How Does a Bike-share Navigate Speedy Success?

Yi Wang

9/7/2021

```
library(tidyverse)
library(ggplot2)
library(here)
library(skimr)
library(janitor)
library(lubridate)
```

1 ASK

1.1 What is the problem you are trying to solve?

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

1.2 How can your insights drive business decisions?

The analysis finding may be used for the company to make decisions as to influence casual riders to become members.

1.3 (Deliverable) A clear statement of the business task

Identify the difference of annual members and casual riders using Cyclistic's bikes.

2 PREPARE

2.1 Where is your data located?

We will use Cyclistic's historical trip data: the previous 12 months of Cyclistic trip data to analyze and identify trends. The data has been made available by Motivate International Inc. under this license.) This is public data that can be used to explore how different customer types are using Cyclistic bikes.

Four zipped data sets (Q2, Q3, Q4 of 2019 and Q1 of 2020) are downloaded to a local folder named `data` and unzipped.

2.2 How is the data organized?

It appears that the four data sets are not organized in the same way. We next skim each data set.

```
head(Q2_2019)
```

```
## # A tibble: 6 x 12
##   `01 - Rental De~ `01 - Rental Detai~ `01 - Rental Detai~ `01 - Rental De~
##              <dbl> <dttm>              <dttm>                         <dbl>
## 1         22178529 2019-04-01 00:02:22 2019-04-01 00:09:48             6251
```

```
## 2        22178530 2019-04-01 00:03:02 2019-04-01 00:20:30             6226
## 3        22178531 2019-04-01 00:11:07 2019-04-01 00:15:19             5649
## 4        22178532 2019-04-01 00:13:01 2019-04-01 00:18:58             4151
## 5        22178533 2019-04-01 00:19:26 2019-04-01 00:36:13             3270
## 6        22178534 2019-04-01 00:19:39 2019-04-01 00:23:56             3123
## # ... with 8 more variables: ‘01 - Rental Details Duration In Seconds
## #   Uncapped‘ <dbl>, ‘03 - Rental Start Station ID‘ <dbl>, ‘03 - Rental Start
## #   Station Name‘ <chr>, ‘02 - Rental End Station ID‘ <dbl>, ‘02 - Rental End
## #   Station Name‘ <chr>, ‘User Type‘ <chr>, ‘Member Gender‘ <chr>, ‘05 - Member
## #   Details Member Birthday Year‘ <dbl>
```

```
skim_without_charts(Q2_2019)
```

Table 1: Data summary

Name	Q2_2019
Number of rows	1108163
Number of columns	12
Column type frequency:	
character	4
numeric	6
POSIXct	2
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
03 - Rental Start Station Name	0	1.00	10	43	0	610	0
02 - Rental End Station Name	0	1.00	10	43	0	612	0
User Type	0	1.00	8	10	0	2	0
Member Gender	185554	0.83	4	6	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
01 - Rental Details Rental ID	0	1.00	22829855.71	375397.70	22178529	22505075	22831158	23154496	23479387
01 - Rental Details Bike ID	0	1.00	3398.80	1907.73	1	1738	3471	5080	6471
01 - Rental Details Duration In Seconds Uncapped	0	1.00	1327.29	13393.78	61	426	742	1347	4757640
03 - Rental Start Station ID	0	1.00	200.30	154.85	1	77	174	289	669
02 - Rental End Station ID	0	1.00	201.27	155.01	1	77	174	290	669
05 - Member Details Member Birthday Year	180953	0.84	1983.94	10.80	1759	1979	1987	1992	2014

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
01 - Rental Details Local Start Time	0	1	2019-04-01 00:02:22	2019-06-30 23:59:05	2019-05-26 11:38:51	957330
01 - Rental Details Local End Time	0	1	2019-04-01 00:09:48	2019-07-06 14:22:25	2019-05-26 12:03:20	922638

```
colnames(Q2_2019)
```

```
##  [1] "01 - Rental Details Rental ID"
##  [2] "01 - Rental Details Local Start Time"
##  [3] "01 - Rental Details Local End Time"
##  [4] "01 - Rental Details Bike ID"
##  [5] "01 - Rental Details Duration In Seconds Uncapped"
##  [6] "03 - Rental Start Station ID"
##  [7] "03 - Rental Start Station Name"
##  [8] "02 - Rental End Station ID"
##  [9] "02 - Rental End Station Name"
## [10] "User Type"
## [11] "Member Gender"
## [12] "05 - Member Details Member Birthday Year"
```

For the data set of 2019_Q2, it is organized by ascending `Rental ID` which is also ordered in the ascending order of `Start Time` of a trip.

```
head(Q3_2019)
```

```
## # A tibble: 6 x 12
##   trip_id start_time          end_time            bikeid tripduration
##     <dbl> <dttm>              <dttm>               <dbl>        <dbl>
## 1  2.35e7 2019-07-01 00:00:27 2019-07-01 00:20:41   3591         1214
## 2  2.35e7 2019-07-01 00:01:16 2019-07-01 00:18:44   5353         1048
## 3  2.35e7 2019-07-01 00:01:48 2019-07-01 00:27:42   6180         1554
## 4  2.35e7 2019-07-01 00:02:07 2019-07-01 00:27:10   5540         1503
## 5  2.35e7 2019-07-01 00:02:13 2019-07-01 00:22:26   6014         1213
## 6  2.35e7 2019-07-01 00:02:21 2019-07-01 00:07:31   4941          310
## # ... with 7 more variables: from_station_id <dbl>, from_station_name <chr>,
## #   to_station_id <dbl>, to_station_name <chr>, usertype <chr>, gender <chr>,
## #   birthyear <dbl>
```

```
skim_without_charts(Q3_2019)
```

Table 5: Data summary

Name	Q3_2019
Number of rows	1640718
Number of columns	12
Column type frequency:	
character	4
numeric	6
POSIXct	2
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
from_station_name	0	1.00	10	43	0	612	0
to_station_name	0	1.00	10	43	0	613	0
usertype	0	1.00	8	10	0	2	0
gender	287350	0.82	4	6	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75
trip_id	0	1.00	24364471.07	499548.45	23479388	23935498	24367416	24797401
bikeid	0	1.00	3349.86	1888.88	1	1713	3419	4997
tripduration	0	1.00	1741.74	38503.44	61	465	813	1460
from_station_id	0	1.00	202.40	156.72	2	77	174	289
to_station_id	0	1.00	203.90	156.70	2	80	176	291
birthyear	278094	0.83	1984.90	10.61	1888	1980	1988	1992

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
start_time	0	1	2019-07-01 00:00:27	2019-09-30 23:59:37	2019-08-14 07:11:50	1372358
end_time	0	1	2019-07-01 00:07:31	2019-11-04 08:09:47	2019-08-14 07:28:07	1344539

```
colnames(Q3_2019)
```

```
##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "bikeid"            "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "gender"            "birthyear"
```

```
head(Q4_2019)
```

```
## # A tibble: 6 x 12
##   trip_id start_time          end_time            bikeid tripduration
##     <dbl> <dttm>              <dttm>               <dbl>        <dbl>
## 1  2.35e7 2019-07-01 00:00:27 2019-07-01 00:20:41   3591         1214
## 2  2.35e7 2019-07-01 00:01:16 2019-07-01 00:18:44   5353         1048
## 3  2.35e7 2019-07-01 00:01:48 2019-07-01 00:27:42   6180         1554
## 4  2.35e7 2019-07-01 00:02:07 2019-07-01 00:27:10   5540         1503
## 5  2.35e7 2019-07-01 00:02:13 2019-07-01 00:22:26   6014         1213
## 6  2.35e7 2019-07-01 00:02:21 2019-07-01 00:07:31   4941          310
## # ... with 7 more variables: from_station_id <dbl>, from_station_name <chr>,
## #   to_station_id <dbl>, to_station_name <chr>, usertype <chr>, gender <chr>,
## #   birthyear <dbl>
```

```
skim_without_charts(Q4_2019)
```

Table 9: Data summary

Name	Q4_2019
Number of rows	1640718
Number of columns	12
Column type frequency:	
character	4
numeric	6
POSIXct	2
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
from_station_name	0	1.00	10	43	0	612	0
to_station_name	0	1.00	10	43	0	613	0
usertype	0	1.00	8	10	0	2	0
gender	287350	0.82	4	6	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75
trip_id	0	1.00	24364471.07	499548.45	23479388	23935498	24367416	24797401
bikeid	0	1.00	3349.86	1888.88	1	1713	3419	4997
tripduration	0	1.00	1741.74	38503.44	61	465	813	1460
from_station_id	0	1.00	202.40	156.72	2	77	174	289
to_station_id	0	1.00	203.90	156.70	2	80	176	291
birthyear	278094	0.83	1984.90	10.61	1888	1980	1988	1992

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
start_time	0	1	2019-07-01 00:00:27	2019-09-30 23:59:37	2019-08-14 07:11:50	1372358
end_time	0	1	2019-07-01 00:07:31	2019-11-04 08:09:47	2019-08-14 07:28:07	1344539

```
colnames(Q4_2019)
```

```
##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "bikeid"            "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "gender"            "birthyear"
```

For the data sets of 2019_Q3 and 2019_Q4, they are organized by ascending `trip_id` which is also ordered in the ascending order of `start_time` of a trip.

```r
head(Q1_2020)
```

```
## # A tibble: 6 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 EACB19~ docked_bike   2020-01-21 20:06:59 2020-01-21 20:14:30 Western Ave & L~
## 2 8FED87~ docked_bike   2020-01-30 14:22:39 2020-01-30 14:26:22 Clark St & Mont~
## 3 789F3C~ docked_bike   2020-01-09 19:29:26 2020-01-09 19:32:17 Broadway & Belm~
## 4 C9A388~ docked_bike   2020-01-06 16:17:07 2020-01-06 16:25:56 Clark St & Rand~
## 5 943BC3~ docked_bike   2020-01-30 08:37:16 2020-01-30 08:42:48 Clinton St & La~
## 6 6D9C8A~ docked_bike   2020-01-10 12:33:05 2020-01-10 12:37:54 Wells St & Hubb~
## # ... with 8 more variables: start_station_id <dbl>, end_station_name <chr>,
## #   end_station_id <dbl>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

```r
glimpse(Q1_2020)
```

```
## Rows: 426,887
## Columns: 13
## $ ride_id            <chr> "EACB19130B0CDA4A", "8FED874C809DC021", "789F3C2...
## $ rideable_type      <chr> "docked_bike", "docked_bike", "docked_bike", "do...
## $ started_at         <dttm> 2020-01-21 20:06:59, 2020-01-30 14:22:39, 2020-...
## $ ended_at           <dttm> 2020-01-21 20:14:30, 2020-01-30 14:26:22, 2020-...
## $ start_station_name <chr> "Western Ave & Leland Ave", "Clark St & Montrose...
## $ start_station_id   <dbl> 239, 234, 296, 51, 66, 212, 96, 96, 212, 38, 117...
## $ end_station_name   <chr> "Clark St & Leland Ave", "Southport Ave & Irving...
## $ end_station_id     <dbl> 326, 318, 117, 24, 212, 96, 212, 212, 96, 100, 6...
## $ start_lat          <dbl> 41.9665, 41.9616, 41.9401, 41.8846, 41.8856, 41....
## $ start_lng          <dbl> -87.6884, -87.6660, -87.6455, -87.6319, -87.6418...
## $ end_lat            <dbl> 41.9671, 41.9542, 41.9402, 41.8918, 41.8899, 41....
## $ end_lng            <dbl> -87.6674, -87.6644, -87.6530, -87.6206, -87.6343...
## $ member_casual      <chr> "member", "member", "member", "member", "member"...
```

```r
skim_without_charts(Q1_2020)
```

Table 13: Data summary

Name	Q1_2020
Number of rows	426887
Number of columns	13
Column type frequency:	
character	5
numeric	6
POSIXct	2
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ride_id	0	1	16	16	0	426887	0
rideable_type	0	1	11	11	0	1	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
start_station_name	0	1	5	43	0	607	0
end_station_name	1	1	5	43	0	602	0
member_casual	0	1	6	6	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
start_station_id	0	1	209.80	163.22	2.00	77.00	176.00	298.00	675.00
end_station_id	1	1	209.34	163.20	2.00	77.00	175.00	297.00	675.00
start_lat	0	1	41.90	0.04	41.74	41.88	41.89	41.92	42.06
start_lng	0	1	-87.64	0.02	-87.77	-87.66	-87.64	-87.63	-87.55
end_lat	1	1	41.90	0.04	41.74	41.88	41.89	41.92	42.06
end_lng	1	1	-87.64	0.02	-87.77	-87.66	-87.64	-87.63	-87.55

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
started_at	0	1	2020-01-01 00:04:44	2020-03-31 23:51:34	2020-02-17 05:01:27	399265
ended_at	0	1	2020-01-01 00:10:54	2020-05-19 20:10:34	2020-02-17 05:48:58	399532

```
colnames(Q1_2020)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "start_station_id"
##  [7] "end_station_name"   "end_station_id"     "start_lat"
## [10] "start_lng"          "end_lat"            "end_lng"
## [13] "member_casual"
```

For the data sets of 2020_Q1, it appears the data are not organized at all. The bike rental records seem to appear in random order in the data file.

2.3 Are there issues with bias or credibility in this data? Does your data ROCCC?

It appears there is no issue of bias or credibility in the data set. However, we notice that from the reports by `skim_without_charts`, the three data sets for 2019 have missing values in the columns of `gendre` and `birthyear`, and the data set Q1-2020 has one missing value in four columns regarding `end_station`. We will find out more about the missing values and determine how to deal with them.

2.4 How are you addressing licensing, privacy, security, and accessibility?

Note that data-privacy issues prohibit one from using riders' personally identifiable information. This means that it is not possible to connect pass purchases to credit card numbers to determine if casual riders live in the Cyclistic service area or if they have purchased multiple single passes.

The data has been made available by Motivate International Inc. under this license.) This is public data that can be used to explore how different customer types are using Cyclistic bikes.

2.5 How did you verify the data's integrity?

We need to examine the following: 1. if there are missing values in a data set, where they are and how to handle them. 2. if the four data sets have consistent columns:column names and column data type.

2.6 How does it help you answer your question?

Checking data integrity is important to make sure what data is used to process, and allow combining the four data sets into one data set to analyze.

2.7 Are there any problems with the data?

A quick glance of the data indicates that there are some differences among the four data files. The three data sets for 2019 have the same number of columns, but 2019-Q2 uses different column names. 2019-Q3 and 2019-Q4 use identical column names. The good news is that the column names of 2019-Q2 match the corresponding column names in 2019-Q3 and 2019-Q4 in terms what information is recorded in each column.

2020-Q1 has one more column compared to the other three files. Not only it uses different column names, it doesn't have the exact matching columns. But it does contain the essential columns that we need for this analysis.

A quick glance of the data indicates there is no obvious bias or credibility in the data. However, the column `gender` and `birthyear` has some missing values for small number of records. Those missing values may not be significant for our task.

2.8 (Deliverable) A description of all data sources used

The Cyclistic's historical trip data: the previous 12 months of Cyclistic trip data is used. The data has been made available by Motivate International Inc. under this license.) This is public data that can be used to explore how different customer types are using Cyclistic bikes.

3 PROCESS

3.1 What tools are you choosing and why?

I decided to use R, as R can provide more powerful tools. Besides the data set is large. When using Excel to open the files Q2-2019 and Q3-2019, the Excel reports that the file is not loaded completely. This is because the data set exceeds the limit of 1,048,576 rows for Excel to process. On the other hand, R can readily hand much larger a data set.

3.2 Have you ensured your data's integrity?

To this end, we first check if there are missing values (`NA`) in the data sets.

```
apply(Q2_2019,2, function(x) {sum(is.na(x))})
```

```
##                     01 - Rental Details Rental ID
##                                                 0
##              01 - Rental Details Local Start Time
##                                                 0
##                01 - Rental Details Local End Time
##                                                 0
##                       01 - Rental Details Bike ID
##                                                 0
## 01 - Rental Details Duration In Seconds Uncapped
##                                                 0
```

```
##                      03 - Rental Start Station ID
##                                                 0
##                    03 - Rental Start Station Name
##                                                 0
##                        02 - Rental End Station ID
##                                                 0
##                      02 - Rental End Station Name
##                                                 0
##                                         User Type
##                                                 0
##                                     Member Gender
##                                            185554
##          05 - Member Details Member Birthday Year
##                                            180953
```

```
apply(Q3_2019,2, function(x) {sum(is.na(x))})
```

```
##           trip_id        start_time          end_time            bikeid
##                 0                 0                 0                 0
##      tripduration   from_station_id from_station_name     to_station_id
##                 0                 0                 0                 0
##   to_station_name          usertype            gender         birthyear
##                 0                 0            287350            278094
```

```
apply(Q4_2019,2, function(x) {sum(is.na(x))})
```

```
##           trip_id        start_time          end_time            bikeid
##                 0                 0                 0                 0
##      tripduration   from_station_id from_station_name     to_station_id
##                 0                 0                 0                 0
##   to_station_name          usertype            gender         birthyear
##                 0                 0            287350            278094
```

```
apply(Q1_2020,2, function(x) {sum(is.na(x))})
```

```
##            ride_id      rideable_type         started_at           ended_at
##                  0                  0                  0                  0
## start_station_name   start_station_id   end_station_name     end_station_id
##                  0                  0                  1                  1
##          start_lat          start_lng            end_lat            end_lng
##                  0                  0                  1                  1
##      member_casual
##                  0
```

It appears that the three data sets for 2019 has missing values, but only in columns of `gendre` and `birthyear` which are not the interest of this analysis hence those missing values will not affect our analysis.

For the data set Q1_2020, it appears it has only one missing value in each of the columns of `end_station_name`, `end_station_id`, `end_lat` and `end_lng`. To find out where exactly the missing values are, we perform the following operation:

```
na_rows <- Q1_2020[is.na(Q1_2020$end_station_name),]
na_rows
```

```
## # A tibble: 1 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 157EAA~ docked_bike   2020-03-16 11:23:36 2020-03-16 11:23:24 HQ QR
```

```
## # ... with 8 more variables: start_station_id <dbl>, end_station_name <chr>,
## #   end_station_id <dbl>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

So there is only one row of record with the four missing column values. Due to the desired analysis will not use those information, we decide to keep this line of record.

3.3 What steps have you taken to ensure that your data is clean?

As mentioned before, the column names need to be unified. We will retain the column names of Q3_2019 and Q4_2019 and change the column names of Q2_2019 and Q1_2020 to the column names used in Q3_2019 and Q4_2019.

We first clean the column names of Q3_2019 and Q4_2019.

```
clean_names(Q3_2019)
```

```
## # A tibble: 1,640,718 x 12
##    trip_id start_time          end_time            bikeid tripduration
##      <dbl> <dttm>              <dttm>               <dbl>        <dbl>
##  1  2.35e7 2019-07-01 00:00:27 2019-07-01 00:20:41   3591         1214
##  2  2.35e7 2019-07-01 00:01:16 2019-07-01 00:18:44   5353         1048
##  3  2.35e7 2019-07-01 00:01:48 2019-07-01 00:27:42   6180         1554
##  4  2.35e7 2019-07-01 00:02:07 2019-07-01 00:27:10   5540         1503
##  5  2.35e7 2019-07-01 00:02:13 2019-07-01 00:22:26   6014         1213
##  6  2.35e7 2019-07-01 00:02:21 2019-07-01 00:07:31   4941          310
##  7  2.35e7 2019-07-01 00:02:24 2019-07-01 00:23:12   3770         1248
##  8  2.35e7 2019-07-01 00:02:26 2019-07-01 00:28:16   5442         1550
##  9  2.35e7 2019-07-01 00:02:34 2019-07-01 00:28:57   2957         1583
## 10  2.35e7 2019-07-01 00:02:45 2019-07-01 00:29:14   6091         1589
## # ... with 1,640,708 more rows, and 7 more variables: from_station_id <dbl>,
## #   from_station_name <chr>, to_station_id <dbl>, to_station_name <chr>,
## #   usertype <chr>, gender <chr>, birthyear <dbl>
```

```
clean_names(Q4_2019)
```

```
## # A tibble: 1,640,718 x 12
##    trip_id start_time          end_time            bikeid tripduration
##      <dbl> <dttm>              <dttm>               <dbl>        <dbl>
##  1  2.35e7 2019-07-01 00:00:27 2019-07-01 00:20:41   3591         1214
##  2  2.35e7 2019-07-01 00:01:16 2019-07-01 00:18:44   5353         1048
##  3  2.35e7 2019-07-01 00:01:48 2019-07-01 00:27:42   6180         1554
##  4  2.35e7 2019-07-01 00:02:07 2019-07-01 00:27:10   5540         1503
##  5  2.35e7 2019-07-01 00:02:13 2019-07-01 00:22:26   6014         1213
##  6  2.35e7 2019-07-01 00:02:21 2019-07-01 00:07:31   4941          310
##  7  2.35e7 2019-07-01 00:02:24 2019-07-01 00:23:12   3770         1248
##  8  2.35e7 2019-07-01 00:02:26 2019-07-01 00:28:16   5442         1550
##  9  2.35e7 2019-07-01 00:02:34 2019-07-01 00:28:57   2957         1583
## 10  2.35e7 2019-07-01 00:02:45 2019-07-01 00:29:14   6091         1589
## # ... with 1,640,708 more rows, and 7 more variables: from_station_id <dbl>,
## #   from_station_name <chr>, to_station_id <dbl>, to_station_name <chr>,
## #   usertype <chr>, gender <chr>, birthyear <dbl>
```

We then change the column names of Q2_2019.

```
Q2_2019_new <- Q2_2019  # preserve the original data
colnames(Q2_2019_new) <- colnames(Q3_2019)
colnames(Q2_2019_new)
```

```
##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "bikeid"            "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "gender"            "birthyear"
```

```
clean_names(Q2_2019_new)
```

```
## # A tibble: 1,108,163 x 12
##    trip_id start_time          end_time            bikeid tripduration
##      <dbl> <dttm>              <dttm>               <dbl>        <dbl>
##  1  2.22e7 2019-04-01 00:02:22 2019-04-01 00:09:48   6251          446
##  2  2.22e7 2019-04-01 00:03:02 2019-04-01 00:20:30   6226         1048
##  3  2.22e7 2019-04-01 00:11:07 2019-04-01 00:15:19   5649          252
##  4  2.22e7 2019-04-01 00:13:01 2019-04-01 00:18:58   4151          357
##  5  2.22e7 2019-04-01 00:19:26 2019-04-01 00:36:13   3270         1007
##  6  2.22e7 2019-04-01 00:19:39 2019-04-01 00:23:56   3123          257
##  7  2.22e7 2019-04-01 00:26:33 2019-04-01 00:35:41   6418          548
##  8  2.22e7 2019-04-01 00:29:48 2019-04-01 00:36:11   4513          383
##  9  2.22e7 2019-04-01 00:32:07 2019-04-01 01:07:44   3280         2137
## 10  2.22e7 2019-04-01 00:32:19 2019-04-01 01:07:39   5534         2120
## # ... with 1,108,153 more rows, and 7 more variables: from_station_id <dbl>,
## #   from_station_name <chr>, to_station_id <dbl>, to_station_name <chr>,
## #   usertype <chr>, gender <chr>, birthyear <dbl>
```

Since the column names of Q1_2020 do not completely match those of Q3_2019, we need do the renaming one-by_one for Q1_2020.

```
Q1_2020_copy <- Q1_2020 # preserve the original data
Q1_2020_copy <- Q1_2020_copy %>%
                rename(trip_id = ride_id) %>%
                rename(start_time = started_at) %>%
                rename(end_time = ended_at) %>%
                rename(from_station_name = start_station_name) %>%
                rename(from_station_id = start_station_id) %>%
                rename(to_station_name = end_station_name) %>%
                rename(to_station_id = end_station_id) %>%
                rename(usertype = member_casual)
colnames(Q1_2020_copy)
```

```
##  [1] "trip_id"           "rideable_type"     "start_time"
##  [4] "end_time"          "from_station_name" "from_station_id"
##  [7] "to_station_name"   "to_station_id"     "start_lat"
## [10] "start_lng"         "end_lat"           "end_lng"
## [13] "usertype"
```

```
clean_names(Q1_2020_copy)
```

```
## # A tibble: 426,887 x 13
##    trip_id rideable_type start_time          end_time
##    <chr>   <chr>         <dttm>              <dttm>
##  1 EACB19~ docked_bike   2020-01-21 20:06:59 2020-01-21 20:14:30
##  2 8FED87~ docked_bike   2020-01-30 14:22:39 2020-01-30 14:26:22
##  3 789F3C~ docked_bike   2020-01-09 19:29:26 2020-01-09 19:32:17
##  4 C9A388~ docked_bike   2020-01-06 16:17:07 2020-01-06 16:25:56
##  5 943BC3~ docked_bike   2020-01-30 08:37:16 2020-01-30 08:42:48
```

```
##   6 6D9C8A~ docked_bike   2020-01-10 12:33:05 2020-01-10 12:37:54
##   7 31EB9B~ docked_bike   2020-01-10 13:07:35 2020-01-10 13:12:24
##   8 A2B24E~ docked_bike   2020-01-10 07:24:53 2020-01-10 07:29:50
##   9 5E3F01~ docked_bike   2020-01-31 16:37:16 2020-01-31 16:42:11
## 10 19DC57~ docked_bike   2020-01-31 09:39:17 2020-01-31 09:42:40
## # ... with 426,877 more rows, and 9 more variables: from_station_name <chr>,
## #   from_station_id <dbl>, to_station_name <chr>, to_station_id <dbl>,
## #   start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, end_lng <dbl>,
## #   usertype <chr>
```

The Q1_2020 dataset does not have a trip_duration column. We next add this column into the dataset.

```
Q1_2020_copy <- Q1_2020_copy %>% mutate(tripduration = end_time - start_time )
colnames(Q1_2020_copy)
```

```
##  [1] "trip_id"           "rideable_type"     "start_time"
##  [4] "end_time"          "from_station_name" "from_station_id"
##  [7] "to_station_name"   "to_station_id"     "start_lat"
## [10] "start_lng"         "end_lat"           "end_lng"
## [13] "usertype"          "tripduration"
```

We also reorder the columns of Q1_2020 so that the columns match those in the other three data sets in the same order.

```
Q1_2020_new <- Q1_2020_copy[,c(1,3,4,2,14,6,5,8,7,13,9,10,11,12)]
colnames(Q1_2020_new)
```

```
##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "rideable_type"     "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "start_lat"         "start_lng"
## [13] "end_lat"           "end_lng"
```

Now the first 10 columns of Q1_2020_new match the first 10 columns of the other three data sets in the same order. And the last four columns of Q1_2020_new do not match the last two columns of the other three data sets. Those columns will not be used in our analysis.

3.4 How can you verify that your data is clean and ready to analyze?

We have 1. checked the missing values and decided to keep them as the missing column values do not affect our analysis. 2. We have renamed the column names of Q2-2019 and Q1_2020 to match those of Q3_2019 and Q4_2019. 3. We have reordered the columns of Q1_2020 to match those of the other three data sets in the same order.

3.5 Have you documented your cleaning process so you can review and share those results?

Yes. See what were performed above.

3.6 (Deliverable) Documentation of any cleaning or manipulation of data

We have performed the following cleaning and manipulation of data. 1. Checked if there are missing values, and determined the missing values do not affect the desired data analysis. 2. renamed the column names of Q2-2019 and Q1_2020 to match those of Q3_2019 and Q4_2019 so as to enforce all data sets have matching column names. 3. reordered the columns of Q1_2020 to match those of the other three data sets in the same order. 4. We will check the data type of the matching columns in all four data files to make sure the matching columns of the four data sets have the same type of data in order to combine the four data sets into one data set.

4 ANALYZE

4.1 How should you organize your data to perform analysis on it?

We next join the four data sets into one data set for the ease of processing all data of the past year including the four quarters. As mentioned earlier, due to the discrepancy of columns between Q1_2020 and the other there data sets, only the first 10 columns will be selected.

```
Q2_2019_select <- Q2_2019_new %>%
  select(1:10)
Q3_2019_select <- Q3_2019 %>%
  select(1:10)
Q4_2019_select <- Q4_2019 %>%
  select(1:10)
Q1_2020_select <- Q1_2020_new %>%
  select(1:10)
```

The following code reveals that Q1_2020 use different notation to record `member` and `casual` customer than the other three data sets do.

```
unique(Q2_2019_select$usertype)
```

```
## [1] "Subscriber" "Customer"
```

```
unique(Q3_2019_select$usertype)
```

```
## [1] "Subscriber" "Customer"
```

```
unique(Q4_2019_select$usertype)
```

```
## [1] "Subscriber" "Customer"
```

```
unique(Q1_2020_select$usertype)
```

```
## [1] "member" "casual"
```

We know that in the three data sets of 2019, `Subscriber` matches `member` and `Customer` matches `casual`. So we decide to covert those values of `Subscriber` and `Customer` to `member` and `casual` to be consistent.

```
Q2_2019_select <- Q2_2019_select %>%
  mutate(usertype =case_when(
    usertype == "Subscriber" ~ "member",
    usertype == "Customer"   ~ "casual"
  ))
unique(Q2_2019_select$usertype)
```

```
## [1] "member" "casual"
```

```
Q3_2019_select <- Q3_2019_select %>%
  mutate(usertype =case_when(
    usertype == "Subscriber" ~ "member",
    usertype == "Customer"   ~ "casual"
  ))
unique(Q3_2019_select$usertype)
```

```
## [1] "member" "casual"
```

```r
Q4_2019_select <- Q4_2019_select %>%
  mutate(usertype =case_when(
    usertype == "Subscriber" ~ "member",
    usertype == "Customer"   ~ "casual"
  ))
unique(Q4_2019_select$usertype)
```

```
## [1] "member" "casual"
```

4.2 Has your data been properly formatted?

Some columns do not. Note that the trip_id column in Q2_2019, Q3_2019 and Q4_2019 needs to be converted into character type to conform to the type of trip_id column in Q1_2020, so as to perform the combining operation.

```r
Q2_2019_select <- Q2_2019_select %>% mutate(trip_id = as.character(trip_id))
Q3_2019_select <- Q3_2019_select %>% mutate(trip_id = as.character(trip_id))
Q4_2019_select <- Q4_2019_select %>% mutate(trip_id = as.character(trip_id))
```

Moreover, a closer look at the `tripduration` column of `Q1_20201 reveals it is of type`difftime. `The column needs to be converted to type`numeric'.

```r
Q1_2020_select <- Q1_2020_select %>%
  mutate(tripduration = as.numeric(tripduration))
```

4.3 What surprises did you discover in the data?

When trying to combining the four data sets, R reports error and prompts column types do not match. After performing above conversions, now it's ready to combine the four data sets into one giant data set for further analysis.

```r
bike_share <- Q2_2019_select %>%
              bind_rows(Q3_2019_select) %>%
              bind_rows(Q4_2019_select) %>%
              bind_rows(Q1_2020_select)
```

To the new dataset `bike_share`, we next add a column called `day_of_week` of the `start_time` to prepare for further analysis.

```r
bike_share <- bike_share %>%
  mutate ( day_of_week = weekdays(start_time))
```

4.4 What trends or relationships did you find in the data?

We shall computer on what week day there are the most number of rides. To this end, we need to compute the mode of a vector, we first define a user-defined function to this end, as R does not have a built-in function to find the mode.

```r
# Create the function.
get_mode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}  # getmode only finds the first element with the most frequency

get_mode_a <- function(v) {
   uniqv <- unique(v)
   tab <- tabulate(match(v, uniqv))
```

```
  uniqv[tab==max(tab)]
} # getmode_a finds all the elements with the most frequency when there are ties. Both functions works

# # Create the vector with numbers.
# v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)
#
# # Calculate the mode using the user function.
# result <- getmode(v)
# print(result)
#
# # Create the vector with characters.
# charv <- c("o","it","the","it","it","o","o")
#
# # Calculate the mode using the user function.
# result <- getmode(charv)
# print(result)
# result <- getmode1(charv)
# print(result)
```

We now are ready to perform analysis. We first compute some simple descriptive statistics.

```
summary_analysis <- bike_share %>%
  summarize ("avg_ride_length (min)"= mean(tripduration)/60,
             "max_ride_length (min)"=max(tripduration)/60,
             most_riders_day=get_mode_a(day_of_week))
summary_analysis
```

```
## # A tibble: 1 x 3
##   `avg_ride_length (min)` `max_ride_length (min)` most_riders_day
##                     <dbl>                   <dbl> <chr>
## 1                    26.8                 156450. Wednesday
```

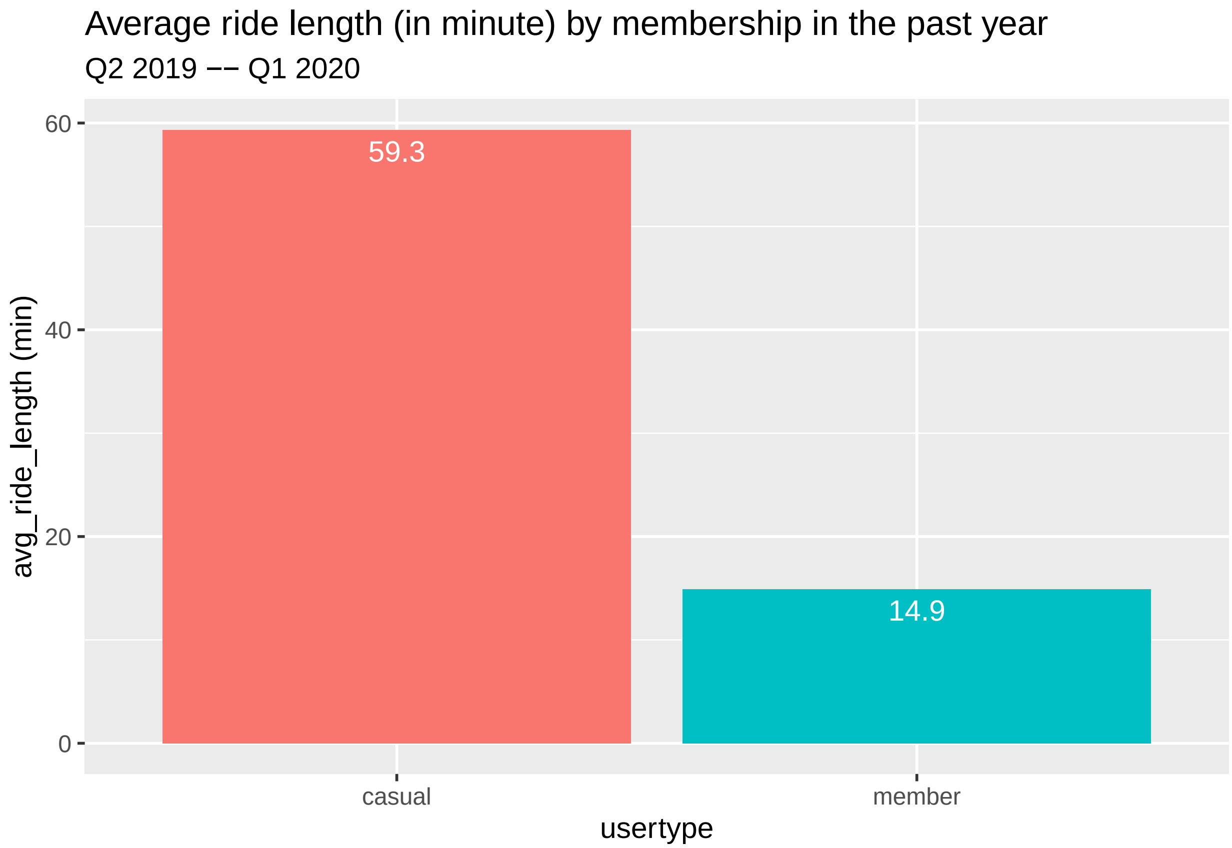
Observation 1: On average, the average ride length including both members and casual customers, is only 26 minutes. Wednesday has the most number of rides on average.

We next look at the difference between casual riders and subscribed members. We first look at their difference in terms of ride length.

```
avg_ride_by_group <- bike_share %>%
  group_by(usertype) %>%
  summarize(
  "avg_ride_length (min)" =mean(tripduration)/60)
avg_ride_by_group
```

```
## # A tibble: 2 x 2
##   usertype `avg_ride_length (min)`
## * <chr>                      <dbl>
## 1 casual                      59.3
## 2 member                      14.9
```

```
p <- avg_ride_by_group %>% ggplot(aes(x=usertype,y=`avg_ride_length (min)`,fill=usertype)) +
   geom_bar(stat="identity",show.legend = FALSE) +
  labs(title="Average ride length (in minute) by membership in the past year",
       subtitle = "Q2 2019 -- Q1 2020") +
  geom_text(aes(label=round(`avg_ride_length (min)`,1)),vjust=1.5,color="white",size=4)
print(p)
```



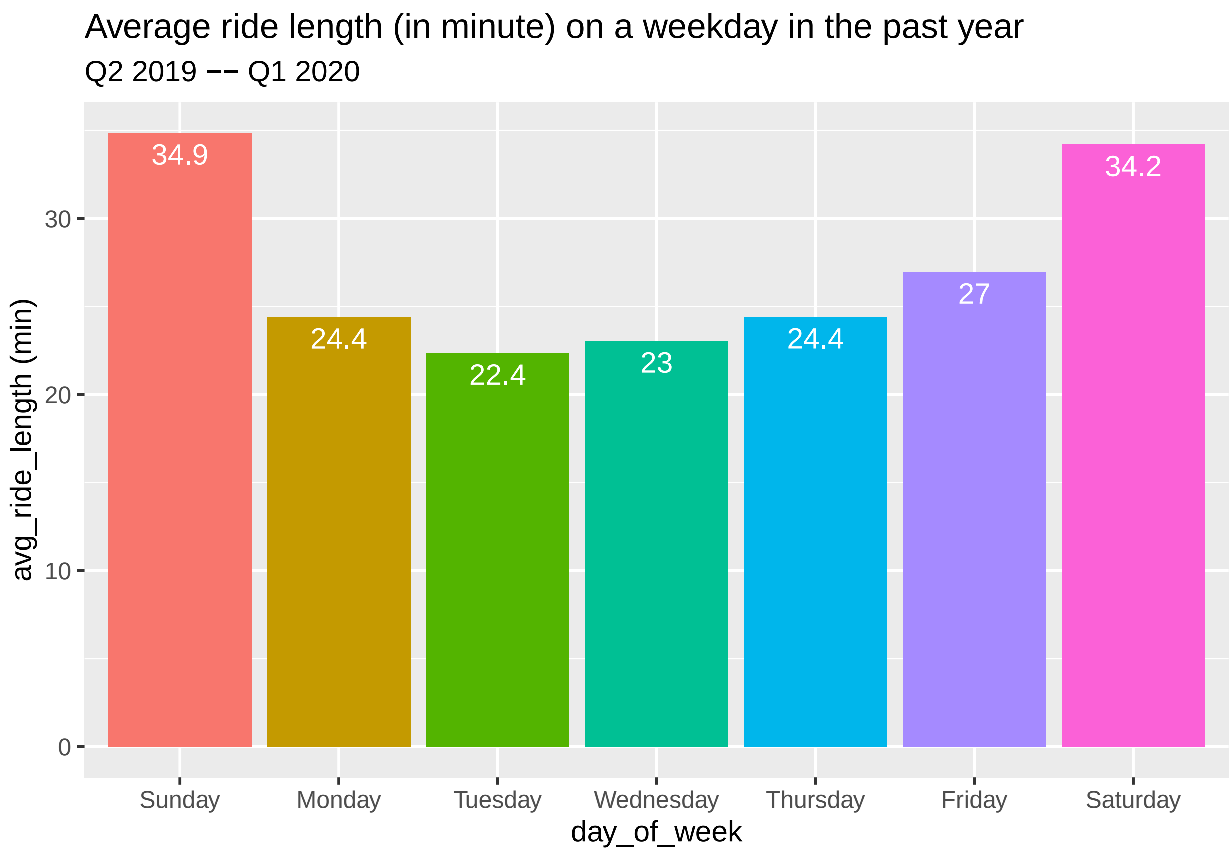
Observation 2: Subscription members on average have a much shorter ride length per ride compared to casual customers, who on average ride about 4 times as long as a subscription member does. This might attribute to that a casual customer mainly rent the bike for ad hoc purpose, e.g., recreational or an impromtu need, and typically spend longer time than a subscription member who may use the bike for a routine riding with a fixed route, such as commuting.

We next analyze the data by weekday.

```
stat_by_weekday <- bike_share %>%
  group_by(day_of_week) %>%
  summarize(
  "avg_ride_length (min)" =mean(tripduration)/60,
  number_of_rides =length(trip_id)
  ) %>%
  arrange(factor(day_of_week, level=c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Satu

stat_by_weekday
```

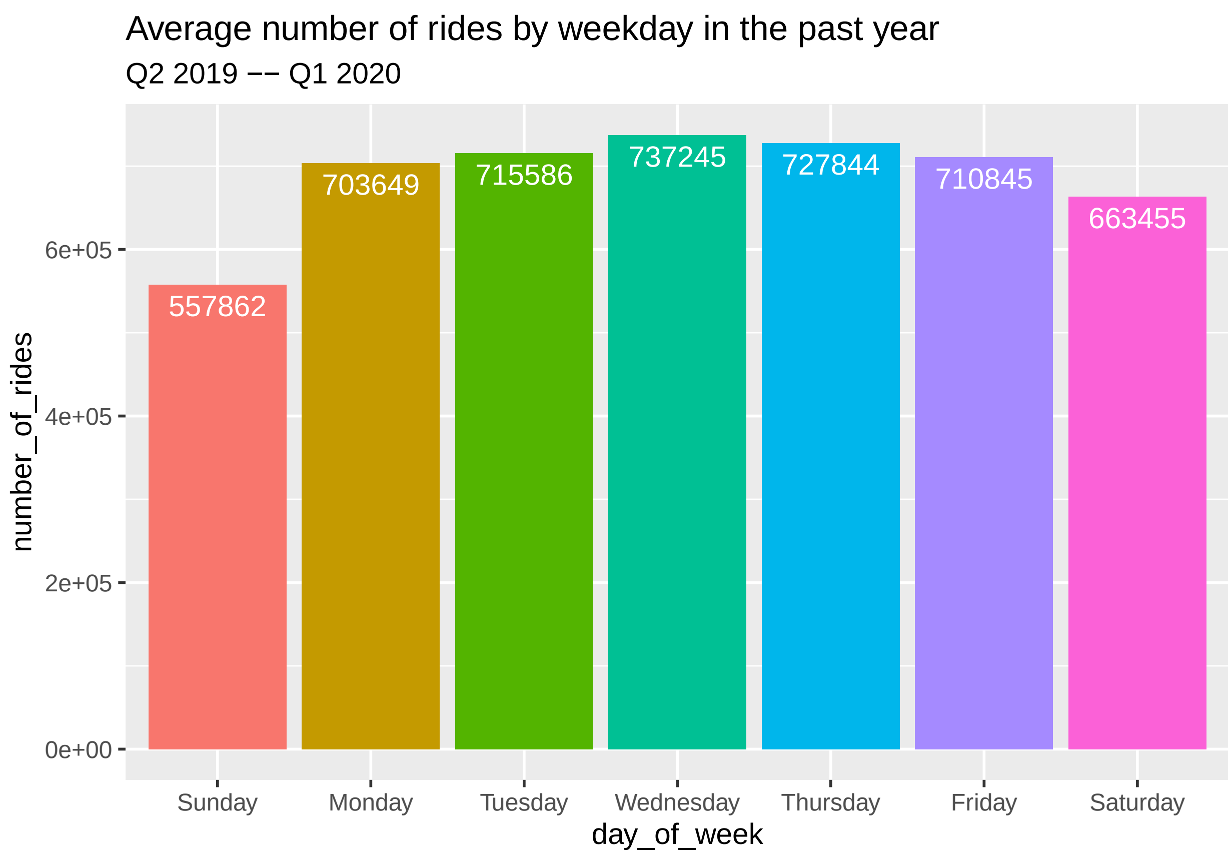
```
## # A tibble: 7 x 3
##   day_of_week `avg_ride_length (min)` number_of_rides
##   <chr>                         <dbl>           <int>
## 1 Sunday                         34.9          557862
## 2 Monday                         24.4          703649
## 3 Tuesday                        22.4          715586
## 4 Wednesday                      23.0          737245
## 5 Thursday                       24.4          727844
## 6 Friday                         27.0          710845
## 7 Saturday                       34.2          663455
```

```r
p <- stat_by_weekday %>%
  mutate(day_of_week =factor(day_of_week,level=c("Sunday","Monday","Tuesday","Wednesday","Thursday","Fri
  arrange(day_of_week) %>%
  ggplot(aes(day_of_week,`avg_ride_length (min)`,fill=day_of_week)) +
   geom_bar(stat="identity",
          show.legend = FALSE) +
  labs(title="Average ride length (in minute) on a weekday in the past year",
       subtitle = "Q2 2019 -- Q1 2020") +
  geom_text(aes(label=round(`avg_ride_length (min)`,1)),
          vjust=1.5,color="white",size=4)

print(p)
```



Observation 3 On average, bike users ride longer in weekend than on a week day. The average ride length is about 10 minutes longer in the weekend. This may be due to that there are more casual customers in weekend than in weekdays and a casual customer on average rides for about 60 minutes per ride.

```r
p <- stat_by_weekday %>%
  mutate(day_of_week =factor(day_of_week,level=c("Sunday","Monday","Tuesday","Wednesday","Thursday","Fri
  arrange(day_of_week) %>%
  ggplot(aes(x=day_of_week,y=number_of_rides,fill=day_of_week)) +
   geom_bar(stat="identity",show.legend = FALSE) +
  labs(title="Average number of rides by weekday in the past year",
       subtitle = "Q2 2019 -- Q1 2020") +
  geom_text(aes(label=number_of_rides),vjust=1.5,color="white",size=4)

print(p)
```



Observation 4 The number of rides in the weekend is significantly fewer than that on a weekday, with Sunday having the least number of rides of ~558k and Saturday having ~663k. The number of rides on a week day is always above 700k.
There are no significant difference for the number of rides on weekdays. However, Wednesday has the most number of rides on average in a week, peaks at ~734k.

We next analyze the data set by membership on different weekdays.

```
stat_by_weekday_usertype <- bike_share %>%
  group_by(day_of_week,usertype) %>%
  summarize(
  "avg_ride_length (min)" =mean(tripduration)/60,
  number_of_rides =length(trip_id)
    ) %>%
  arrange(factor(day_of_week, level=c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Satu
```

```
## `summarise()` has grouped output by 'day_of_week'. You can override using the `.groups` argument.
```

```
stat_by_weekday_usertype
```

```
## # A tibble: 14 x 4
## # Groups:   day_of_week [7]
##    day_of_week usertype `avg_ride_length (min)` number_of_rides
##    <chr>       <chr>                      <dbl>           <int>
##  1 Sunday      casual                      59.5          241714
##  2 Sunday      member                      16.0          316148
##  3 Monday      casual                      58.9          150892
##  4 Monday      member                      15.0          552757
##  5 Tuesday     casual                      59.0          127470
```

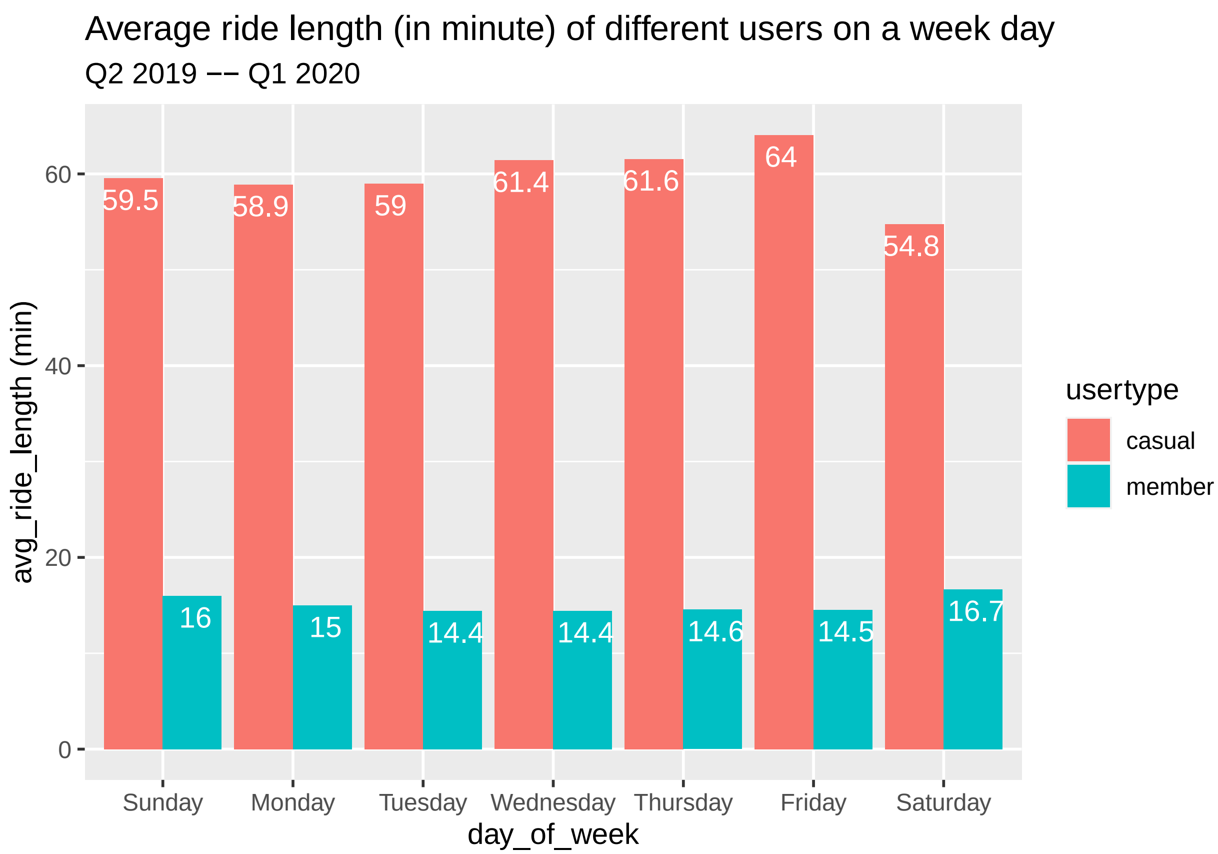
```
##  6 Tuesday     member                     14.4          588116
##  7 Wednesday   casual                     61.4          135090
##  8 Wednesday   member                     14.4          602155
##  9 Thursday    casual                     61.6          152616
## 10 Thursday    member                     14.6          575228
## 11 Friday      casual                     64.0          178285
## 12 Friday      member                     14.5          532560
## 13 Saturday    casual                     54.8          305387
## 14 Saturday    member                     16.7          358068
```

We also calculate the percentage of each customer type on a given day.

```
stat_by_weekday_usertype <- stat_by_weekday_usertype %>%
  ungroup() %>% group_by(day_of_week) %>%
  mutate(usertype_pct = number_of_rides/sum(number_of_rides)*100)
stat_by_weekday_usertype
```

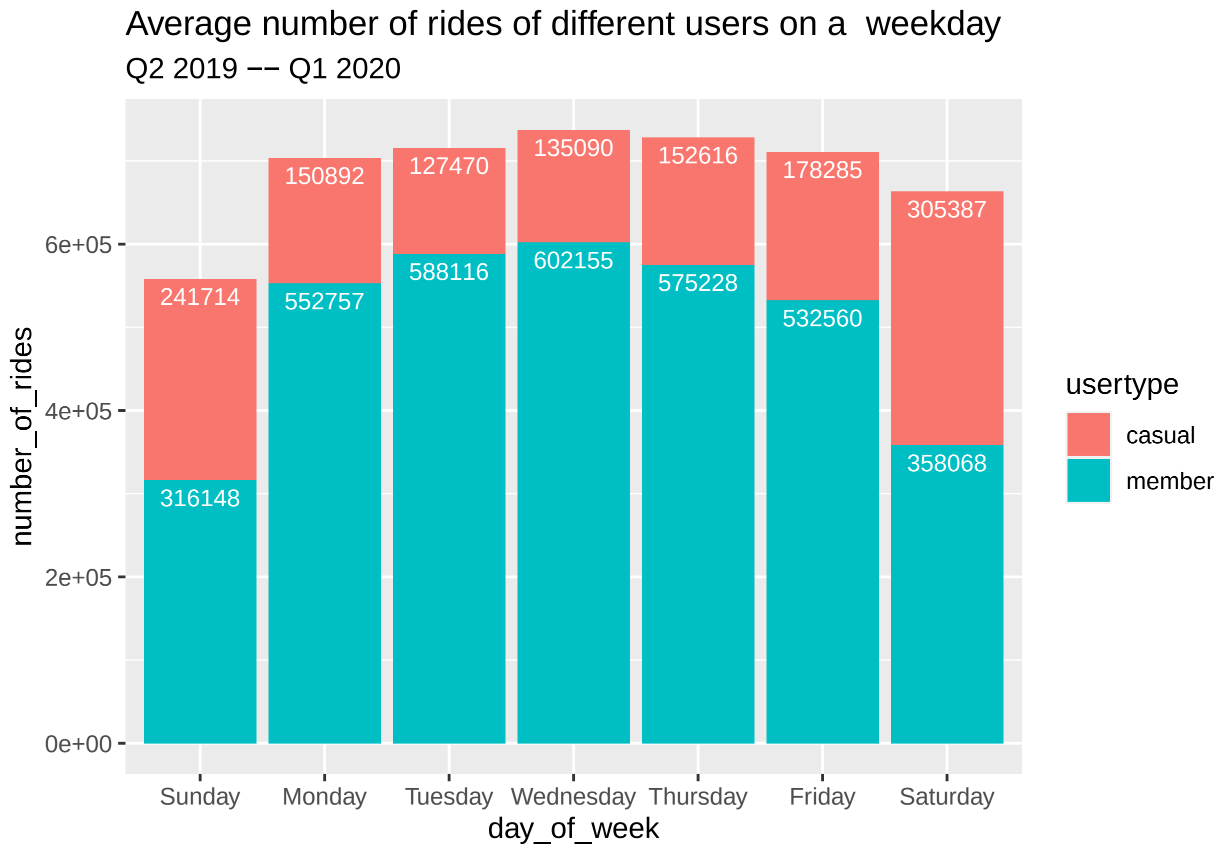
```
## # A tibble: 14 x 5
## # Groups:   day_of_week [7]
##    day_of_week usertype `avg_ride_length (min)` number_of_rides usertype_pct
##    <chr>       <chr>                      <dbl>           <int>        <dbl>
##  1 Sunday      casual                     59.5          241714         43.3
##  2 Sunday      member                     16.0          316148         56.7
##  3 Monday      casual                     58.9          150892         21.4
##  4 Monday      member                     15.0          552757         78.6
##  5 Tuesday     casual                     59.0          127470         17.8
##  6 Tuesday     member                     14.4          588116         82.2
##  7 Wednesday   casual                     61.4          135090         18.3
##  8 Wednesday   member                     14.4          602155         81.7
##  9 Thursday    casual                     61.6          152616         21.0
## 10 Thursday    member                     14.6          575228         79.0
## 11 Friday      casual                     64.0          178285         25.1
## 12 Friday      member                     14.5          532560         74.9
## 13 Saturday    casual                     54.8          305387         46.0
## 14 Saturday    member                     16.7          358068         54.0
```

```
p <- stat_by_weekday_usertype %>%
  mutate(day_of_week =factor(day_of_week,level=c("Sunday","Monday","Tuesday","Wednesday","Thursday","Fr
  arrange(day_of_week,usertype) %>%
  ggplot(aes(day_of_week,`avg_ride_length (min)`,fill=usertype)) +
   geom_bar(stat="identity",
           position = "dodge",
           show.legend = TRUE) +
  labs(title="Average ride length (in minute) of different users on a week day",
       subtitle = "Q2 2019 -- Q1 2020") +
  geom_text(aes(label=round(`avg_ride_length (min)`,1)),
           position = position_dodge(width = 1),
           vjust=1.5,color="white",size=4)

print(p)
```

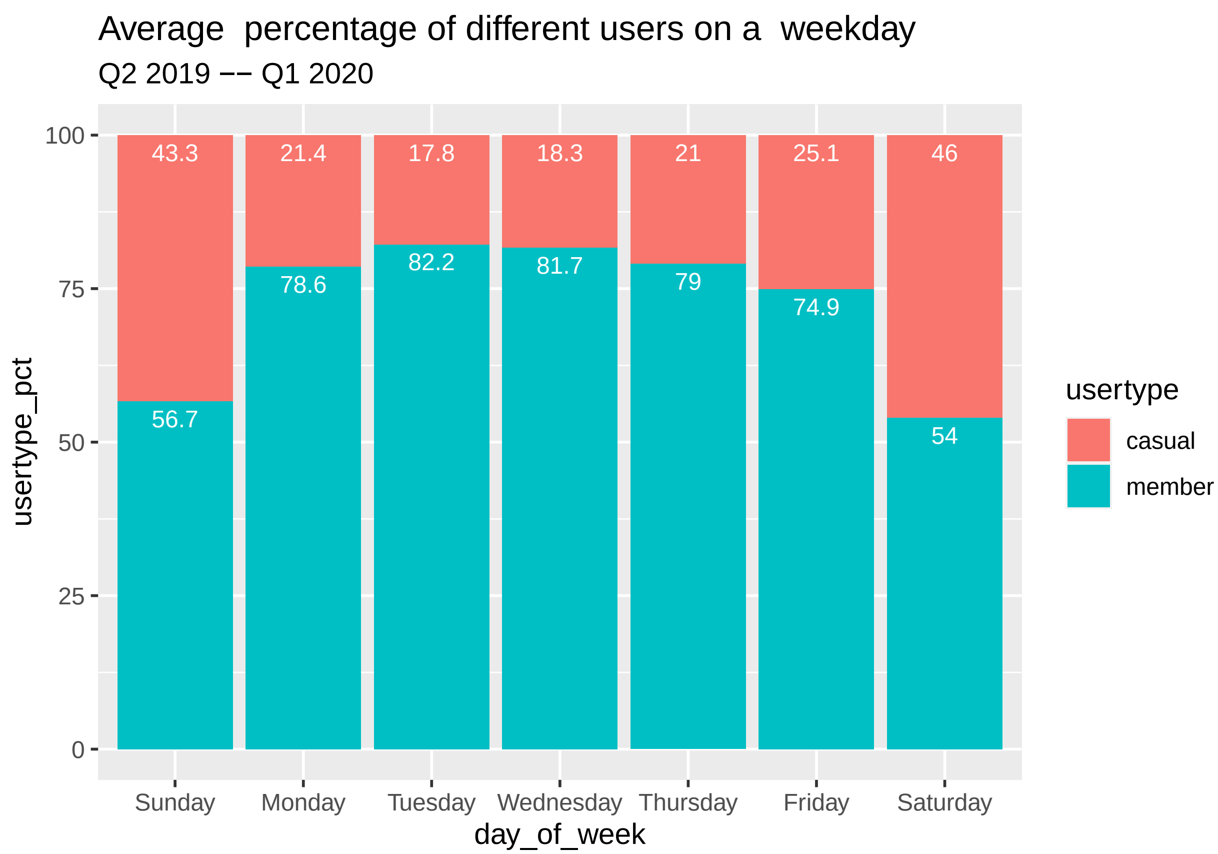


Observation 5 When diving into what type of customers rides longer on a week day, one can see that it appears subscription members ride about 1 to 1.7 minutes longer on average in weekend, while the casual customers do not ride significantly different than their overall average of 60 minutes per ride; except on Fridays, casual customers ride about 4 minutes longer, but about 5 minutes shorter on Saturday. Since we know the overall average ride length on weekend days are about 10 minutes longer than on a week day, this seems imply that there are more subscription members use the bike service on weekend, which is confirmed by the analysis below.

```
p <- stat_by_weekday_usertype %>%
  mutate(day_of_week =factor(day_of_week,level=c("Sunday","Monday","Tuesday","Wednesday","Thursday","Fr
  arrange(day_of_week,usertype) %>%
  ggplot(aes(day_of_week,number_of_rides,fill=usertype)) +
   geom_bar(stat="identity",
            #position = "dodge",
            show.legend = TRUE) +
  labs(title="Average number of rides of different users on a  weekday",
       subtitle = "Q2 2019 -- Q1 2020") +
  geom_text(aes(label=number_of_rides),
            position = "stack",
            vjust=1.5,color="white",size=3)

print(p)
```




```
p <- stat_by_weekday_usertype %>%
  mutate(day_of_week =factor(day_of_week,level=c("Sunday","Monday","Tuesday","Wednesday","Thursday","Fr
  arrange(day_of_week,usertype) %>%
  ggplot(aes(day_of_week,usertype_pct,fill=usertype)) +
   geom_bar(stat="identity",
            #position = "dodge",
            show.legend = TRUE) +
  labs(title="Average  percentage of different users on a  weekday",
       subtitle = "Q2 2019 -- Q1 2020") +
  geom_text(aes(label=round(usertype_pct,1)),
            position = "stack",
            vjust=1.5,color="white",size=3)

print(p)
```



Observation 6 The above two bar chart indicate a surprising fact: there are much more rides used by subscription members on any weekday than casual members. This difference is even larger on a weekday. Although on average more casual customers appear in weekend than those on a weekday, their numbers are still outnumbered by the number of subscription members. During the week, ~75-82% of rides are used by subscription members; while on a weekend day, there are still more than half (~55%) of rides are consumed by subscription members.

4.5 How will these insights help answer your business questions?

The findings directly answer the business questions raised in the beginig.

4.6 A summary of your analysis

The findings are very important to see the difference in usage of bikes by casual customers and subscription members. Subscription members contributes about 75-82% of rides during a weekday and ~55% of rides in a weekend day. Casual customers increase significantly on a week end day, consuming ~45% of rides compared to only 18-25% of rides during a week day.

In another word, on a week day, roughly there are four subscription members using the bike service for every casual customer; while on the weekend, roughly there are 55 subscription members versus 45 casual customers using the bike service among every 100 customers.

Another interesting finding is that a casual customer on average rides one hour per ride and a subscription member on average rides only 15 minutes per ride.

The third interesting finding is that on average subscription members ride about 2 minutes longer in the weekend than on a week day. The percentage of subscription members use the bike service drops from ~80% on a week day to ~55% on a weekend day, but always outnumbers the casual customers.

The fourth interesting finding is that the percentage of casual customers using the bike service doubles from ~20% on a weekday to ~ 45% on a weekend day.

5 SHARE

5.1 Were you able to answer the question of how annual members and casual riders use Cyclistic bikes differently?

Yes. See the above summary.

5.2 What story does your data tell?

See the summary analysis.

5.3 How do your findings relate to your original question?

They are directly related.

5.4 Who is your audience? What is the best way to communicate with them?

Managing team. Use a presentation with viz.

5.5 Can data visualization help you share your findings?

Yes, absolutely.

5.6 Is your presentation accessible to your audience?

Yes, Of course.

5.7 (deliverable) Supporting visualizations and key findings

See above.

6 ACT

6.1 What is your final conclusion based on your analysis?

See the summary analysis.

6.2 How could your team and business apply your insights?

6.3 What next steps would you or your stakeholders take based on your findings?

6.4 Is there additional data you could use to expand on your findings?

6.5 Your top three recommendations based on your analysis

1. Promotion should focus on increasing number of subscription members.
2. For a casual customer ride pass, the fee may consider to charge on a minimum 1 hour basis with half an hour incremental.
3. There can be a promotion to increase casual customers on a weekend day and the hourly charge may be slightly increased.