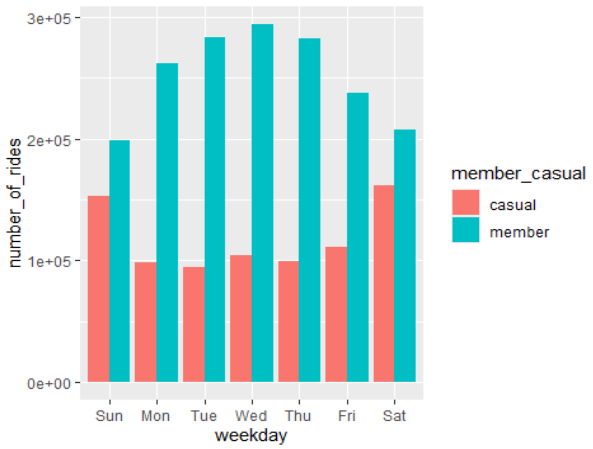
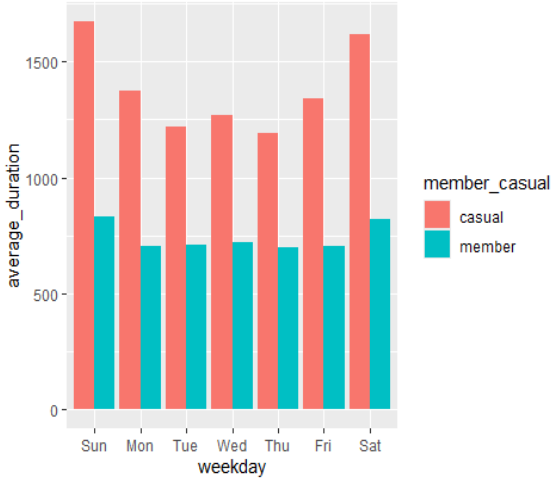
MINIMUM VALUE FOR RIDE LENGTH	
Rider Type	**Minimum Value Ride Length Seconds**
Casual	60.00
Member	60.00
MINIMUM Value of Ride Length for Members and Casual Riders is the same	

Order days of the week

```
merged_data$day_of_week <- ordered(merged_data$day_of_week, levels=c("Sunday"
, "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))
```

average ride time and ride length by each day for members vs casual users

Day	Casual Total Rides	Member Total Rides	Average Ride Time Secs Casual	Average Ride Time Secs Members
Sunday	152,451	198,359	1674.3165	832.022
Monday	98,368	261,675	1376.7453	705.487
Tuesday	94,566	283,112	1219.3249	710.7073
Wednesday	103,914	294,148	1270.0814	720.5954
Thursday	99,216	282,269	1189.1467	698.3024
Friday	110,606	237,995	1341.8147	705.7202
Saturday	161,568	207,807	1616.2734	822.0136
Total/Average	**820,689**	**1,765,365**	**1383.957557**	**742.1211286**

Number of Rides by Rider Type	Average Duration
	

Delivery 04 B Detailed Analysis

Statistical Analysis of Information

Alfonso

2024-11-13

R Markdown

First we call the necessary libraries

```
 library(tidyverse)  #helps wrangle data
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────
tidyverse_conflicts() ──
## ✗ dplyr::filter() masks stats::filter()
## ✗ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
library(conflicted) # Use the conflicted package to manage conflicts
library(dplyr)
library(readr)
library(lubridate)
library(ggplot2)
```

now setting the default choices for conflict

```
# Set dplyr::filter and dplyr::lag as the default choices
conflict_prefer("filter", "dplyr")
## [conflicted] Will prefer dplyr::filter over any other package.
conflict_prefer("lag", "dplyr")
## [conflicted] Will prefer dplyr::lag over any other package.
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

#==================== # STEP 1: COLLECT DATA #====================

```
#Open file
setwd("C:/clean")
```

#========== # STEP 2: WRANGLE DATA AND COMBINE INTO A SINGLE FILE #==========

```
# Load the CSV file into a data frame
merged_data <-
read_csv("merged_data_trip_length_day_of_week_seconds_numeric.csv")
## New names:
## Rows: 3528667 Columns: 20
## ── Column specification
## ──────────────────────────────────────────────────────── Delimiter: ","
chr
## (8): ride_id, rideable_type, start_station_name, start_station_id, end_...
dbl
## (9): ...1, ...2, ...3, ...4, start_lat, start_lng, end_lat, end_lng, ri...
dttm
## (2): started_at, ended_at time (1): ride_length
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ
## Specify the column types or set `show_col_types = FALSE` to quiet this
message.
## • `` -> `...1`
## • `...1` -> `...2`
## • `...2` -> `...3`
## • `...3` -> `...4`
#View(merged_data)

#merged_data <- na.omit(merged_data)
```

#========== # STEP 3: CLEAN UP AND ADD DATA TO PREPARE FOR ANALYSIS

#========== # Inspect the new table that has been created

Obtaining the column names:

```
colnames(merged_data)  #List of column names
##  [1] "...1"               "...2"               "...3"
##  [4] "...4"               "ride_id"            "rideable_type"
##  [7] "started_at"         "ended_at"           "start_station_name"
## [10] "start_station_id"   "end_station_name"   "end_station_id"
## [13] "start_lat"          "start_lng"          "end_lat"
## [16] "end_lng"            "member_casual"      "ride_length"
## [19] "ride_weekday"       "ride_length_sec"
```

How many rows are there in the data frame?

```
nrow(merged_data)  #How many rows are in data frame?
## [1] 3528667
```

Dimensions of the Data Frame

```
dim(merged_data)
## [1] 3528667      20
```

first and last rows of the Data Frame

```
head(merged_data)  #See the first 6 rows of data frame.
## # A tibble: 6 × 20
##    ...1  ...2  ...3  ...4 ride_id          rideable_type started_at
##   <dbl> <dbl> <dbl> <dbl> <chr>            <chr>         <dttm>
## 1     1     1     1     1 B541441FAF64B31C classic_bike  2023-10-01
00:00:00
```

```
## 2     2     2     2     2 163D8093FDDEBBC8 classic_bike  2023-10-01
00:00:00
## 3     3     3     3     3 46803FC8C419FBB0 classic_bike  2023-10-01
00:00:00
## 4     4     4     4     4 0ED9A33B7B80912D classic_bike  2023-10-01
00:00:00
## 5     5     5     5     5 0FBFC8BCF712A9B1 classic_bike  2023-10-01
00:00:00
## 6     6     6     6     6 5F08B2CEF05DAF9F classic_bike  2023-10-01
00:00:00
## # ℹ 13 more variables: ended_at <dttm>, start_station_name <chr>,
## #   start_station_id <chr>, end_station_name <chr>, end_station_id <chr>,
## #   start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, end_lng <dbl>,
## #   member_casual <chr>, ride_length <time>, ride_weekday <chr>,
## #   ride_length_sec <dbl>
tail(merged_data)  #See the last 6 rows of data frame.
## # A tibble: 6 × 20
##      ...1    ...2    ...3    ...4 ride_id      rideable_type started_at
##     <dbl>   <dbl>   <dbl>   <dbl> <chr>        <chr>         <dttm>
## 1 3528662 3528662 3528662 3528662 E3BEED04143… electric_bike 2024-06-24
17:12:00
## 2 3528663 3528663 3528663 3528663 1D1EBE57758… electric_bike 2024-06-11
08:25:00
## 3 3528664 3528664 3528664 3528664 2F63E9CD01D… electric_bike 2024-06-24
11:40:00
## 4 3528665 3528665 3528665 3528665 97D225818F9… electric_bike 2024-06-30
10:43:00
## 5 3528666 3528666 3528666 3528666 C8D2A48B901… electric_bike 2024-06-11
18:20:00
## 6 3528667 3528667 3528667 3528667 C372E7A1A7B… electric_bike 2024-06-15
15:48:00
## # ℹ 13 more variables: ended_at <dttm>, start_station_name <chr>,
## #   start_station_id <chr>, end_station_name <chr>, end_station_id <chr>,
## #   start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, end_lng <dbl>,
## #   member_casual <chr>, ride_length <time>, ride_weekday <chr>,
## #   ride_length_sec <dbl>
```

List of columns and data Types

```
str(merged_data)  #See list of columns and data types (numeric, character,
etc)
## spc_tbl_ [3,528,667 × 20] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ...2              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ...3              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ...4              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ride_id           : chr [1:3528667] "B541441FAF64B31C"
"163D8093FDDEBBC8" "46803FC8C419FBB0" "0ED9A33B7B80912D" ...
##  $ rideable_type     : chr [1:3528667] "classic_bike" "classic_bike"
"classic_bike" "classic_bike" ...
##  $ started_at        : POSIXct[1:3528667], format: "2023-10-01 00:00:00"
"2023-10-01 00:00:00" ...
```

```
##  $ ended_at          : POSIXct[1:3528667], format: "2023-10-01 00:32:00"
"2023-10-01 00:17:00" ...
##  $ start_station_name: chr [1:3528667] "Mies van der Rohe Way & Chicago
Ave" "Wells St & Elm St" "Racine Ave & 18th St" "Broadway & Waveland Ave" ...
##  $ start_station_id  : chr [1:3528667] "13338" "KA1504000135" "13164"
"13325" ...
##  $ end_station_name  : chr [1:3528667] "Michigan Ave & Oak St" "Southport
Ave & Wrightwood Ave" "Racine Ave & 18th St" "Clarendon Ave & Gordon Ter" ...
##  $ end_station_id    : chr [1:3528667] "13042" "TA1307000113" "13164"
"13379" ...
##  $ start_lat         : num [1:3528667] 41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num [1:3528667] -87.6 -87.6 -87.7 -87.6 -87.7 ...
##  $ end_lat           : num [1:3528667] 41.9 41.9 41.9 42 41.9 ...
##  $ end_lng           : num [1:3528667] -87.6 -87.7 -87.7 -87.6 -87.7 ...
##  $ member_casual     : chr [1:3528667] "casual" "casual" "casual" "member"
...
##  $ ride_length       : 'hms' num [1:3528667] 00:32:00 00:17:00 00:02:00
00:05:00 ...
##   ..- attr(*, "units")= chr "secs"
##  $ ride_weekday      : chr [1:3528667] "Sunday" "Sunday" "Sunday" "Sunday"
...
##  $ ride_length_sec   : num [1:3528667] 1920 1020 120 300 2340 240 420 1260
420 540 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_double(),
##   ..   ...2 = col_double(),
##   ..   ...3 = col_double(),
##   ..   ...4 = col_double(),
##   ..   ride_id = col_character(),
##   ..   rideable_type = col_character(),
##   ..   started_at = col_datetime(format = ""),
##   ..   ended_at = col_datetime(format = ""),
##   ..   start_station_name = col_character(),
##   ..   start_station_id = col_character(),
##   ..   end_station_name = col_character(),
##   ..   end_station_id = col_character(),
##   ..   start_lat = col_double(),
##   ..   start_lng = col_double(),
##   ..   end_lat = col_double(),
##   ..   end_lng = col_double(),
##   ..   member_casual = col_character(),
##   ..   ride_length = col_time(format = ""),
##   ..   ride_weekday = col_character(),
##   ..   ride_length_sec = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Statistical summary of Data

```
str(merged_data)  #See list of columns and data types (numeric, character,
etc)
```

```
## spc_tbl_ [3,528,667 × 20] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ...2              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ...3              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ...4              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ride_id           : chr [1:3528667] "B541441FAF64B31C"
"163D8093FDDEBBC8" "46803FC8C419FBB0" "0ED9A33B7B80912D" ...
##  $ rideable_type     : chr [1:3528667] "classic_bike" "classic_bike"
"classic_bike" "classic_bike" ...
##  $ started_at        : POSIXct[1:3528667], format: "2023-10-01 00:00:00"
"2023-10-01 00:00:00" ...
##  $ ended_at          : POSIXct[1:3528667], format: "2023-10-01 00:32:00"
"2023-10-01 00:17:00" ...
##  $ start_station_name: chr [1:3528667] "Mies van der Rohe Way & Chicago
Ave" "Wells St & Elm St" "Racine Ave & 18th St" "Broadway & Waveland Ave" ...
##  $ start_station_id  : chr [1:3528667] "13338" "KA1504000135" "13164"
"13325" ...
##  $ end_station_name  : chr [1:3528667] "Michigan Ave & Oak St" "Southport
Ave & Wrightwood Ave" "Racine Ave & 18th St" "Clarendon Ave & Gordon Ter" ...
##  $ end_station_id    : chr [1:3528667] "13042" "TA1307000113" "13164"
"13379" ...
##  $ start_lat         : num [1:3528667] 41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num [1:3528667] -87.6 -87.6 -87.7 -87.6 -87.7 ...
##  $ end_lat           : num [1:3528667] 41.9 41.9 41.9 42 41.9 ...
##  $ end_lng           : num [1:3528667] -87.6 -87.7 -87.7 -87.6 -87.7 ...
##  $ member_casual     : chr [1:3528667] "casual" "casual" "casual" "member"
...
##  $ ride_length       : 'hms' num [1:3528667] 00:32:00 00:17:00 00:02:00
00:05:00 ...
##   ..- attr(*, "units")= chr "secs"
##  $ ride_weekday      : chr [1:3528667] "Sunday" "Sunday" "Sunday" "Sunday"
...
##  $ ride_length_sec   : num [1:3528667] 1920 1020 120 300 2340 240 420 1260
420 540 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_double(),
##   ..   ...2 = col_double(),
##   ..   ...3 = col_double(),
##   ..   ...4 = col_double(),
##   ..   ride_id = col_character(),
##   ..   rideable_type = col_character(),
##   ..   started_at = col_datetime(format = ""),
##   ..   ended_at = col_datetime(format = ""),
##   ..   start_station_name = col_character(),
##   ..   start_station_id = col_character(),
##   ..   end_station_name = col_character(),
##   ..   end_station_id = col_character(),
##   ..   start_lat = col_double(),
##   ..   start_lng = col_double(),
```

```
##   ..   end_lat = col_double(),
##   ..   end_lng = col_double(),
##   ..   member_casual = col_character(),
##   ..   ride_length = col_time(format = ""),
##   ..   ride_weekday = col_character(),
##   ..   ride_length_sec = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
summary(merged_data)  #Statistical summary of data. Mainly for numerics
##       ...1              ...2              ...3              ...4
##  Min.   :      1   Min.   :      1   Min.   :      1   Min.   :      1
##  1st Qu.: 882168   1st Qu.: 882168   1st Qu.: 882168   1st Qu.: 882168
##  Median :1764334   Median :1764334   Median :1764334   Median :1764334
##  Mean   :1764334   Mean   :1764334   Mean   :1764334   Mean   :1764334
##  3rd Qu.:2646500   3rd Qu.:2646500   3rd Qu.:2646500   3rd Qu.:2646500
##  Max.   :3528667   Max.   :3528667   Max.   :3528667   Max.   :3528667
##
##    ride_id          rideable_type        started_at
##  Length:3528667     Length:3528667     Min.   :2023-10-01 00:00:00.00
##  Class :character   Class :character   1st Qu.:2023-11-29 16:27:00.00
##  Mode  :character   Mode  :character   Median :2024-03-28 19:14:00.00
##                                        Mean   :2024-03-03 03:24:30.35
##                                        3rd Qu.:2024-05-23 13:28:00.00
##                                        Max.   :2024-06-30 23:55:00.00
##
##     ended_at                      start_station_name start_station_id
##  Min.   :2023-10-01 00:02:00.00   Length:3528667     Length:3528667
##  1st Qu.:2023-11-29 16:37:30.00   Class :character   Class :character
##  Median :2024-03-28 19:28:00.00   Mode  :character   Mode  :character
##  Mean   :2024-03-03 03:41:15.49
##  3rd Qu.:2024-05-23 13:46:00.00
##  Max.   :2024-06-30 23:59:00.00
##
##  end_station_name   end_station_id       start_lat       start_lng
##  Length:3528667     Length:3528667     Min.   :41.64   Min.   :-87.94
##  Class :character   Class :character   1st Qu.:41.88   1st Qu.:-87.66
##  Mode  :character   Mode  :character   Median :41.90   Median :-87.64
##                                        Mean   :41.90   Mean   :-87.65
##                                        3rd Qu.:41.93   3rd Qu.:-87.63
##                                        Max.   :42.07   Max.   :-87.52
##
##     end_lat         end_lng       member_casual      ride_length
##  Min.   :41.60   Min.   :-88.12   Length:3528667     Length:3528667
##  1st Qu.:41.88   1st Qu.:-87.66   Class :character   Class1:hms
##  Median :41.90   Median :-87.64   Mode  :character   Class2:difftime
##  Mean   :41.90   Mean   :-87.65                      Mode  :numeric
##  3rd Qu.:41.93   3rd Qu.:-87.63
##  Max.   :42.19   Max.   :-87.46
##  NA's   :4570    NA's   :4570
##  ride_weekday       ride_length_sec
```

```
##  Length:3528667     Min.   :    0
##  Class :character   1st Qu.:  300
##  Mode  :character   Median :  540
##                     Mean   : 1007
##                     3rd Qu.:  960
##                     Max.   :93600
##                     NA's   :123
```

Begin by seeing how many observations fall under each usertype

```
table(merged_data$member_casual)
##
##  casual  member
## 1145263 2383404
```

Add columns that list the date, month, day, and year of each ride

This will allow us to aggregate ride data for each month, day, or year … before completing these operations we could only aggregate at the ride level

https://www.statmethods.net/input/dates.html more on date formats in R found at that link

```
merged_data$date <- as.Date(merged_data$started_at) #The default format is yyyy-mm-dd
merged_data$month <- format(as.Date(merged_data$date), "%m")
merged_data$day <- format(as.Date(merged_data$date), "%d")
merged_data$year <- format(as.Date(merged_data$date), "%Y")
merged_data$day_of_week <- format(as.Date(merged_data$date), "%A")
```

Add a "ride_length" calculation to all_trips (in seconds)

https://stat.ethz.ch/R-manual/R-devel/library/base/html/difftime.html

#there is another column called ride_length_seconds that also calculated this value

```
merged_data$ride_length <-
difftime(merged_data$ended_at,merged_data$started_at)
```

Inspect the structure of the columns

```
str(merged_data)
## spc_tbl_ [3,528,667 × 25] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ...2              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ...3              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ...4              : num [1:3528667] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ride_id           : chr [1:3528667] "B541441FAF64B31C"
"163D8093FDDEBBC8" "46803FC8C419FBB0" "0ED9A33B7B80912D" ...
##  $ rideable_type     : chr [1:3528667] "classic_bike" "classic_bike"
"classic_bike" "classic_bike" ...
##  $ started_at        : POSIXct[1:3528667], format: "2023-10-01 00:00:00"
"2023-10-01 00:00:00" ...
##  $ ended_at          : POSIXct[1:3528667], format: "2023-10-01 00:32:00"
"2023-10-01 00:17:00" ...
##  $ start_station_name: chr [1:3528667] "Mies van der Rohe Way & Chicago
```

```
Ave" "Wells St & Elm St" "Racine Ave & 18th St" "Broadway & Waveland Ave" ...
##  $ start_station_id  : chr [1:3528667] "13338" "KA1504000135" "13164"
"13325" ...
##  $ end_station_name  : chr [1:3528667] "Michigan Ave & Oak St" "Southport
Ave & Wrightwood Ave" "Racine Ave & 18th St" "Clarendon Ave & Gordon Ter" ...
##  $ end_station_id    : chr [1:3528667] "13042" "TA1307000113" "13164"
"13379" ...
##  $ start_lat         : num [1:3528667] 41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num [1:3528667] -87.6 -87.6 -87.7 -87.6 -87.7 ...
##  $ end_lat           : num [1:3528667] 41.9 41.9 41.9 42 41.9 ...
##  $ end_lng           : num [1:3528667] -87.6 -87.7 -87.7 -87.6 -87.7 ...
##  $ member_casual     : chr [1:3528667] "casual" "casual" "casual" "member"
...
##  $ ride_length       : 'difftime' num [1:3528667] 1920 1020 120 300 ...
##   ..- attr(*, "units")= chr "secs"
##  $ ride_weekday      : chr [1:3528667] "Sunday" "Sunday" "Sunday" "Sunday"
...
##  $ ride_length_sec   : num [1:3528667] 1920 1020 120 300 2340 240 420 1260
420 540 ...
##  $ date              : Date[1:3528667], format: "2023-10-01" "2023-10-01"
...
##  $ month             : chr [1:3528667] "10" "10" "10" "10" ...
##  $ day               : chr [1:3528667] "01" "01" "01" "01" ...
##  $ year              : chr [1:3528667] "2023" "2023" "2023" "2023" ...
##  $ day_of_week       : chr [1:3528667] "Sunday" "Sunday" "Sunday" "Sunday"
...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_double(),
##   ..   ...2 = col_double(),
##   ..   ...3 = col_double(),
##   ..   ...4 = col_double(),
##   ..   ride_id = col_character(),
##   ..   rideable_type = col_character(),
##   ..   started_at = col_datetime(format = ""),
##   ..   ended_at = col_datetime(format = ""),
##   ..   start_station_name = col_character(),
##   ..   start_station_id = col_character(),
##   ..   end_station_name = col_character(),
##   ..   end_station_id = col_character(),
##   ..   start_lat = col_double(),
##   ..   start_lng = col_double(),
##   ..   end_lat = col_double(),
##   ..   end_lng = col_double(),
##   ..   member_casual = col_character(),
##   ..   ride_length = col_time(format = ""),
##   ..   ride_weekday = col_character(),
##   ..   ride_length_sec = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# Remove "bad" data
# The dataframe includes a few hundred entries when bikes were taken out of
docks and checked for quality by Divvy or ride_length was negative
# https://www.datasciencemadesimple.com/delete-or-drop-rows-in-r-with-
conditions-2/


# Need to remove trips where ride length is >=0, the other field
ride_length_seconds is NA,
# Need to remove trips where ride length in seconds is NA



merged_data <- merged_data[!(merged_data$start_station_name == "HQ QR" |
merged_data$ride_length<0),]

merged_data <- merged_data[!(merged_data$start_station_name == "HQ QR" |
merged_data$ride_length==0),]



merged_data <- na.omit(merged_data) # this was too much, it deleted millions
or records, better only
#remove NA for the column of time
#continue tomorrow
```

removing columns 2,3,4

```
 merged_data <- merged_data[, -4]
 merged_data <- merged_data[, -3]
 merged_data <- merged_data[, -2]
```

#==========# STEP 4: CONDUCT DESCRIPTIVE ANALYSIS
#==========

Begin by seeing how many observations fall under each usertype

```
table(merged_data$member_casual)
##
##  casual  member
##  820689 1765365
```

Calculate the mean of ride_length

```
mean_value <- mean(merged_data$ride_length, na.rm = TRUE)
print(mean_value)
## Time difference of 952.8819 secs
mean_value <- mean(merged_data$ride_length_sec, na.rm = TRUE)
print(mean_value)
## [1] 952.8819
output <- paste0("The mean ride_length value in minutes is: ", mean_value/60)
print(output)
## [1] "The mean ride_length value in minutes is: 15.8813648129544"
```

Calculate the max ride_length

```
max_value <- max(merged_data$ride_length, na.rm = TRUE)
print(max_value)
## Time difference of 90600 secs
max_value <- max(merged_data$ride_length_sec, na.rm = TRUE)
print(max_value)
## [1] 90600
output <- paste0("The max ride_length value in hours is: ", max_value/3600)
print(output)
## [1] "The max ride_length value in hours is: 25.1666666666667"
```

Calculate the mode of day_of_week # Mode is the unique value with the highest frequency

```
# Custom function to calculate the mode
calculate_mode <- function(x) {
  unique_x <- unique(x)
  unique_x[which.max(tabulate(match(x, unique_x)))]
}

# Calculate the mode of the 'values' column
mode_value <- calculate_mode(merged_data$day_of_week)
# Print the mode value
print("mode value:")
## [1] "mode value:"
print(mode_value)
## [1] "Wednesday"
```

Calculate the average ride_length for members and casual riders

```
# Calculate the mean of ride_length for each member_casual group

mean_ride_length <- merged_data %>% group_by(member_casual) %>%
summarize(mean_ride_length = mean(ride_length_sec, na.rm = TRUE))

mean_ride_length_min <- merged_data %>% group_by(member_casual) %>%
summarize(mean_ride_length_min = mean(ride_length_sec, na.rm = TRUE)/60)

# Print the result
print("mean ride length for members and casuals")
## [1] "mean ride length for members and casuals"
print(mean_ride_length)
## # A tibble: 2 × 2
##   member_casual mean_ride_length
##   <chr>                    <dbl>
## 1 casual                   1420.
## 2 member                    736.
print("mean ride length for members and casuals in minutes")
## [1] "mean ride length for members and casuals in minutes"
print(mean_ride_length_min)
## # A tibble: 2 × 2
##   member_casual mean_ride_length_min
##   <chr>                        <dbl>
```

```
## 1 casual                          23.7
## 2 member                          12.3
```

Calculate the average ride_length for users by day_of_week

#Prompt R language: my data frame called merged_data has three columns: column 1 is called ride_length, the format is col_time(format = “”) Column 2 is called member_casual , the format is col_character() this column only has two possible values”users” and “casual” and the third column is called ride_weekday the format is col_character() and has the strings for the days of the week “Monday” , “Tuesday”….. I need to Calculate the average ride_length for member_casual = “users”users by day_of_week

```
# Calculate the average ride_length for member_casual = member by day_of_week

average_ride_length_M <- merged_data %>%
  filter(member_casual == "member") %>%
  group_by(ride_weekday) %>%
  summarise(avg_ride_length_M = mean(ride_length_sec / 60, na.rm = TRUE))

# Print the result
print ("average ride_length in minutes for member_casual = member by
day_of_week ")
## [1] "average ride_length in minutes for member_casual = member by
day_of_week "
print(average_ride_length_M)
## # A tibble: 7 × 2
##   ride_weekday avg_ride_length_M
##   <chr>                    <dbl>
## 1 Friday                    11.8
## 2 Monday                    11.8
## 3 Saturday                  13.7
## 4 Sunday                    13.9
## 5 Thursday                  11.6
## 6 Tuesday                   11.8
## 7 Wednesday                 12.0
```

Calculate the average ride_length for casuals by day_of_week

```
# Calculate the average ride_length for member_casual = casual by day_of_week

average_ride_length_C <- merged_data %>%
  filter(member_casual == "casual") %>%
  group_by(ride_weekday) %>%
  summarise(avg_ride_length_C = mean(ride_length_sec / 60, na.rm = TRUE))

# Print the result
print ("average ride_length in minutes for casuals by day_of_week ")
## [1] "average ride_length in minutes for casuals by day_of_week "
print(average_ride_length_C)
## # A tibble: 7 × 2
##   ride_weekday avg_ride_length_C
##   <chr>                    <dbl>
## 1 Friday                    22.4
```

```
## 2 Monday                    22.9
## 3 Saturday                  26.9
## 4 Sunday                    27.9
## 5 Thursday                  19.8
## 6 Tuesday                   20.3
## 7 Wednesday                 21.2
```

Calculate the number of rides for users by day_of_week by adding Count of trip_id to Values.

```
# Calculate the total number of trips when member_casual equals "member" by
ride_weekday

total_trips <- merged_data %>%
  filter(member_casual == "member") %>%
  group_by(ride_weekday) %>%
  summarise(total_count = n())

# Print the result
print("total trips members by days of week")
## [1] "total trips members by days of week"
print(total_trips)
## # A tibble: 7 × 2
##   ride_weekday total_count
##   <chr>              <int>
## 1 Friday            237995
## 2 Monday            261675
## 3 Saturday          207807
## 4 Sunday            198359
## 5 Thursday          282269
## 6 Tuesday           283112
## 7 Wednesday         294148
```

Calculate the number of rides for casuals by day_of_week by adding Count of trip_id to Values.

```
# Calculate the total number of trips when member_casual equals "member" by
ride_weekday

total_trips <- merged_data %>%
  filter(member_casual == "casual") %>%
  group_by(ride_weekday) %>%
  summarise(total_count = n())

# Print the result
print("total trips casuals by days of week")
## [1] "total trips casuals by days of week"
print(total_trips)
## # A tibble: 7 × 2
##   ride_weekday total_count
##   <chr>              <int>
## 1 Friday            110606
```

```
## 2 Monday              98368
## 3 Saturday           161568
## 4 Sunday             152451
## 5 Thursday            99216
## 6 Tuesday             94566
## 7 Wednesday          103914
```

Sort the dataframe by the "Score" column in descending order using dplyr

```
merged_data  <- merged_data %>% arrange(desc(ride_length_sec))
#View(merged_data)
```

Descriptive analysis on ride_length (figures in seconds are being changed to minutes)

Mean Value in minutes

```
# Note result in minutes
mean(merged_data$ride_length_sec)/60 #straight average (total ride length /
rides)
## [1] 15.88136
```

Median Value in Minutes

```
median(merged_data$ride_length_sec)/60  #midpoint number in the ascending
array of ride lengths
## [1] 10
```

MAX value was calculated in hours

```
max(merged_data$ride_length_sec)/3600 #longest ride
## [1] 25.16667
```

Min Value calculated in seconds

```
min(merged_data$ride_length_sec) #shortest ride
## [1] 60
```

You can condense the four lines above to one line using summary() on the specific attribute

```
summary(merged_data$ride_length_sec)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    60.0   360.0   600.0   952.9  1020.0 90600.0
summary(merged_data$ride_length_sec/60)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    6.00   10.00   15.88   17.00 1510.00
summary(merged_data$ride_length_sec/3600)
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.01667  0.10000  0.16667  0.26469  0.28333 25.16667
```

Compare members and casual users

```
aggregate(merged_data$ride_length ~ merged_data$member_casual, FUN = mean)
##   merged_data$member_casual merged_data$ride_length
## 1                    casual          1420.1458 secs
## 2                    member           735.6586 secs
aggregate(merged_data$ride_length ~ merged_data$member_casual, FUN = median)
```

```
##   merged_data$member_casual merged_data$ride_length
## 1                    casual                780 secs
## 2                    member                540 secs
aggregate(merged_data$ride_length ~ merged_data$member_casual, FUN = max)
##   merged_data$member_casual merged_data$ride_length
## 1                    casual              90600 secs
## 2                    member              89880 secs
aggregate(merged_data$ride_length ~ merged_data$member_casual, FUN = min)
##   merged_data$member_casual merged_data$ride_length
## 1                    casual                 60 secs
## 2                    member                 60 secs
```

See the average ride time by each day for members vs casual users

```
aggregate(merged_data$ride_length ~ merged_data$member_casual +
merged_data$day_of_week, FUN = mean)
##    merged_data$member_casual merged_data$day_of_week
merged_data$ride_length
## 1                     casual                  Friday          1341.8147
secs
## 2                     member                  Friday           705.7202
secs
## 3                     casual                  Monday          1376.7453
secs
## 4                     member                  Monday           705.4870
secs
## 5                     casual                Saturday          1616.2734
secs
## 6                     member                Saturday           822.0136
secs
## 7                     casual                  Sunday          1674.3165
secs
## 8                     member                  Sunday           832.0220
secs
## 9                     casual                Thursday          1189.1467
secs
## 10                    member                Thursday           698.3024
secs
## 11                    casual                 Tuesday          1219.3249
secs
## 12                    member                 Tuesday           710.7073
secs
## 13                    casual               Wednesday          1270.0814
secs
## 14                    member               Wednesday           720.5954
secs
```

Notice that the days of the week are out of order. Let's fix that.

```
merged_data$day_of_week <- ordered(merged_data$day_of_week,
levels=c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"))
```

Now, let's run the average ride time by each day for members vs casual users

```
aggregate(merged_data$ride_length ~ merged_data$member_casual +
merged_data$day_of_week, FUN = mean)
##    merged_data$member_casual merged_data$day_of_week
merged_data$ride_length
## 1                    casual                  Sunday          1674.3165
secs
## 2                    member                  Sunday           832.0220
secs
## 3                    casual                  Monday          1376.7453
secs
## 4                    member                  Monday           705.4870
secs
## 5                    casual                 Tuesday          1219.3249
secs
## 6                    member                 Tuesday           710.7073
secs
## 7                    casual               Wednesday          1270.0814
secs
## 8                    member               Wednesday           720.5954
secs
## 9                    casual                Thursday          1189.1467
secs
## 10                   member                Thursday           698.3024
secs
## 11                   casual                  Friday          1341.8147
secs
## 12                   member                  Friday           705.7202
secs
## 13                   casual                Saturday          1616.2734
secs
## 14                   member                Saturday           822.0136
secs
```

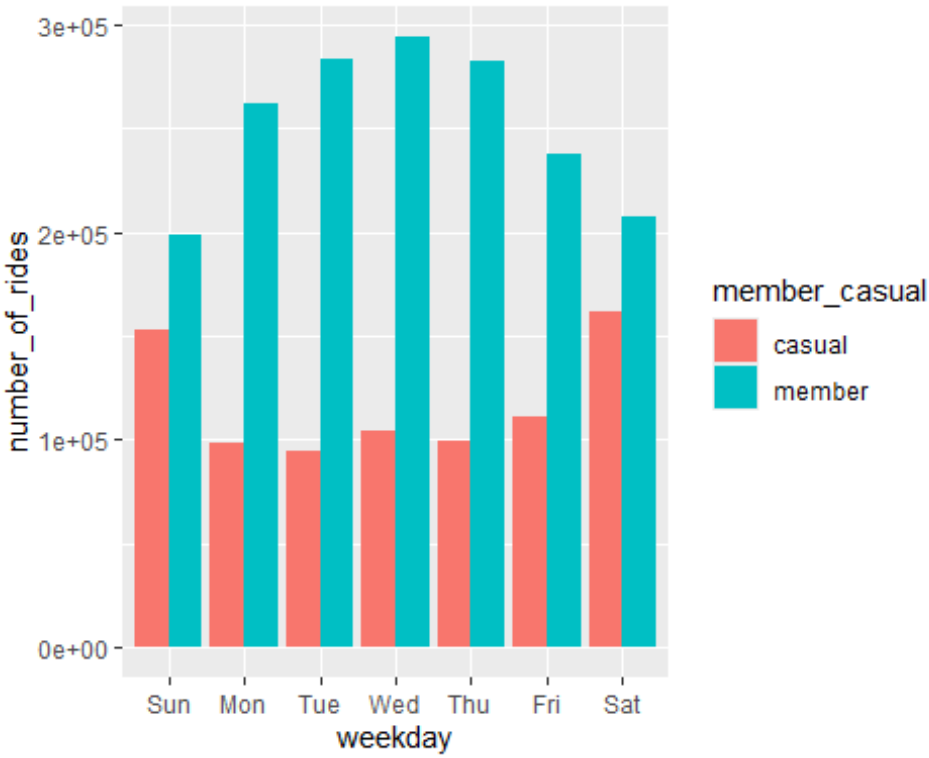
analyze ridership data by type and weekday

```
merged_data %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%  #creates weekday
field using wday()
  group_by(member_casual, weekday) %>%  #groups by usertype and weekday
  summarise(number_of_rides = n()                           #calculates the
number of rides and average duration
            ,average_duration = mean(ride_length)) %>%      # calculates the
average duration
  arrange(member_casual, weekday)                               # sorts
## `summarise()` has grouped output by 'member_casual'. You can override
using the
## `.groups` argument.
```

```
## # A tibble: 14 × 4
## # Groups:   member_casual [2]
##    member_casual weekday number_of_rides average_duration
##    <chr>         <ord>             <int> <drtn>
##  1 casual        Sun              152451 1674.3165 secs
##  2 casual        Mon               98368 1376.7453 secs
##  3 casual        Tue               94566 1219.3249 secs
##  4 casual        Wed              103914 1270.0814 secs
##  5 casual        Thu               99216 1189.1467 secs
##  6 casual        Fri              110606 1341.8147 secs
##  7 casual        Sat              161568 1616.2734 secs
##  8 member        Sun              198359  832.0220 secs
##  9 member        Mon              261675  705.4870 secs
## 10 member        Tue              283112  710.7073 secs
## 11 member        Wed              294148  720.5954 secs
## 12 member        Thu              282269  698.3024 secs
## 13 member        Fri              237995  705.7202 secs
## 14 member        Sat              207807  822.0136 secs
```

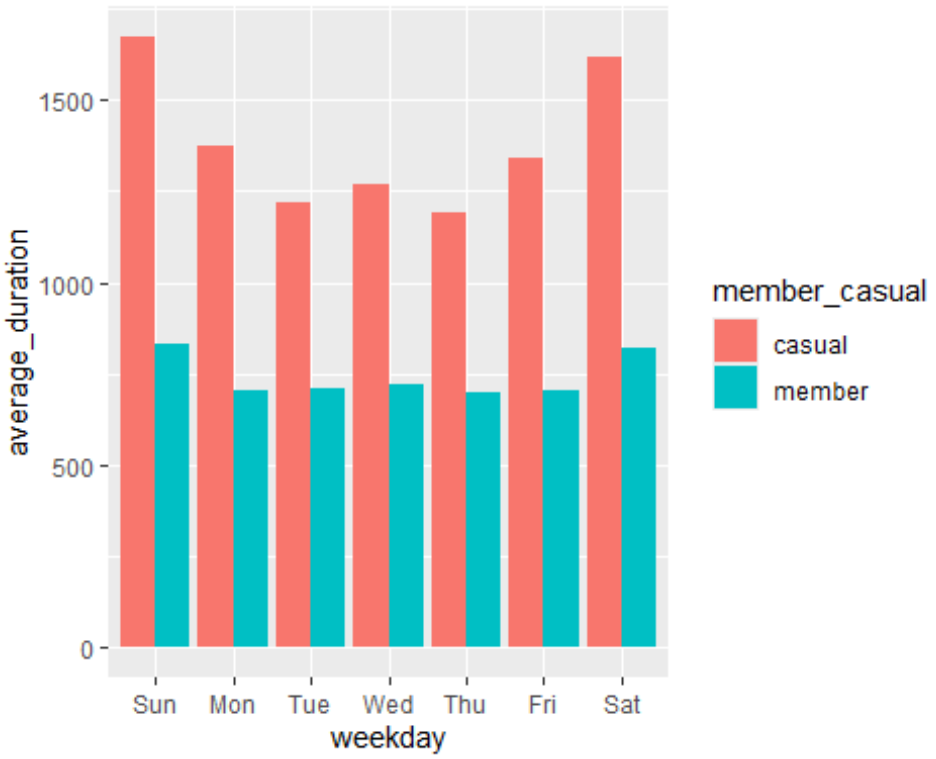
Let's visualize the number of rides by rider type

```
merged_data %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday)  %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge")
## `summarise()` has grouped output by 'member_casual'. You can override
using the
## `.groups` argument.
```



Let's create a visualization for average duration

```
merged_data %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday)  %>%
  ggplot(aes(x = weekday, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge")
## `summarise()` has grouped output by 'member_casual'. You can override
using the
## `.groups` argument.
## Don't know how to automatically pick scale for object of type <difftime>.
## Defaulting to continuous.
```

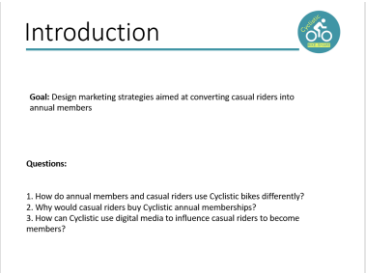


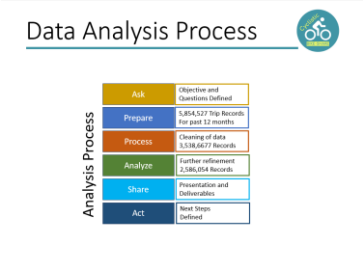
#========== # STEP 5: EXPORT SUMMARY FILE FOR FURTHER ANALYSIS #========== # Create a csv file that we will visualize in Excel, Tableau, or my presentation software # N.B.: This file location is for a Mac. If you are working on a PC, change the file location accordingly (most likely "C:_USERNAME...") to export the data. You can read more here: https://datatofish.com/export-dataframe-to-csv-in-r/

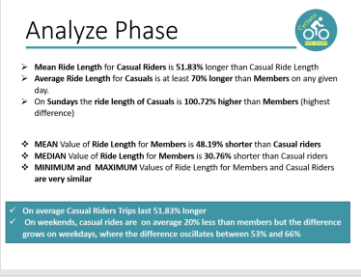
```
final_counts <- aggregate(merged_data$ride_length ~ merged_data$member_casual
+ merged_data$day_of_week, FUN = mean)
write.csv(final_counts, file = 'final_avg_ride_length.csv')
```

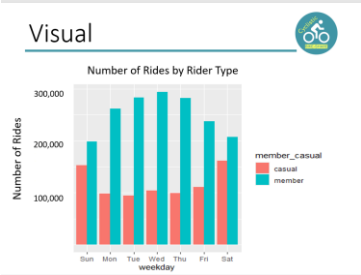
Deliverable 05. Supporting visualizations and key findings

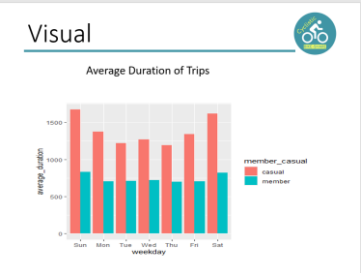
Power Point Presentation Attached

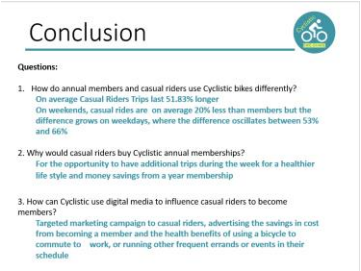


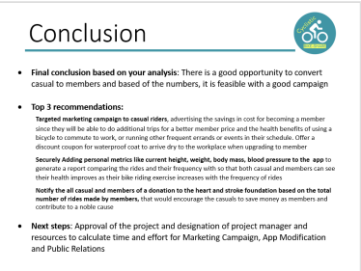












Deliverable 06. Top three recommendations based on your analysis

Analysis Fundamental Questions

1. How do annual members and casual riders use Cyclistic bikes differently?
On average Casual Riders Trips last 51.83% longer
On weekends, casual rides are on average 20% less than members but the difference grows on weekdays, where the difference oscillates between 53% and 66%

2. Why would casual riders buy Cyclistic annual memberships?

For the opportunity to have additional trips during the week for a healthier life style and money savings from a year membership

3. How can Cyclistic use digital media to influence casual riders to become members?
Targeted marketing campaign to casual riders, advertising the savings in cost for becoming a member and the health benefits of using a bicycle to commute to work, or running other frequent errands or events in their schedule

Top Three recommendations

Targeted marketing campaign to casual riders, advertising the savings in cost for becoming a member since they will be able to do additional trips for a better member price and the health benefits of using a bicycle to commute to work, or running other frequent errands or events in their schedule. Offer a discount coupon for waterproof coat to arrive dry to the workplace when upgrading to member

Securely Adding personal metrics like current height, weight, body mass, blood pressure in the app to generate a report comparing the rides and their frequency with so that both casual and members can see their health improves as their bike riding exercise increases with the frequency of rides

Notify the all casual and members of a donation to the heart and stroke foundation based on the total number of rides made by members, that would encourage the casuals to save money as members and contribute to a noble cause

Note the top three recommendations have been added to the final presentation.