Bike Share Data Analysis

Abdullahi Faysal

2024-05-21

Case Study: Analyzing Bike-Share Data for Marketing Insights

The objective of this document is to analyze bike-share data to gain insights into customer behavior and preferences, specifically focusing on differentiating factors between casual riders and annual members.

Introduction

This document presents an analysis of bike-share data to understand customer behavior and preferences, focusing on casual riders and annual members.

The analysis will follow a structured approach, encompassing the key phases of data analysis: Ask, Prepare, Process, Analyze, Share, and Act.

Approach

Ask

- Formulate research questions to guide the analysis
- Define the business problem and objectives
- Establish metrics for evaluating success

Prepare

- Validate data quality and consistency
- Merge and clean datasets for analysis
- Ensure data integrity and reliability

Process

- Cleanse and transform data for analysis
- Document data processing steps for transparency

Analyze

- Identify trends and patterns in customer behavior
- Extract meaningful insights from the data
- Develop data-driven recommendations

Share

- Create visualizations to communicate findings effectively
- Craft a narrative around the data insights
- Present actionable recommendations to stakeholders

Act

- Provide strategic recommendations based on analysis -Address business challenges and opportunities
- Propose data-informed strategies for decision-making

1. Ask

Scenario The marketing team aims to enhance customer retention by converting casual riders into annual members. Understanding the usage disparities between these two customer segments is crucial for devising targeted marketing campaigns.

Stakeholders:

- Marketing Director
- Executive Team

Objective

- The goal is to analyze how casual riders and annual members utilize the bike-share program differently based on available data.

Deliverables:

- Insights on usage patterns of casual riders and annual members
- Visualizations supporting key findings
- Recommendations for converting casual riders into annual members

2. Prepare

Data Source The data for this analysis is sourced from the Divvy bike-share program, specifically the Divvy 2019 Q2, Divvy 2019 Q3, Divvy 2019 Q4, and Divvy 2020 Q1 datasets. These datasets contain trip details and customer information that will be used to analyze the behavior of casual riders and annual members.

```
library(tidyverse)
```

Load Libraries

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)
library(lubridate)
library(dplyr)
library(readr)
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(tidyr)
```

```
q4_2019 <- read_csv("C:\\Users\\Abdullahi77\\OneDrive\\Desktop\\Data analyst Project 1\\Divvy_Trips_2019
```

Load Divvy Bike-Share Data.

```
## Rows: 704054 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): from_station_name, to_station_name, usertype, gender
## dbl  (5): trip_id, bikeid, from_station_id, to_station_id, birthyear
## num  (1): tripduration
## dttm (2): start_time, end_time
##
## 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.
```

```
q3_2019 <- read_csv("C:\\Users\\Abdullahi77\\OneDrive\\Desktop\\Data analyst Project 1\\Divvy_Trips_2019
```

```
## Rows: 1640718 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): from_station_name, to_station_name, usertype, gender
## dbl  (5): trip_id, bikeid, from_station_id, to_station_id, birthyear
## num  (1): tripduration
## dttm (2): start_time, end_time
##
## 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.
```

```
q2_2019 <- read_csv("C:\\Users\\Abdullahi77\\OneDrive\\Desktop\\Data analyst Project 1\\Divvy_Trips_2019
```

```
## Rows: 1108163 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): 03 - Rental Start Station Name, 02 - Rental End Station Name, User...
## dbl  (5): 01 - Rental Details Rental ID, 01 - Rental Details Bike ID, 03 - R...
## num  (1): 01 - Rental Details Duration In Seconds Uncapped
## dttm (2): 01 - Rental Details Local Start Time, 01 - Rental Details Local En...
##
## 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.
```

```
q1_2020 <- read_csv("C:\\Users\\Abdullahi77\\OneDrive\\Desktop\\Data analyst Project 1\\Divvy_Trips_2020
```

```
## Rows: 426887 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## dttm (2): started_at, ended_at
##
## 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.
```

2. Process

```
colnames(q2_2019)
```

Display columns names of each dataframe.

```
##  [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"
```

```
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"
```

```
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"
```

```
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"
```

Renaming columns to ensure uniformity.

```
(q4_2019 <- rename(q4_2019
                  ,ride_id = trip_id
```

```
                   ,rideable_type = bikeid
                   ,started_at = start_time
                   ,ended_at = end_time
                   ,start_station_name = from_station_name
                   ,start_station_id = from_station_id
                   ,end_station_name = to_station_name
                   ,end_station_id = to_station_id
                   ,member_casual = usertype))
```

Certainly! Ensuring consistent column names across dataframes is crucial for effective data analysis. Let's proceed with the cleaning process by renaming the columns using the rename() function.

```
## # A tibble: 704,054 x 12
##     ride_id started_at          ended_at            rideable_type tripduration
##       <dbl> <dttm>              <dttm>                      <dbl>        <dbl>
##  1 25223640 2019-10-01 00:01:39 2019-10-01 00:17:20          2215          940
##  2 25223641 2019-10-01 00:02:16 2019-10-01 00:06:34          6328          258
##  3 25223642 2019-10-01 00:04:32 2019-10-01 00:18:43          3003          850
##  4 25223643 2019-10-01 00:04:32 2019-10-01 00:43:43          3275         2350
##  5 25223644 2019-10-01 00:04:34 2019-10-01 00:35:42          5294         1867
##  6 25223645 2019-10-01 00:04:38 2019-10-01 00:10:51          1891          373
##  7 25223646 2019-10-01 00:04:52 2019-10-01 00:22:45          1061         1072
##  8 25223647 2019-10-01 00:04:57 2019-10-01 00:29:16          1274         1458
##  9 25223648 2019-10-01 00:05:20 2019-10-01 00:29:18          6011         1437
## 10 25223649 2019-10-01 00:05:20 2019-10-01 02:23:46          2957         8306
## # i 704,044 more rows
## # i 7 more variables: start_station_id <dbl>, start_station_name <chr>,
## #   end_station_id <dbl>, end_station_name <chr>, member_casual <chr>,
## #   gender <chr>, birthyear <dbl>
```

```
(q3_2019 <- rename(q3_2019
                   ,ride_id = trip_id
                   ,rideable_type = bikeid
                   ,started_at = start_time
                   ,ended_at = end_time
                   ,start_station_name = from_station_name
                   ,start_station_id = from_station_id
                   ,end_station_name = to_station_name
                   ,end_station_id = to_station_id
                   ,member_casual = usertype))
```

```
## # A tibble: 1,640,718 x 12
##     ride_id started_at          ended_at            rideable_type tripduration
##       <dbl> <dttm>              <dttm>                      <dbl>        <dbl>
##  1 23479388 2019-07-01 00:00:27 2019-07-01 00:20:41          3591         1214
##  2 23479389 2019-07-01 00:01:16 2019-07-01 00:18:44          5353         1048
##  3 23479390 2019-07-01 00:01:48 2019-07-01 00:27:42          6180         1554
##  4 23479391 2019-07-01 00:02:07 2019-07-01 00:27:10          5540         1503
##  5 23479392 2019-07-01 00:02:13 2019-07-01 00:22:26          6014         1213
##  6 23479393 2019-07-01 00:02:21 2019-07-01 00:07:31          4941          310
##  7 23479394 2019-07-01 00:02:24 2019-07-01 00:23:12          3770         1248
##  8 23479395 2019-07-01 00:02:26 2019-07-01 00:28:16          5442         1550
```

```
##  9 23479396 2019-07-01 00:02:34 2019-07-01 00:28:57          2957         1583
## 10 23479397 2019-07-01 00:02:45 2019-07-01 00:29:14          6091         1589
## # i 1,640,708 more rows
## # i 7 more variables: start_station_id <dbl>, start_station_name <chr>,
## #   end_station_id <dbl>, end_station_name <chr>, member_casual <chr>,
## #   gender <chr>, birthyear <dbl>
```

```
(q2_2019 <- rename(q2_2019
                  ,ride_id = "01 - Rental Details Rental ID"
                  ,rideable_type = "01 - Rental Details Bike ID"
                  ,started_at = "01 - Rental Details Local Start Time"
                  ,ended_at = "01 - Rental Details Local End Time"
                  ,start_station_name = "03 - Rental Start Station Name"
                  ,start_station_id = "03 - Rental Start Station ID"
                  ,end_station_name = "02 - Rental End Station Name"
                  ,end_station_id = "02 - Rental End Station ID"
                  ,member_casual = "User Type"))
```

```
## # A tibble: 1,108,163 x 12
##     ride_id started_at          ended_at            rideable_type
##       <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
##  7 22178535 2019-04-01 00:26:33 2019-04-01 00:35:41          6418
##  8 22178536 2019-04-01 00:29:48 2019-04-01 00:36:11          4513
##  9 22178537 2019-04-01 00:32:07 2019-04-01 01:07:44          3280
## 10 22178538 2019-04-01 00:32:19 2019-04-01 01:07:39          5534
## # i 1,108,153 more rows
## # i 8 more variables: `01 - Rental Details Duration In Seconds Uncapped` <dbl>,
## #   start_station_id <dbl>, start_station_name <chr>, end_station_id <dbl>,
## #   end_station_name <chr>, member_casual <chr>, `Member Gender` <chr>,
## #   `05 - Member Details Member Birthday Year` <dbl>
```

```
str(q2_2019)
```

Using the str() function, we can notice the datatype for ride_id column and the rideable_type column in 2019 quarters is different than that of the 2020 quarter.

```
## spc_tbl_ [1,108,163 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id                                          : num [1:1108163] 22178529 22178530 22178531 2217
##  $ started_at                                       : POSIXct[1:1108163], format: "2019-04-01 00:02:2
##  $ ended_at                                         : POSIXct[1:1108163], format: "2019-04-01 00:09:4
##  $ rideable_type                                    : num [1:1108163] 6251 6226 5649 4151 3270 ...
##  $ 01 - Rental Details Duration In Seconds Uncapped: num [1:1108163] 446 1048 252 357 1007 ...
##  $ start_station_id                                 : num [1:1108163] 81 317 283 26 202 420 503 260 2
##  $ start_station_name                               : chr [1:1108163] "Daley Center Plaza" "Wood St &
```

```
##  $ end_station_id                                    : num [1:1108163] 56 59 174 133 129 426 500 499 2
##  $ end_station_name                                  : chr [1:1108163] "Desplaines St & Kinzie St" "Wal
##  $ member_casual                                     : chr [1:1108163] "Subscriber" "Subscriber" "Subs
##  $ Member Gender                                     : chr [1:1108163] "Male" "Female" "Male" "Male" .
##  $ 05 - Member Details Member Birthday Year          : num [1:1108163] 1975 1984 1990 1993 1992 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `01 - Rental Details Rental ID` = col_double(),
##   ..   `01 - Rental Details Local Start Time` = col_datetime(format = ""),
##   ..   `01 - Rental Details Local End Time` = col_datetime(format = ""),
##   ..   `01 - Rental Details Bike ID` = col_double(),
##   ..   `01 - Rental Details Duration In Seconds Uncapped` = col_number(),
##   ..   `03 - Rental Start Station ID` = col_double(),
##   ..   `03 - Rental Start Station Name` = col_character(),
##   ..   `02 - Rental End Station ID` = col_double(),
##   ..   `02 - Rental End Station Name` = col_character(),
##   ..   `User Type` = col_character(),
##   ..   `Member Gender` = col_character(),
##   ..   `05 - Member Details Member Birthday Year` = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
str(q3_2019)
```

```
## spc_tbl_ [1,640,718 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : num [1:1640718] 23479388 23479389 23479390 23479391 23479392 ...
##  $ started_at        : POSIXct[1:1640718], format: "2019-07-01 00:00:27" "2019-07-01 00:01:16" ...
##  $ ended_at          : POSIXct[1:1640718], format: "2019-07-01 00:20:41" "2019-07-01 00:18:44" ...
##  $ rideable_type     : num [1:1640718] 3591 5353 6180 5540 6014 ...
##  $ tripduration      : num [1:1640718] 1214 1048 1554 1503 1213 ...
##  $ start_station_id  : num [1:1640718] 117 381 313 313 168 300 168 313 43 43 ...
##  $ start_station_name: chr [1:1640718] "Wilton Ave & Belmont Ave" "Western Ave & Monroe St" "Lakevie
##  $ end_station_id    : num [1:1640718] 497 203 144 144 62 232 62 144 195 195 ...
##  $ end_station_name  : chr [1:1640718] "Kimball Ave & Belmont Ave" "Western Ave & 21st St" "Larrabee
##  $ member_casual     : chr [1:1640718] "Subscriber" "Customer" "Customer" "Customer" ...
##  $ gender            : chr [1:1640718] "Male" NA NA NA ...
##  $ birthyear         : num [1:1640718] 1992 NA NA NA NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   trip_id = col_double(),
##   ..   start_time = col_datetime(format = ""),
##   ..   end_time = col_datetime(format = ""),
##   ..   bikeid = col_double(),
##   ..   tripduration = col_number(),
##   ..   from_station_id = col_double(),
##   ..   from_station_name = col_character(),
##   ..   to_station_id = col_double(),
##   ..   to_station_name = col_character(),
##   ..   usertype = col_character(),
##   ..   gender = col_character(),
##   ..   birthyear = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
str(q4_2019)
```

```
## spc_tbl_ [704,054 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : num [1:704054] 25223640 25223641 25223642 25223643 25223644 ...
##  $ started_at        : POSIXct[1:704054], format: "2019-10-01 00:01:39" "2019-10-01 00:02:16" ...
##  $ ended_at          : POSIXct[1:704054], format: "2019-10-01 00:17:20" "2019-10-01 00:06:34" ...
##  $ rideable_type     : num [1:704054] 2215 6328 3003 3275 5294 ...
##  $ tripduration      : num [1:704054] 940 258 850 2350 1867 ...
##  $ start_station_id  : num [1:704054] 20 19 84 313 210 156 84 156 156 336 ...
##  $ start_station_name: chr [1:704054] "Sheffield Ave & Kingsbury St" "Throop (Loomis) St & Taylor St"
##  $ end_station_id    : num [1:704054] 309 241 199 290 382 226 142 463 463 336 ...
##  $ end_station_name  : chr [1:704054] "Leavitt St & Armitage Ave" "Morgan St & Polk St" "Wabash Ave &
##  $ member_casual     : chr [1:704054] "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
##  $ gender            : chr [1:704054] "Male" "Male" "Female" "Male" ...
##  $ birthyear         : num [1:704054] 1987 1998 1991 1990 1987 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   trip_id = col_double(),
##   ..   start_time = col_datetime(format = ""),
##   ..   end_time = col_datetime(format = ""),
##   ..   bikeid = col_double(),
##   ..   tripduration = col_number(),
##   ..   from_station_id = col_double(),
##   ..   from_station_name = col_character(),
##   ..   to_station_id = col_double(),
##   ..   to_station_name = col_character(),
##   ..   usertype = col_character(),
##   ..   gender = col_character(),
##   ..   birthyear = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
str(q1_2020)
```

```
## spc_tbl_ [426,887 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:426887] "EACB19130B0CDA4A" "8FED874C809DC021" "789F3C21E472CA96" "C9A38
##  $ rideable_type     : chr [1:426887] "docked_bike" "docked_bike" "docked_bike" "docked_bike" ...
##  $ started_at        : POSIXct[1:426887], format: "2020-01-21 20:06:59" "2020-01-30 14:22:39" ...
##  $ ended_at          : POSIXct[1:426887], format: "2020-01-21 20:14:30" "2020-01-30 14:26:22" ...
##  $ start_station_name: chr [1:426887] "Western Ave & Leland Ave" "Clark St & Montrose Ave" "Broadway
##  $ start_station_id  : num [1:426887] 239 234 296 51 66 212 96 96 212 38 ...
##  $ end_station_name  : chr [1:426887] "Clark St & Leland Ave" "Southport Ave & Irving Park Rd" "Wilto
##  $ end_station_id    : num [1:426887] 326 318 117 24 212 96 212 212 96 100 ...
##  $ start_lat         : num [1:426887] 42 42 41.9 41.9 41.9 ...
##  $ start_lng         : num [1:426887] -87.7 -87.7 -87.6 -87.6 -87.6 ...
##  $ end_lat           : num [1:426887] 42 42 41.9 41.9 41.9 ...
##  $ end_lng           : num [1:426887] -87.7 -87.7 -87.7 -87.6 -87.6 ...
##  $ member_casual     : chr [1:426887] "member" "member" "member" "member" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   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_double(),
##   ..   end_station_name = col_character(),
##   ..   end_station_id = col_double(),
##   ..   start_lat = col_double(),
##   ..   start_lng = col_double(),
##   ..   end_lat = col_double(),
##   ..   end_lng = col_double(),
##   ..   member_casual = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Checking and converting column datatypes

```
str(q2_2019$ride_id)
```

column datatype of ride_id

```
##  num [1:1108163] 22178529 22178530 22178531 22178532 22178533 ...
```

```
str(q3_2019$ride_id)
```

```
##  num [1:1640718] 23479388 23479389 23479390 23479391 23479392 ...
```

```
str(q4_2019$ride_id)
```

```
##  num [1:704054] 25223640 25223641 25223642 25223643 25223644 ...
```

```
str(q1_2020$ride_id)
```

```
##  chr [1:426887] "EACB19130B0CDA4A" "8FED874C809DC021" "789F3C21E472CA96" ...
```

```
str(q2_2019$rideable_type)
```

column datatype of rideable_type

```
##  num [1:1108163] 6251 6226 5649 4151 3270 ...
```

```
str(q3_2019$rideable_type)
```

```
##  num [1:1640718] 3591 5353 6180 5540 6014 ...
```

```r
str(q4_2019$rideable_type)
```

```
##  num [1:704054] 2215 6328 3003 3275 5294 ...
```

```r
str(q1_2020$rideable_type)
```

```
##  chr [1:426887] "docked_bike" "docked_bike" "docked_bike" "docked_bike" ...
```

```r
q4_2019 <- mutate(q4_2019, ride_id = as.character(ride_id)
                  ,rideable_type = as.character(rideable_type))

q3_2019 <- mutate(q3_2019, ride_id = as.character(ride_id)
                  ,rideable_type = as.character(rideable_type))

q2_2019 <- mutate(q2_2019, ride_id = as.character(ride_id)
                  ,rideable_type = as.character(rideable_type))
```

To change datatypes of columns to character

```r
all_divvy_trips <- bind_rows(q2_2019,q3_2019,q4_2019,q1_2020)
```

Combine all the datasets into one single dataframe

```r
all_divvy_trips <- all_divvy_trips %>%
  select(-c(start_lat,start_lng,end_lat,end_lng,birthyear,gender,
            "01 - Rental Details Duration In Seconds Uncapped",
            "05 - Member Details Member Birthday Year",
            "Member Gender","tripduration"))
```

Deleting unwanted columns

```r
unique(all_divvy_trips$member_casual)
```

Obtaining the unique values in the 'member_casual' column of the 'all_trips' dataset.

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

```r
all_divvy_trips <- all_divvy_trips %>%
  mutate(member_casual = recode(member_casual,"Subscriber" = "member", "Customer" = "casual"))

##### Display the resulting unique member types

cat("Unique member types after recoding:", "\n")
```

Recode values in the 'member_casual' column to their appropriate alternates

```
## Unique member types after recoding:
```

```r
cat(unique(all_divvy_trips$member_casual), "\n")
```

```
## member casual
```

```r
##### add a new column that is the date format of 'started_at' column
all_divvy_trips$date <- as.Date(all_divvy_trips$started_at)

##### extract month (01 for Jan, 02 for Feb, 03 for March,....)
all_divvy_trips$month <- format(as.Date(all_divvy_trips$date),"%m")

##### extract day (01,02,03,04,...)
all_divvy_trips$day <- format(as.Date(all_divvy_trips$date),"%d")

##### extract year (2019, 2020)
all_divvy_trips$year <- format(as.Date(all_divvy_trips$date),"%Y")

# extract day_of_week (Sunday, Monday, Tuesday,...)
all_divvy_trips$day_of_week <- format(as.Date(all_divvy_trips$date),"%A")
```

Insert new month, date, year and day columns

```r
all_divvy_trips <- all_divvy_trips %>%
  mutate(ride_length = difftime(ended_at, started_at)) %>%
  mutate(ride_length = as.numeric(as.character(.$ride_length)))
```

Added 'ride_length' column with ride duration in seconds, calculated by subtracting start time from end time. Converted 'ride_length' data type to numeric.

```r
all_divvy_trips_v2 <- all_divvy_trips[!(all_divvy_trips$start_station_name == "HQ QR" | all_divvy_trips$
```

Filter out rows with negative ride_length values or invalid start_station_name entries.

3. Analyze

```
# Mean
mean_result <- all_divvy_trips_v2 %>%
  group_by(member_casual) %>%
  summarise(mean_ride_length = mean(ride_length))
print(mean_result)
```

Analyzing the ride_length variable.

```
## # A tibble: 2 x 2
##   member_casual mean_ride_length
##   <chr>                    <dbl>
## 1 casual                   3553.
## 2 member                    850.
```

```
# Median
median_result <- all_divvy_trips_v2 %>%
  group_by(member_casual) %>%
  summarise(median_ride_length = median(ride_length))
print(median_result)
```

```
## # A tibble: 2 x 2
##   member_casual median_ride_length
##   <chr>                       <dbl>
## 1 casual                       1546
## 2 member                        589
```

```
# Maximum
max_result <- all_divvy_trips_v2 %>%
  group_by(member_casual) %>%
  summarise(max_ride_length = max(ride_length))
print(max_result)
```

```
## # A tibble: 2 x 2
##   member_casual max_ride_length
##   <chr>                   <dbl>
## 1 casual                9387024
## 2 member                9056634
```

```
# Minimum
min_result <- all_divvy_trips_v2 %>%
  group_by(member_casual) %>%
  summarise(min_ride_length = min(ride_length))
print(min_result)
```

```
## # A tibble: 2 x 2
##   member_casual min_ride_length
##   <chr>                   <dbl>
## 1 casual                      2
## 2 member                      1
```

```
mean_result <- all_divvy_trips_v2 %>%
  group_by(member_casual, day_of_week) %>%
  summarise(mean_ride_length = mean(ride_length)) %>%
  ungroup()
```

Calculate the mean ride length by membership type and day of the week

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

```
print(mean_result)
```

```
## # A tibble: 14 x 3
##    member_casual day_of_week mean_ride_length
##    <chr>         <chr>                  <dbl>
##  1 casual        Friday                 3774.
##  2 casual        Monday                 3372.
##  3 casual        Saturday               3332.
##  4 casual        Sunday                 3581.
##  5 casual        Thursday               3683.
##  6 casual        Tuesday                3596.
##  7 casual        Wednesday              3719.
##  8 member        Friday                  825.
##  9 member        Monday                  843.
## 10 member        Saturday                969.
## 11 member        Sunday                  920.
## 12 member        Thursday                824.
## 13 member        Tuesday                 826.
## 14 member        Wednesday               824.
```

```
#ordered function
all_divvy_trips_v2$day_of_week <- ordered(all_divvy_trips_v2$day_of_week,levels=c("Sunday","Monday","Tu
```

Create an ordered factor for day_of_week (Sunday to Saturday)

```
result <- all_divvy_trips_v2 %>%
  mutate(weekday = lubridate::wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%
  summarize(number_of_rides = n(), average_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday)
```

Analyzing ridership data by member type and week by calculating number of rides and average ride duration

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

```
print(result)
```

```
## # A tibble: 14 x 4
## # Groups:   member_casual [2]
##    member_casual weekday number_of_rides average_duration
##    <chr>         <ord>             <int>            <dbl>
##  1 casual        Sun              181293            3581.
##  2 casual        Mon              103296            3372.
##  3 casual        Tue               90510            3596.
##  4 casual        Wed               92457            3719.
##  5 casual        Thu              102679            3683.
##  6 casual        Fri              122404            3774.
##  7 casual        Sat              209543            3332.
##  8 member        Sun              267965             920.
##  9 member        Mon              472196             843.
## 10 member        Tue              508445             826.
## 11 member        Wed              500329             824.
## 12 member        Thu              484177             824.
## 13 member        Fri              452790             825.
## 14 member        Sat              287958             969.
```

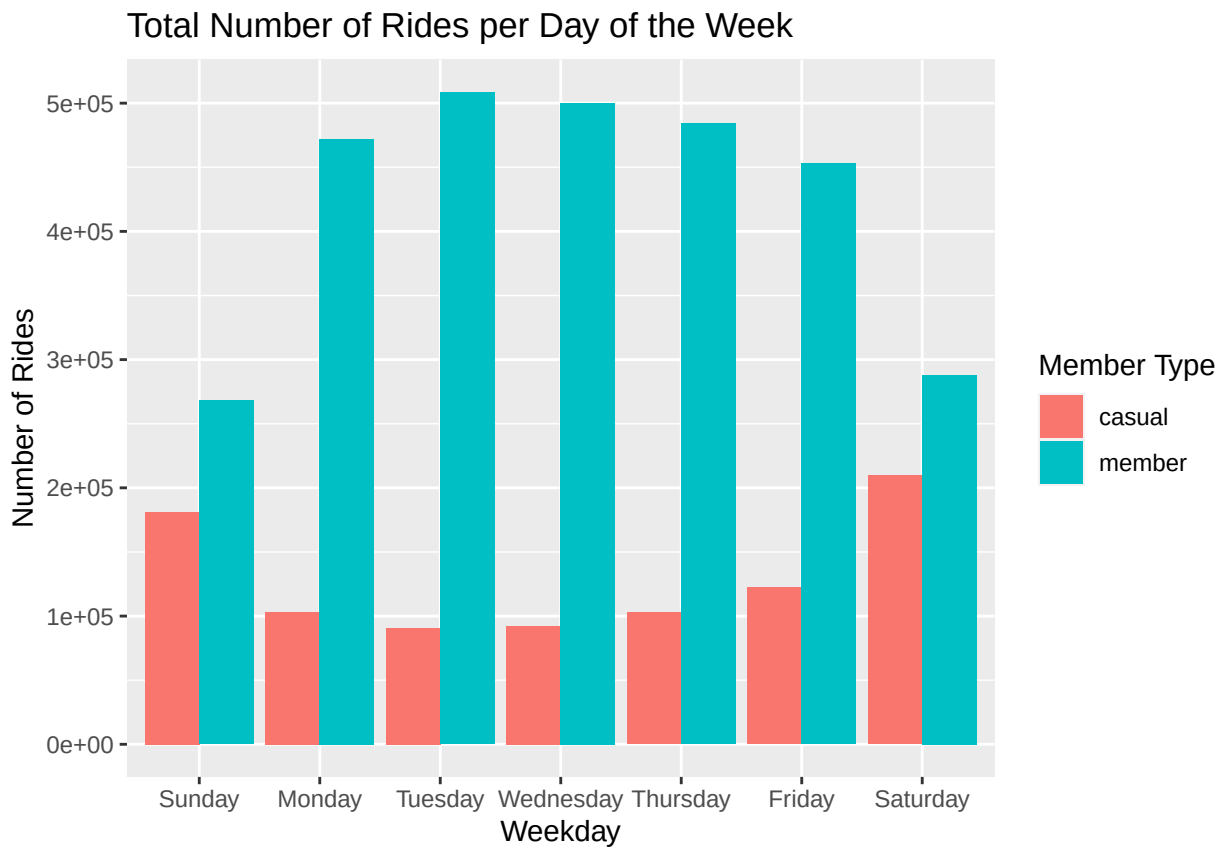
4. Share

```
result <- all_divvy_trips_v2 %>%
  mutate(weekday = weekdays(started_at, abbreviate = FALSE)) %>%
  mutate(weekday = factor(weekday, levels = c("Sunday", "Monday", "Tuesday",   "Wednesday", "Thursday",
  group_by(member_casual, weekday) %>%
  summarize(number_of_rides = n(), average_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday)
```

Calculate the total number of rides per day of the week and create a bar plot to visualize the results

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

```
ggplot(result, aes(x = weekday, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge")+
  labs(title = "Total Number of Rides per Day of the Week",
       x = "Weekday",
       y = "Number of Rides",
       fill = "Member Type")
```



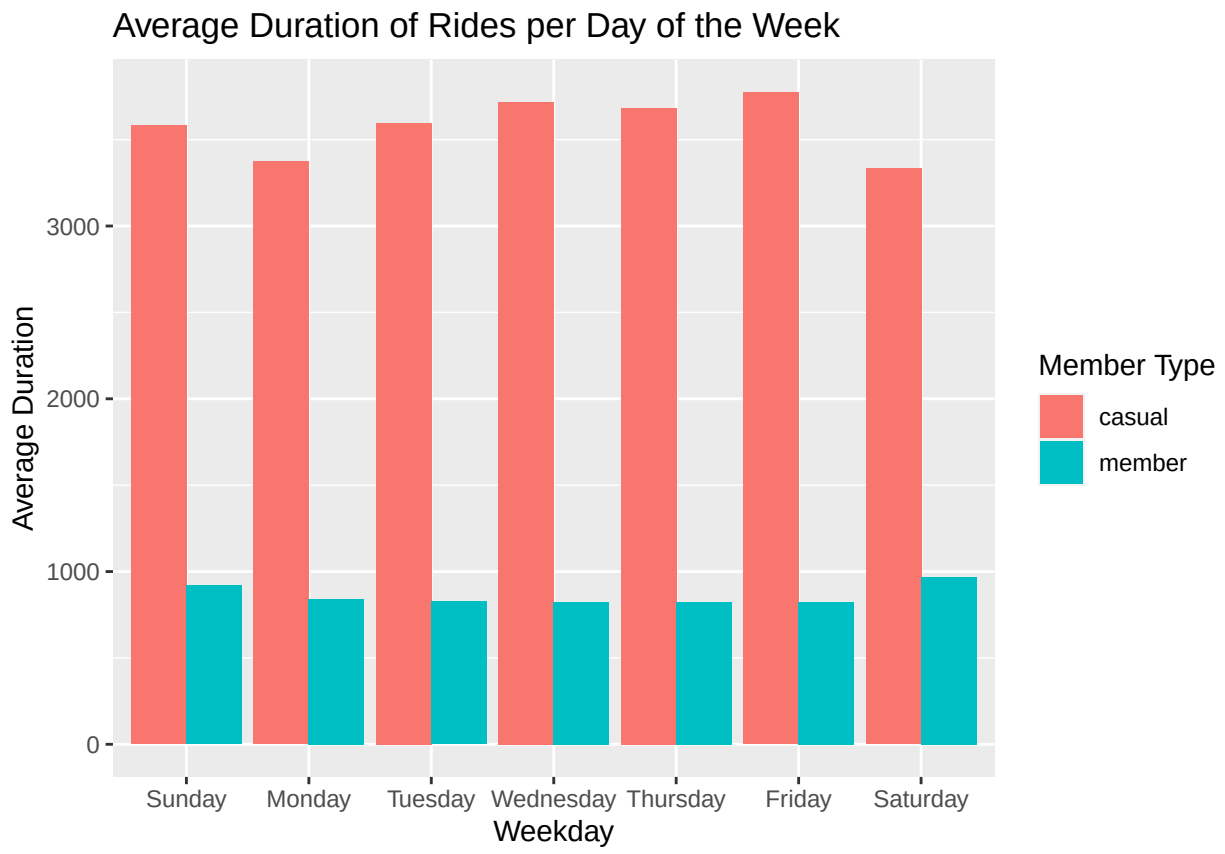

```
# Define the desired order of weekdays

weekday_order <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")

all_divvy_trips_v2 %>%
  mutate(weekday_num = wday(started_at)) %>%
  mutate(weekday = factor(weekday_num, levels = 1:7, labels = weekday_order)) %>%
  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") +
  labs(title = "Average Duration of Rides per Day of the Week",
       x = "Weekday",
       y = "Average Duration",
       fill = "Member Type")
```

Calculate the average duration of rides per day of the week

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



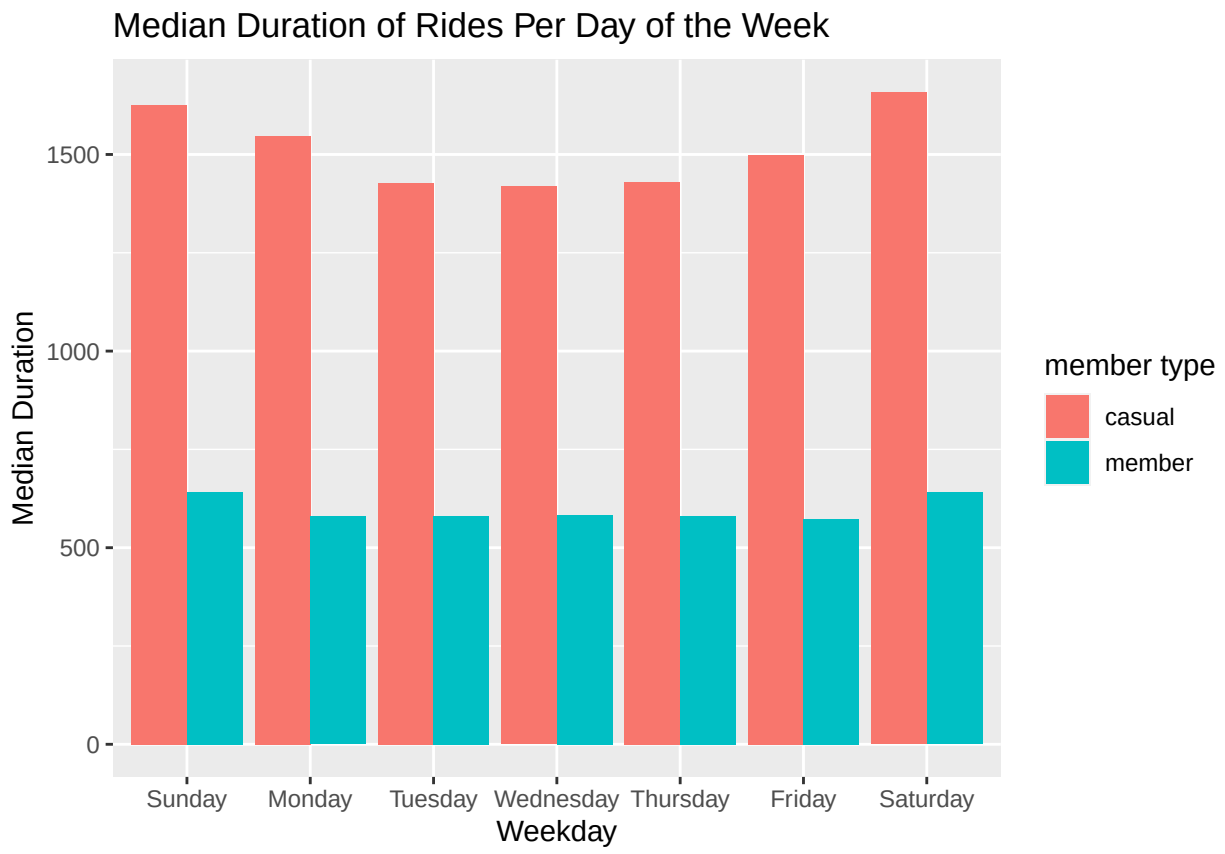

```
weekday_order <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")

all_divvy_trips_v2 %>%
  mutate(weekday_num = wday(started_at)) %>%
  mutate(weekday = factor(weekday_num, levels = 1:7, labels = weekday_order)) %>%

  group_by(member_casual, weekday) %>%
  summarize(number_of_rides = n(), median_duration = median(ride_length)) %>%
  arrange(member_casual, weekday) %>%
  ggplot(aes(x = weekday, y = median_duration, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(title = "Median Duration of Rides Per Day of the Week",
       x = "Weekday",
       y = "Median Duration",
       fill = "member type")
```

Median ride length per day of week

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

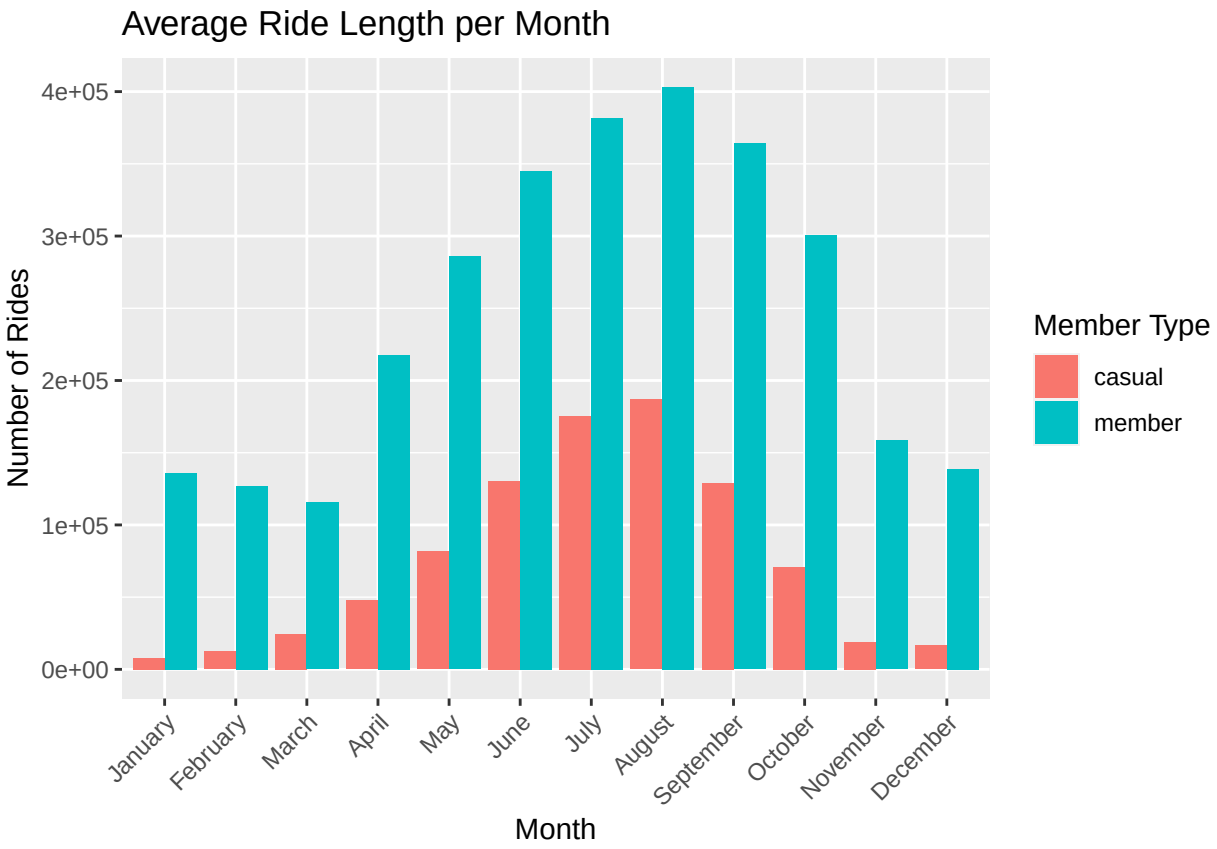



```
# Define the custom order of months starting with January
custom_month_order <- c("January", "February", "March", "April", "May", "June",
                        "July", "August", "September", "October", "November", "December")

all_divvy_trips_v2 %>%
  mutate(month = factor(format(started_at, "%B"), levels = custom_month_order)) %>%
  group_by(member_casual, month) %>%
  summarize(number_of_rides = n(), average_duration = mean(ride_length)) %>%
  arrange(member_casual, month) %>%
  ggplot(aes(x = month, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(title = "Average Ride Length per Month",
       x = "Month",
       y = "Number of Rides",
       fill = "Member Type") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Calculate and visualize the average ride length per month

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



Summary

Analyzing historical bike data from 2019 — 2020, we observed how annual members and casual riders use Cyclistic bikes differently. Key findings include:

- Both groups prefer Sunday and Saturday for longer rides, resulting in a U-shaped trend with a greater dip in the midweek for casual riders.
- Annual members have consistent ride durations throughout the week, while casual riders have varying durations.
- Annual members take more rides overall compared to casual riders.
- Annual members ride more on weekdays, while casual riders ride more on weekends.
- The busiest period for bike trips is July, August, and September, coinciding with the warmer summer months.

6: Act

In the Act phase, we will leverage the insights gained from our analysis of bike-share data to formulate actionable recommendations for stakeholders. By acting upon these recommendations, we aim to address the business problem effectively and facilitate data-driven decision-making.

Three Areas of Focus and Recommendations:

1. Seasonality

Insight

- There is an increase in demand for casual riders during Spring and Summer, presenting an opportunity for targeted advertising campaigns and promotional activities during these seasons.

Recommendation

- Launch a captivating email campaign for casual riders in early Spring, extending it throughout the Summer to attract and retain customers.
- Strategically schedule offers and promotions, such as early bird sign-ups and discounted memberships, encouraging potential riders to join ahead of the busy season. Highlight the advantages of becoming an annual member compared to remaining a casual rider.

2. Purpose of Bike Usage

Insight

- Annual members are more active mid-week with consistent ride lengths, while casual riders use bikes for weekend entertainment.

Recommendation

- Introduce enticing weekend passes tailored to weekend activities and entertainment, providing casual riders with flexible options.
- Leverage the power of digital marketing channels to effectively promote the new weekend passes, capturing the attention of potential riders.
- Identify popular spots frequented by casual riders using a visually appealing map, establish collaborative partnerships, and offer exclusive membership discounts combined with entertainment pairings (e.g., restaurant discounts, entry passes to sports venues). Emphasize the unique advantages of becoming an annual member.

3. Usage Patterns

Insight

- Annual members predominantly use bikes on weekdays, whereas casual riders prefer weekends.

Recommendation

- Utilize captivating digital marketing channels to inspire bike usage during weekdays, positioning it as a sustainable and convenient commuting option. Highlight the benefits of biking for physical fitness, reducing traffic congestion, and contributing to a greener urban environment.