Google_Divvy_Analytics

Bijaya

2023-10-17

This is a data analysis with the data sets from a Chicago based bike rental company Cyclistic. It owns approx 5800 bikes and 692 docking stations across the city.I being a junior data analyst, have analysed the data and derived insights based on the points put forward by the stakeholders.

Data source: `https://divvy-tripdata.s3.amazonaws.com/index.html`
- Downloaded files: - Divvy_Trips_2019_Q1.zip, - Divvy_Trips_2019_Q2.zip, - Divvy_Trips_2019_Q3.zip, - Divvy_Trips_2019_Q4.zip

```
options(repos = c(CRAN = "https://cran.r-project.org"))
```

```
install.packages("dbplyr")
```

```
##
## The downloaded binary packages are in
##  /var/folders/yw/h_p1t11s6r58nxnr10k9cy9r0000gn/T//RtmpxcE3aK/downloaded_packages
```

```
install.packages("RMySQL")
```

```
##
## The downloaded binary packages are in
##  /var/folders/yw/h_p1t11s6r58nxnr10k9cy9r0000gn/T//RtmpxcE3aK/downloaded_packages
```

```
install.packages("DBI")
```

```
##
## The downloaded binary packages are in
##  /var/folders/yw/h_p1t11s6r58nxnr10k9cy9r0000gn/T//RtmpxcE3aK/downloaded_packages
```

```
library(tidyverse)  #helps wrangle data
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.3     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.3     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.0
## ✔ 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 bec
ome errors
```

```
library(lubridate)  #helps wrangle date attributes
library(ggplot2)  #helps visualize data
library(dplyr)
library(RMySQL)
```

```
## Loading required package: DBI
```

```
library(DBI)
getwd() #displays your working directory
```

```
## [1] "/Users/bijayamanandhar/Desktop/cyclistic_analytics"
```

loading data

```
# Upload Divvy datasets (csv files) here
q1 <- read_csv("Divvy_Trips_2019_Q1.csv", show_col_types = FALSE)
q2 <- read_csv("Divvy_Trips_2019_Q2.csv", show_col_types = FALSE)
q3 <- read_csv("Divvy_Trips_2019_Q3.csv", show_col_types = FALSE)
q4 <- read_csv("Divvy_Trips_2019_Q4.csv", show_col_types = FALSE)
```

All dataframes combined

```
all_trips <- bind_rows(q1, q2, q3, q4)
```

Summary

```
nrow(all_trips)  #How many rows are in data frame?
```

```
## [1] 3166273
```

```
dim(all_trips)  #Dimensions of the data frame?
```

```
## [1] 3166273      12
```

```
head(all_trips)  #See the first 6 rows of data frame.  Also tail(qs_raw)
```

```
## # A tibble: 6 × 12
##    trip_id start_time  end_time    bike_id trip_duration_sec from_station_id
##      <dbl> <chr>       <chr>         <dbl>             <dbl>           <dbl>
## 1 21742443 1/1/19 0:04 1/1/19 0:11    2167               390             199
## 2 21742444 1/1/19 0:08 1/1/19 0:15    4386               441              44
## 3 21742445 1/1/19 0:13 1/1/19 0:27    1524               829              15
## 4 21742446 1/1/19 0:13 1/1/19 0:43     252              1783             123
## 5 21742447 1/1/19 0:14 1/1/19 0:20    1170               364             173
## 6 21742448 1/1/19 0:15 1/1/19 0:19    2437               216              98
## # ℹ 6 more variables: from_station_name <chr>, to_station_id <dbl>,
## #   to_station_name <chr>, user_type <chr>, gender <chr>, birth_year <dbl>
```

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

```
## spc_tbl_ [3,166,273 × 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ trip_id          : num [1:3166273] 21742443 21742444 21742445 21742446 21742447 ...
##  $ start_time       : chr [1:3166273] "1/1/19 0:04" "1/1/19 0:08" "1/1/19 0:13" "1/1/19 0:1
3" ...
##  $ end_time         : chr [1:3166273] "1/1/19 0:11" "1/1/19 0:15" "1/1/19 0:27" "1/1/19 0:4
3" ...
##  $ bike_id          : num [1:3166273] 2167 4386 1524 252 1170 ...
##  $ trip_duration_sec: num [1:3166273] 390 441 829 1783 364 ...
##  $ from_station_id  : num [1:3166273] 199 44 15 123 173 98 98 211 150 268 ...
##  $ from_station_name: chr [1:3166273] "Wabash Ave & Grand Ave" "State St & Randolph St" "Ra
cine Ave & 18th St" "California Ave & Milwaukee Ave" ...
##  $ to_station_id    : num [1:3166273] 84 624 644 176 35 49 49 142 148 141 ...
##  $ to_station_name  : chr [1:3166273] "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren
St (*)" "Western Ave & Fillmore St (*)" "Clark St & Elm St" ...
##  $ user_type        : chr [1:3166273] "Subscriber" "Subscriber" "Subscriber" "Subscriber"
...
##  $ gender           : chr [1:3166273] "Male" "Female" "Female" "Male" ...
##  $ birth_year       : num [1:3166273] 1989 1990 1994 1993 1994 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   trip_id = col_double(),
##   ..   start_time = col_character(),
##   ..   end_time = col_character(),
##   ..   bike_id = col_double(),
##   ..   trip_duration_sec = col_number(),
##   ..   from_station_id = col_double(),
##   ..   from_station_name = col_character(),
##   ..   to_station_id = col_double(),
##   ..   to_station_name = col_character(),
##   ..   user_type = col_character(),
##   ..   gender = col_character(),
##   ..   birth_year = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
summary(all_trips)  #Statistical summary of data. Mainly for numerics
```

```
##     trip_id          start_time          end_time            bike_id
##  Min.   :21742443   Length:3166273     Length:3166273     Min.   :   1
##  1st Qu.:22681250   Class :character   Class :character   1st Qu.:1730
##  Median :23673970   Mode  :character   Mode  :character   Median :3457
##  Mean   :23737432                                         Mean   :3387
##  3rd Qu.:24515363                                         3rd Qu.:5061
##  Max.   :25962904                                         Max.   :6946
##
##  trip_duration_sec  from_station_id from_station_name  to_station_id
##  Min.   :      61   Min.   :  1.0   Length:3166273     Min.   :  1.0
##  1st Qu.:     405   1st Qu.: 77.0   Class :character   1st Qu.: 77.0
##  Median :     698   Median :174.0   Mode  :character   Median :174.0
##  Mean   :    1441   Mean   :201.3                      Mean   :202.1
##  3rd Qu.:    1264   3rd Qu.:289.0                      3rd Qu.:290.0
##  Max.   :10628400   Max.   :673.0                      Max.   :673.0
##
##  to_station_name     user_type            gender            birth_year
##  Length:3166273     Length:3166273     Length:3166273     Min.   :1759
##  Class :character   Class :character   Class :character   1st Qu.:1979
##  Mode  :character   Mode  :character   Mode  :character   Median :1987
##                                                           Mean   :1984
##                                                           3rd Qu.:1992
##                                                           Max.   :2014
##                                                           NA's   :434079
```

Empty cells

```
colSums(is.na(all_trips)) #Check for missing cells in data frame
```

```
##           trip_id        start_time          end_time           bike_id
##                 0                 0                 0                 0
## trip_duration_sec   from_station_id from_station_name     to_station_id
##                 0                 0                 1                 0
##   to_station_name         user_type            gender        birth_year
##                 0                 0            450826            434079
```

Remove unwanted cols first

then rows with empty cells

```
all_trips <- subset(all_trips, select = - c(trip_id, bike_id, gender, birth_year))
all_trips <- na.omit(all_trips)
colnames(all_trips)
```

```
## [1] "start_time"        "end_time"          "trip_duration_sec"
## [4] "from_station_id"   "from_station_name" "to_station_id"
## [7] "to_station_name"   "user_type"
```

check to confirm rows with empty cells are removed

```
all_trips[!complete.cases(all_trips), ]
```

```
## # A tibble: 0 × 8
## # ℹ 8 variables: start_time <chr>, end_time <chr>, trip_duration_sec <dbl>,
## #   from_station_id <dbl>, from_station_name <chr>, to_station_id <dbl>,
## #   to_station_name <chr>, user_type <chr>
```

Rename Colnames for convenience

```
all_trips <- rename(all_trips
                    ,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 = user_type
                    )
```

convert date-time column types into `POSIXct`

```
all_trips$started_at <- as.POSIXct(all_trips$started_at, format = "%m/%d/%y %H:%M")
all_trips$ended_at <- as.POSIXct(all_trips$ended_at, format = "%m/%d/%y %H:%M")
```

remove rows with "`ended_at < started_at`" values

```
all_trips <- subset(all_trips, started_at < ended_at)
dim(all_trips)
```

```
## [1] 3166259       8
```

check if above conversion took place

```
str(all_trips$started_at)
```

```
##  POSIXct[1:3166259], format: "2019-01-01 00:04:00" "2019-01-01 00:08:00" "2019-01-01 00:13:
00" ...
```

```
str(all_trips$ended_at)
```

```
##  POSIXct[1:3166259], format: "2019-01-01 00:11:00" "2019-01-01 00:15:00" "2019-01-01 00:27:
00" ...
```

add a column for day of the week

```
all_trips$month_name <- months(as.Date(all_trips$started_at))
all_trips$day <- format(as.Date(all_trips$started_at), "%d")
all_trips$year <- format(as.Date(all_trips$started_at), "%Y")
all_trips$day_name <- format(as.Date(all_trips$started_at), "%A")
```

check for how `day_name` column will look like

```
random_sample <- all_trips[sample(nrow(all_trips), 10), ]
random_sample
```

```
## # A tibble: 10 × 12
##    started_at          ended_at            trip_duration_sec start_station_id
##    <dttm>              <dttm>                          <dbl>            <dbl>
##  1 2019-12-10 13:12:00 2019-12-10 13:14:00               103               43
##  2 2019-07-07 10:50:00 2019-07-07 11:16:00              1548              132
##  3 2019-03-23 15:10:00 2019-03-23 15:16:00               376              107
##  4 2019-04-11 18:18:00 2019-04-11 18:24:00               371              636
##  5 2019-12-16 16:31:00 2019-12-16 16:35:00               228               40
##  6 2019-04-15 13:59:00 2019-04-15 14:09:00               635               25
##  7 2019-01-20 13:25:00 2019-01-20 13:36:00               646              115
##  8 2019-11-14 06:43:00 2019-11-14 06:53:00               585              192
##  9 2019-04-30 18:09:00 2019-04-30 18:15:00               358               87
## 10 2019-07-19 17:25:00 2019-07-19 17:34:00               523              341
## # ℹ 8 more variables: start_station_name <chr>, end_station_id <dbl>,
## #   end_station_name <chr>, member_casual <chr>, month_name <chr>, day <chr>,
## #   year <chr>, day_name <chr>
```

Minimum

```
min(all_trips$trip_duration_sec)
```

```
## [1] 61
```

Reassign to the desired column to easy-to-remember values

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

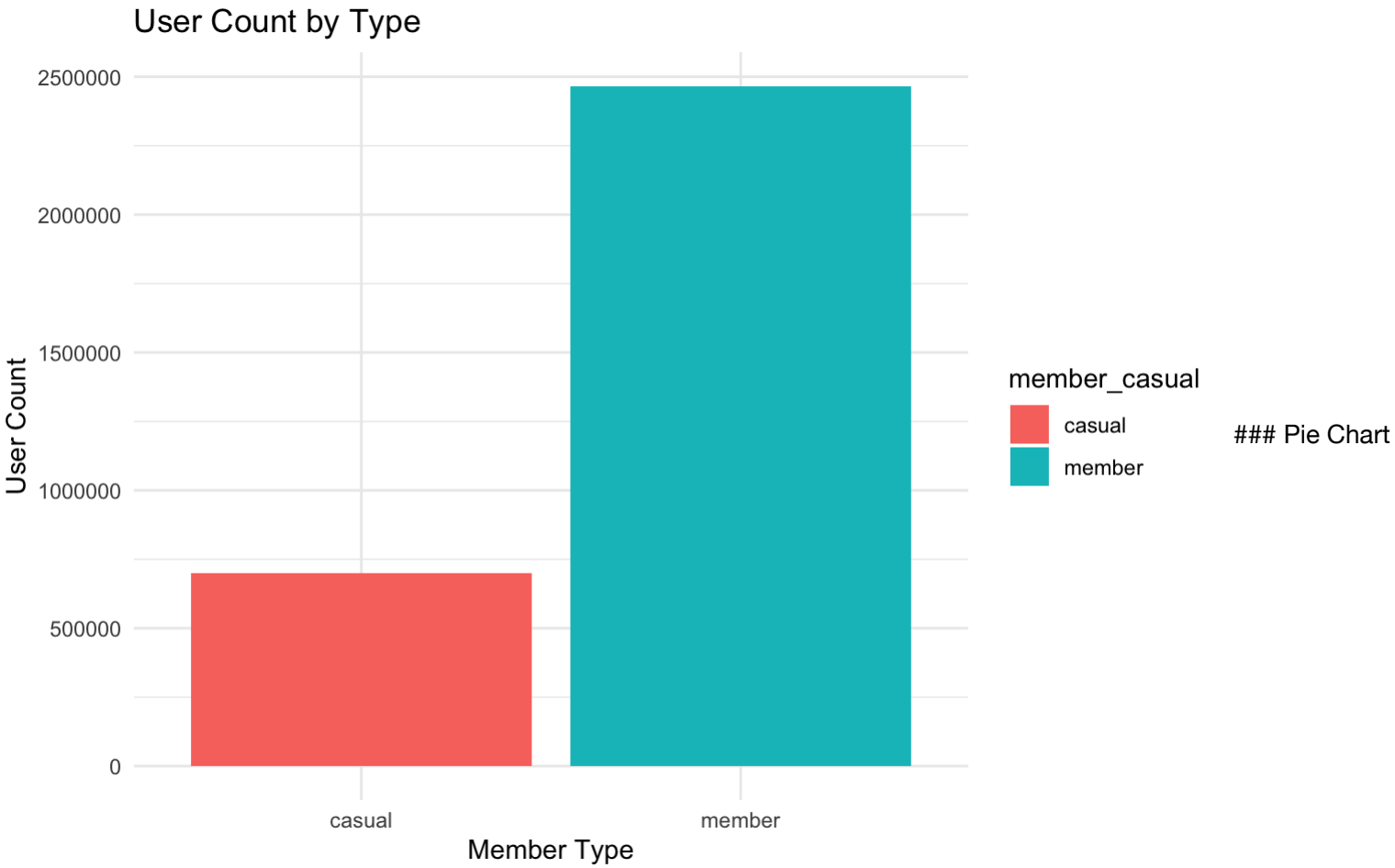
Table by user_type

```
yearly_count_by_user <- all_trips %>%
  group_by(member_casual) %>%
    summarise(count = n())
yearly_count_by_user
```

```
## # A tibble: 2 × 2
##   member_casual   count
##   <chr>           <int>
## 1 casual         700311
## 2 member        2465948
```

Bar Plot for User Count in 2019

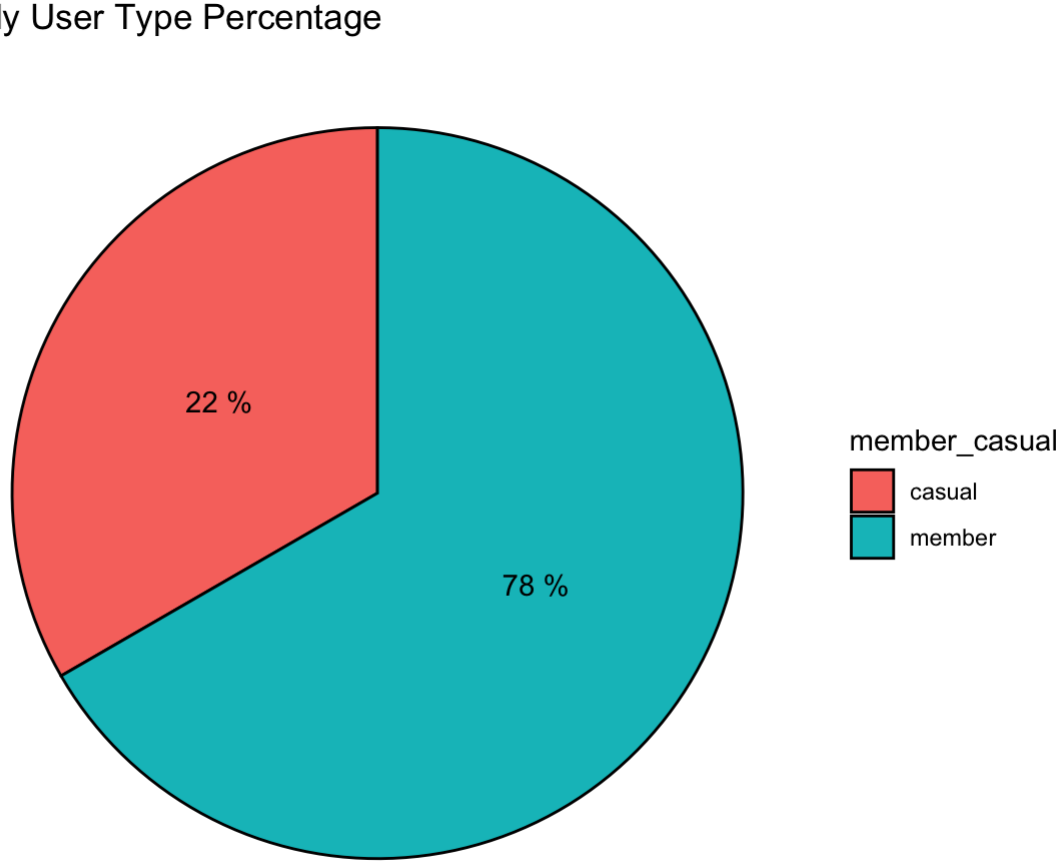
```
# Create a ggplot for the df_mean data frame
ggplot(yearly_count_by_user, aes(x = member_casual, y = count, fill = member_casual)) +
  geom_bar(stat = "identity") +
  labs(
    title = "User Count by Type",
    x = "Member Type",
    y = "User Count"
  ) +
  theme_minimal()
```



Representation

```
yearly_count_by_user$percent = paste(round((yearly_count_by_user $count /sum(yearly_count_by_u
ser $count))*100), "%")

ggplot(yearly_count_by_user , aes(x = "", y = percent, fill = member_casual)) +
  geom_col(color = "black") +
  geom_text(aes(label = percent),
            position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  theme_void()  + labs(title = "Yearly User Type Percentage")
```



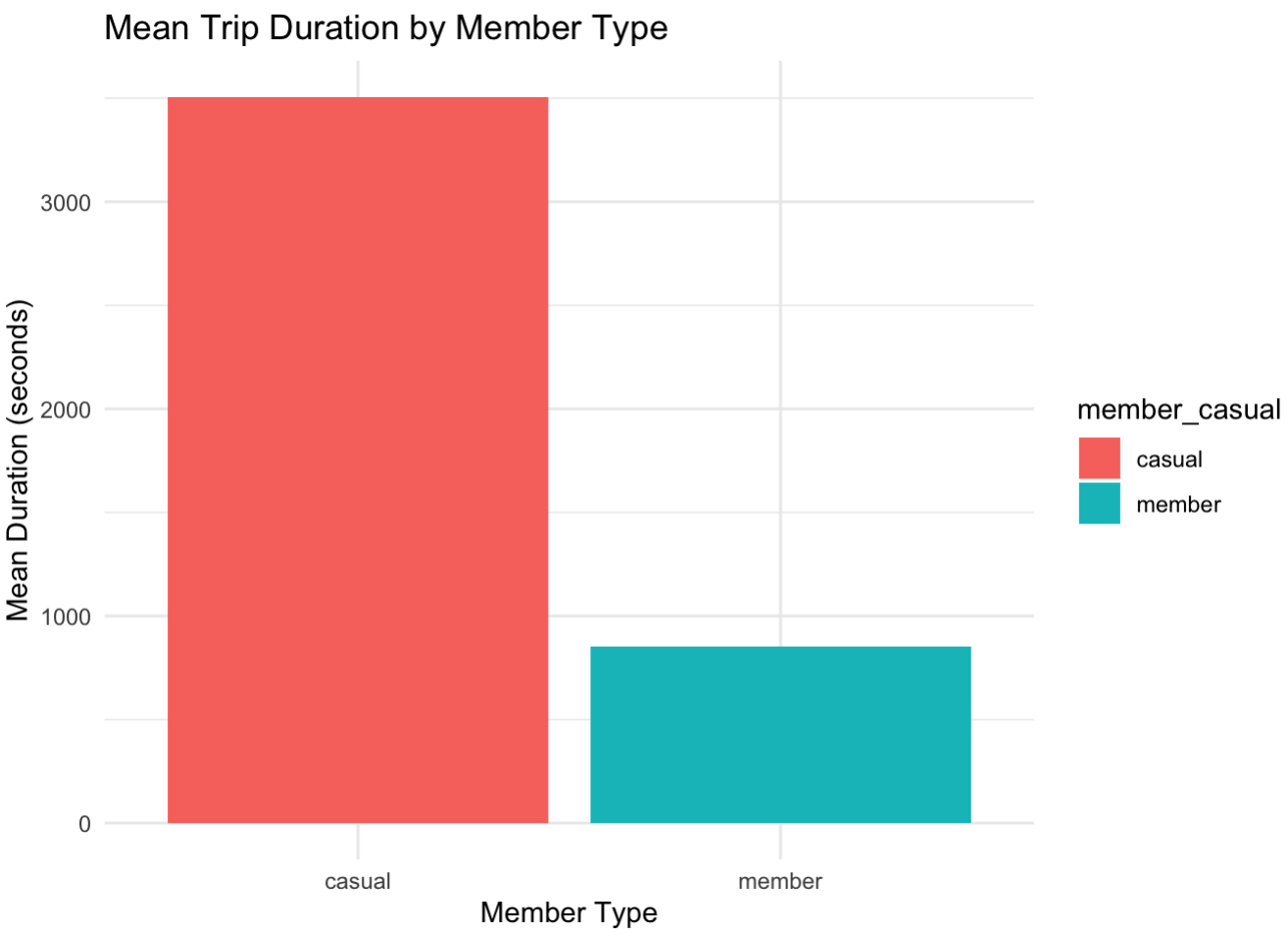
Mean ride_duration by user_type

```
yearly_mean_sec_by_user <- all_trips %>%
  group_by(member_casual) %>%
    summarise(mean_duration_sec = round(mean(trip_duration_sec)))
yearly_mean_sec_by_user
```

```
## # A tibble: 2 × 2
##   member_casual mean_duration_sec
##   <chr>                     <dbl>
## 1 casual                     3506
## 2 member                      854
```

Bar Plot yearly mean trip duration by user_type

```
# Create a ggplot for the df_mean data frame
ggplot(yearly_mean_sec_by_user, aes(x = member_casual, y = mean_duration_sec, fill = member_ca
sual)) +
  geom_bar(stat = "identity") +
  labs(
    title = "Mean Trip Duration by Member Type",
    x = "Member Type",
    y = "Mean Duration (seconds)"
  ) +
  theme_minimal()
```



Bar chart for `rental count vs day of the week`

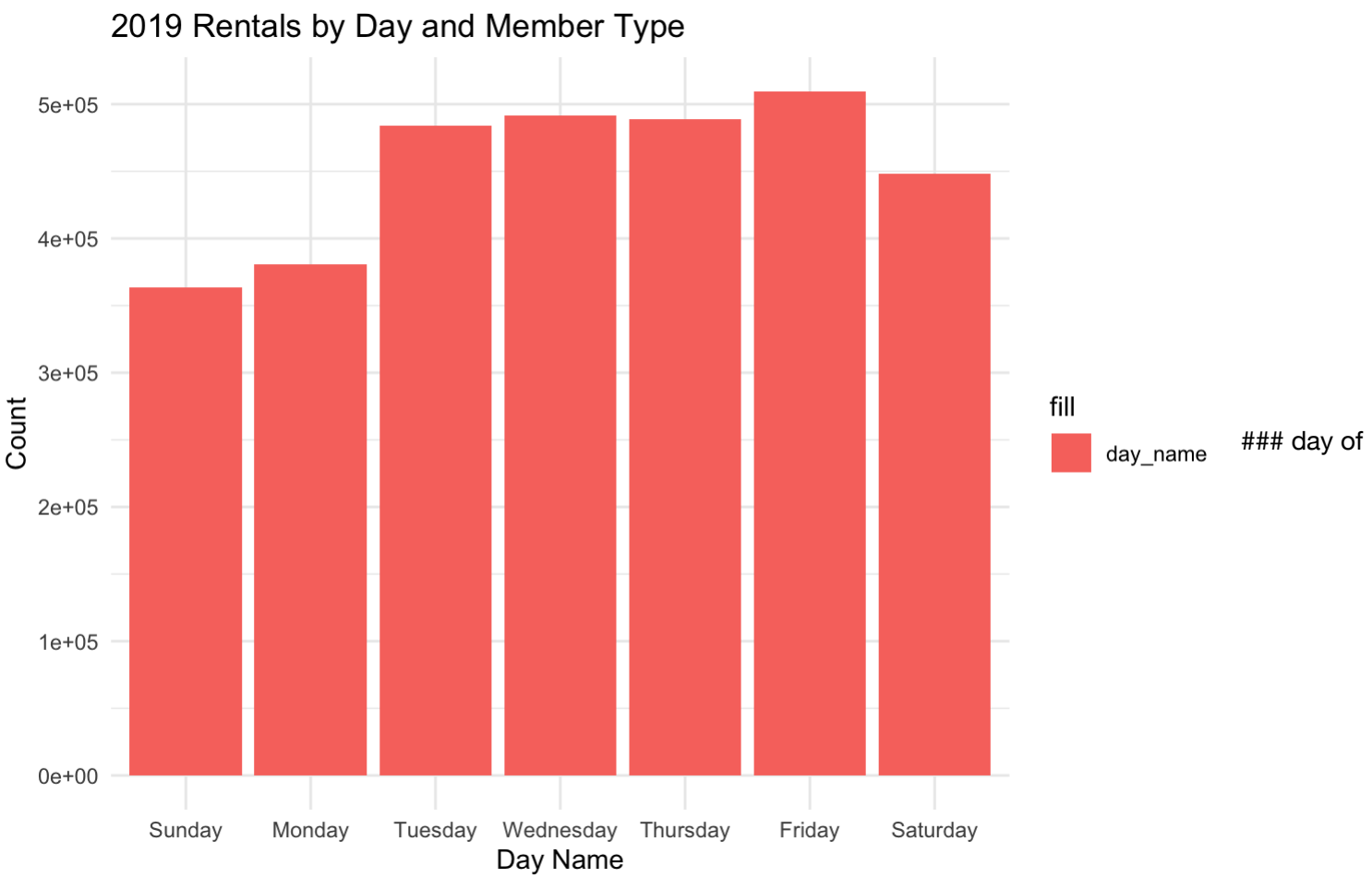
```
# Specify the order of days of the week
day_order <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")

day_of_week_total <- all_trips %>%
  mutate(day_name = factor(day_name, level = day_order, ordered = TRUE)) %>%
    group_by(day_name) %>%
      summarise(count = n(), .groups="drop")
day_of_week_total
```

```
## # A tibble: 7 × 2
##   day_name   count
##   <ord>      <int>
## 1 Sunday    363242
## 2 Monday    380906
## 3 Tuesday   484276
## 4 Wednesday 491558
## 5 Thursday  488589
## 6 Friday    509597
## 7 Saturday  448091
```

Bar Plot average trip count against day of week

```
ggplot(day_of_week_total, aes(x = day_name, y = count, fill = "day_name")) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "2019 Rentals by Day and Member Type", x = "Day Name", y = "Count") +
  theme_minimal()
```



the week by Member Type

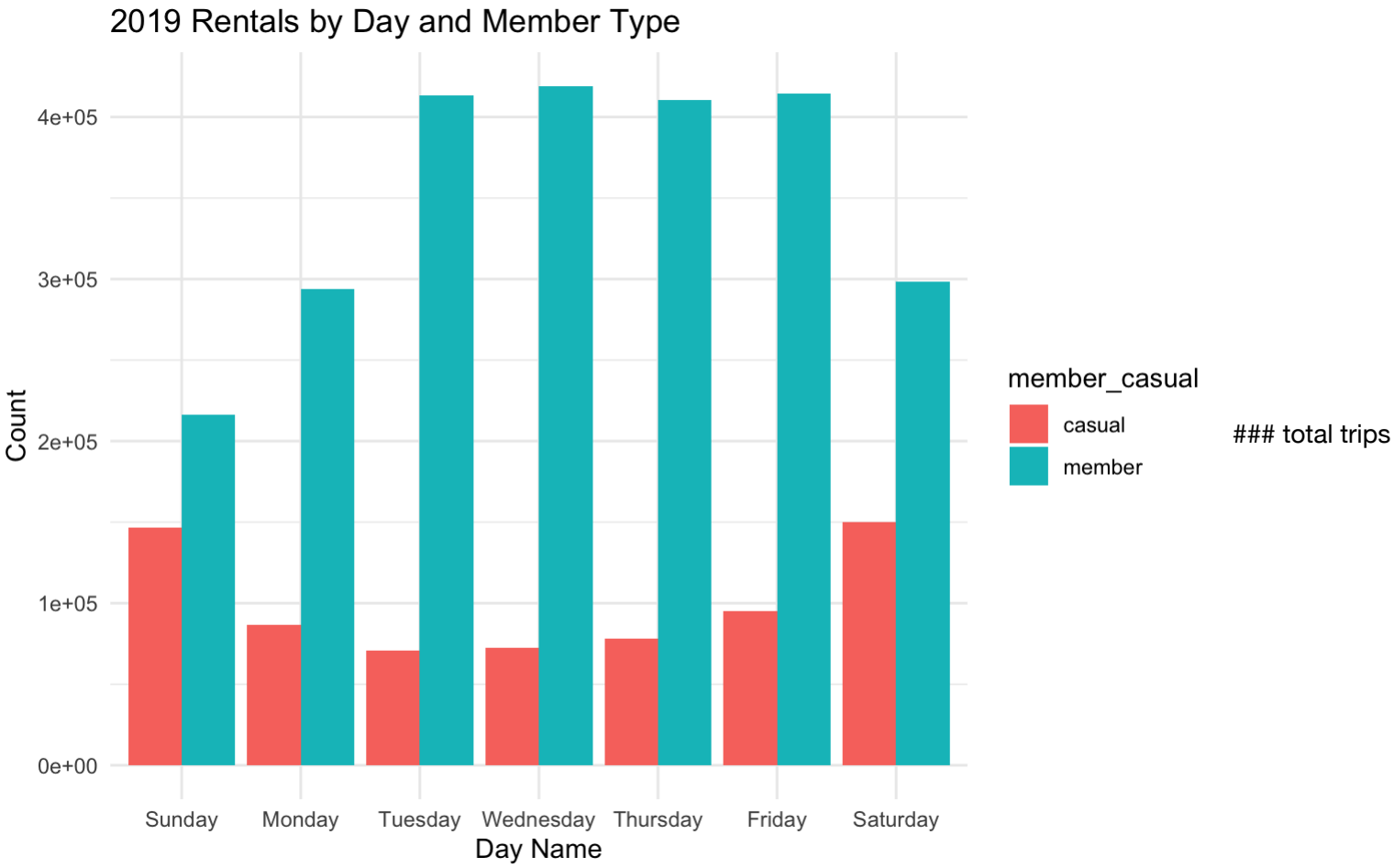
```
day_of_week_by_user <- all_trips %>%
  mutate(day_name = factor(day_name, level = day_order, ordered = TRUE)) %>%
    group_by(day_name, member_casual) %>%
      summarise(count = n(), .groups = "drop")
day_of_week_by_user
```

```
## # A tibble: 14 × 3
##    day_name  member_casual  count
##    <ord>     <chr>          <int>
##  1 Sunday    casual        146739
##  2 Sunday    member        216503
##  3 Monday    casual         86850
##  4 Monday    member        294056
##  5 Tuesday   casual         70719
##  6 Tuesday   member        413557
##  7 Wednesday casual         72479
##  8 Wednesday member        419079
##  9 Thursday  casual         78349
## 10 Thursday  member        410240
## 11 Friday    casual         95392
## 12 Friday    member        414205
## 13 Saturday  casual        149783
## 14 Saturday  member        298308
```

Bar Plot average trip count for day of week by user type

```
# Create the bar chart
# Convert day_name to a factor with the desired order
day_of_week_by_user$day_name <- factor(day_of_week_by_user$day_name, levels = day_order)

ggplot(day_of_week_by_user, aes(x = day_name, y = count, fill = member_casual)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "2019 Rentals by Day and Member Type", x = "Day Name", y = "Count") +
  theme_minimal()
```



total trips

each month

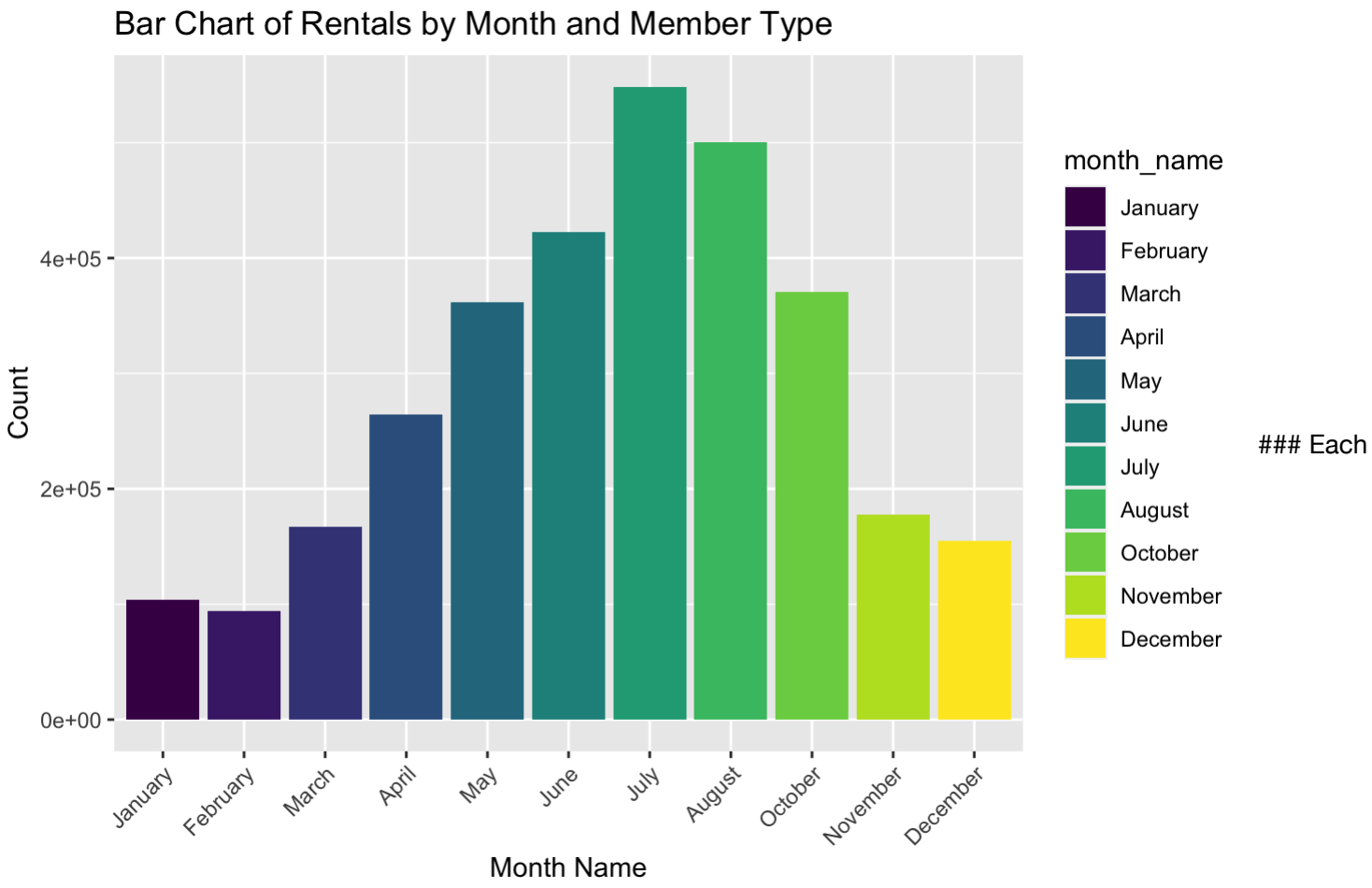
```
# Specify the order of days of the week
month_order <- c("January", "February", "March", "April", "May", "June", "July", "August", "Se
ptember", "October", "November", "December")

month_total <- all_trips %>%
  mutate(month_name = factor(month_name, level = month_order, ordered = TRUE)) %>%
    group_by(month_name) %>%
      summarise(count = n())
month_total
```

```
## # A tibble: 11 × 2
##    month_name  count
##    <ord>       <int>
##  1 January    103804
##  2 February    94375
##  3 March      166787
##  4 April      264376
##  5 May        362104
##  6 June       422915
##  7 July       548544
##  8 August     500031
##  9 October    370549
## 10 November   177941
## 11 December   154833
```

Bar Plot total trips each month

```
ggplot(month_total, aes(x = month_name, y = count, fill = month_name)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Bar Chart of Rentals by Month and Member Type", x = "Month Name", y = "Count")
+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Adjust the angle as needed
```



Each

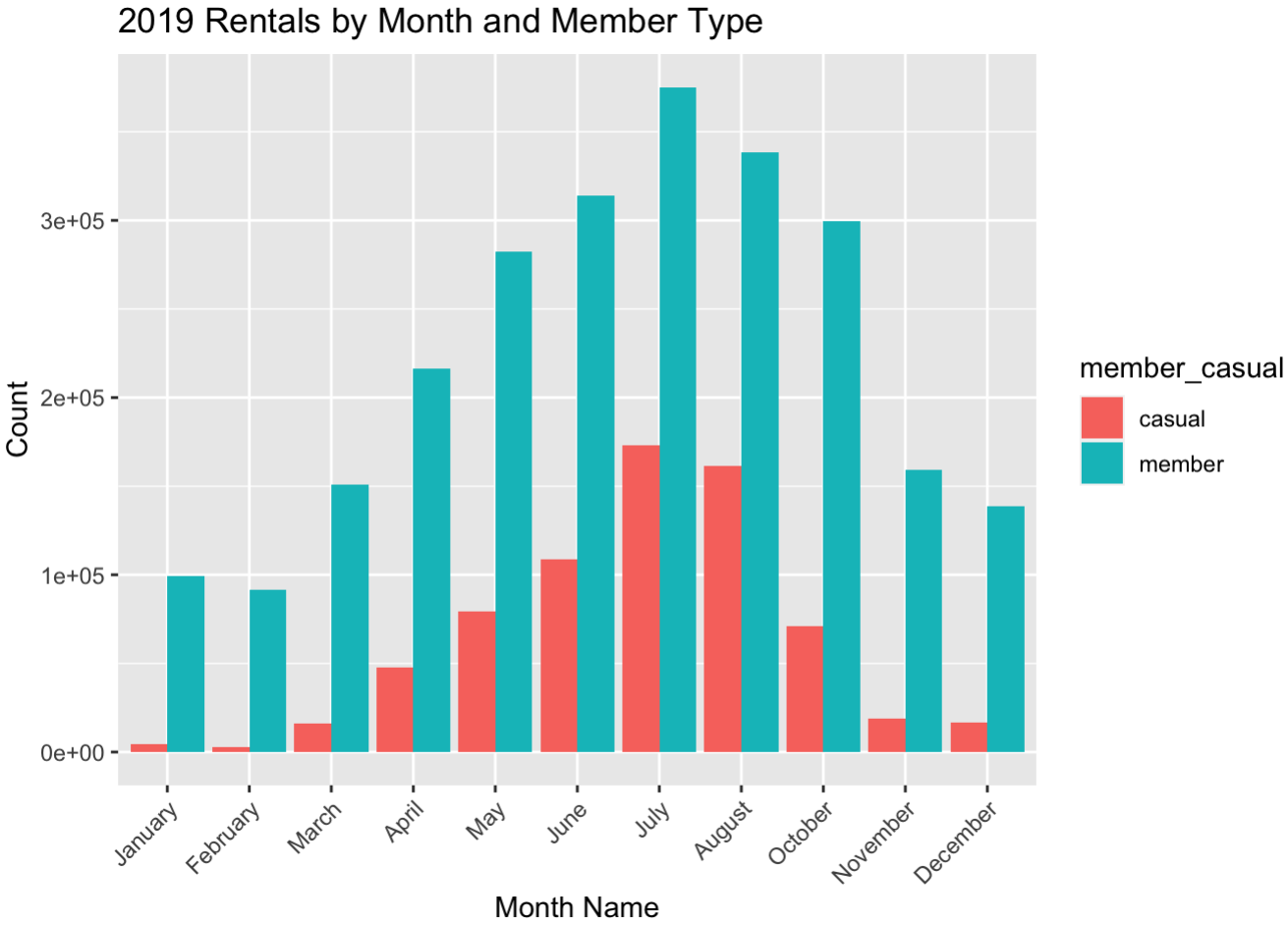
month total by user type

```
month_by_user <- all_trips %>%
  mutate(month_name = factor(month_name, levels  = month_order, ordered = TRUE)) %>%
    group_by(month_name, member_casual) %>%
      summarise(count = n(), .groups = "drop")
month_by_user
```

```
## # A tibble: 22 × 3
##    month_name member_casual  count
##    <ord>      <chr>          <int>
##  1 January    casual          4680
##  2 January    member         99124
##  3 February   casual          2596
##  4 February   member         91779
##  5 March      casual         15841
##  6 March      member        150946
##  7 April      casual         47818
##  8 April      member        216558
##  9 May        casual         79492
## 10 May        member        282612
## # ℹ 12 more rows
```

Bar Plot Each month trip count by user type

```
ggplot(month_by_user, aes(x = month_name, y = count, fill = member_casual)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "2019 Rentals by Month and Member Type", x = "Month Name", y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Adjust the angle as needed
```



Data migration to MySQL database

DataFrame `all_trips` is migrated to MySQL table

`Divvy_Trips_2019` under database `Divvy_Case_Study

```
con <- dbConnect(
  RMySQL::MySQL(),
  dbname = "Divvy_Case_Study",
  host = "localhost",
  username = "root",
  password = "Mnandhar2015_mac"    # encrypted for privacy
)

dbWriteTable(con, name = 'yearly_count_by_user', value = yearly_count_by_user, overwrite = TRU
E)
```

```
## [1] TRUE
```

```
dbWriteTable(con, name = 'yearly_mean_sec_by_user', value = yearly_mean_sec_by_user, overwrite
= TRUE)
```

```
## [1] TRUE
```

```
dbWriteTable(con, name = 'day_of_week_total', value = day_of_week_total, overwrite = TRUE)
```

```
## [1] TRUE
```

```
dbWriteTable(con, name = 'day_of_week_by_user', value = day_of_week_by_user, overwrite = TRUE)
```

```
## [1] TRUE
```

```
dbWriteTable(con, name = 'month_total', value = month_total, overwrite = TRUE)
```

```
## [1] TRUE
```

```
dbWriteTable(con, name = 'month_by_user', value = month_by_user, overwrite = TRUE)
```

```
## [1] TRUE
```

```
dbWriteTable(con, name = 'Divvy_Trips_2019', value = all_trips, overwrite = TRUE)
```

```
## [1] TRUE
```

```
dbDisconnect(con)
```

```
## [1] TRUE
```