Google Data Analytics Certificate - Coursera Capstone

Installing packages

```
#install.packages("tidyverse")
#install.packages("markdown")
#install.packages("sqldf")
#install.packages("maps")
#install.packages("rgdal")
#install.packages("ggrepel")


 library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
## v ggplot2 3.3.3          v purrr     0.3.4
## v tibble 3.1.0           v dplyr     1.0.5
## v tidyr      1.1.3       v stringr 1.4.0
## v readr      1.4.0       v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() -## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("lubridate")
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base': ##
##                 date, intersect, setdiff, union
```

```
library("markdown") library("sqldf")
```

```
## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite
```

```
library("maps")
```

```
##
## Attaching package: 'maps'

## The following object is masked from 'package:purrr':
```

```
##
##          map
```

```
library("rgdal")
```

```
## Loading required package: sp
```

```
## rgdal: version: 1.5-23, (SVN revision 1121)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.2.1, released 2020/12/29
## Path to GDAL shared files: C:/Users/quent/OneDrive/Documents/R/win-library/4.0/rgdal/gdal ##
GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 7.2.1, January 1st, 2021, [PJ_VERSION: 721]
## Path to PROJ shared files: C:/Users/quent/OneDrive/Documents/R/win-library/4.0/rgdal/proj ##
PROJ CDN enabled: FALSE
## Linking to sp version:1.4-5
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
## use options("rgdal_show_exportToProj4_warnings"="none") before loading rgdal.
## Overwritten PROJ_LIB was C:/Users/quent/OneDrive/Documents/R/win-library/4.0/rgdal/proj
```

```
library("ggrepel")
```

Setting working directory, and creating dataframes for each .csv file.

```
setwd("/Users/quent/OneDrive/Documents/R/Projects/Google Capstone/CSVs/")

apr_20 <- read.csv("apr_20.csv", sep=";") may_20 <-
read.csv("may_20.csv", sep=";") jun_20 <-
read.csv("jun_20.csv", sep=";") jul_20 <-
read.csv("jul_20.csv", sep=";") aug_20 <-
read.csv("aug_20.csv", sep=";") sep_20 <-
read.csv("sep_20.csv", sep=";") oct_20 <-
read.csv("oct_20.csv", sep=";") nov_20 <-
read.csv("nov_20.csv", sep=";") dec_20 <-
read.csv("dec_20.csv", sep=";") jan_21 <-
read.csv("jan_21.csv", sep=";") feb_21 <-
read.csv("feb_21.csv", sep=";") mar_21 <-
read.csv("mar_21.csv", sep=";")
```

Glimpsing a dataframe, to see if data types from excel were preserved (they weren’t)

```
glimpse(dec_20)
```

```
## Rows: 131,139 ##
Columns: 15
```

```
## $ ride_id          <chr> "1C46BF5EB60CC524", "1405BFC02FDB5190", "892ECFAB44~ ## $ rideable_type
 <chr> "electric_bike", "electric_bike", "docked_bike", "d~ ## $ started_at         <chr>  "01/12/2020  00:01",  "01/12/2020  00:01",
"01/12/2020~ ## $ ended_at          <chr> "01/12/2020 00:06", "01/12/2020 00:06", "01/12/2020~
## $ start_station_name <chr> "", "", "Larrabee St & Armitage Ave", "Wabash Ave &~
## $ start_station_id                           <chr> "", "", "TA1309000006", "KA1503000015", "TA13070001~
## $ end_station_name          <chr> "", "Wentworth Ave & 63rd St", "Sedgwick St & Webst~
## $ end_station_id            <chr> "", "KA1503000025", "13191", "13158", "13108", "TA1~
## $ start_lat                 <chr> "41.79", "41.78", "41.918.084", "41.879.472", "41.9~
## $ start_lng                 <chr> "-87.59", "-87.62", "-87.643.749", "-87.625.688", "~
## $ end_lat                   <chr> "41.8", "4.178.009.516.666.660", "41.922.167", "418~
## $ end_lng                   <chr> "-87.6", "-876.297.085", "-87.638.888", "-8.764.961~
## $ member_casual             <chr> "member", "casual", "member", "member", "member", "~
## $ ride_length               <chr> "00:05:38", "00:05:06", "00:02:54", "00:09:54", "00~
## $ day_of_week               <int> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, ~
```

Merging all the dataframes together

first, calculate the number of rows in total to verify the merge

```
tot_rows <- nrow(apr_20) + nrow(may_20) + nrow(jun_20) + nrow(jul_20) + nrow(aug_20) + nrow(sep_20) + nrow(oct_20) +

nrow(nov_20) + nrow(dec_20) + nrow(jan_21) + nrow(feb_21) + nrow(mar_21)
```

then create the bind

```
df_1 <- do.call("rbind", list(apr_20, may_20, jun_20, jul_20, aug_20, sep_20, oct_20, nov_20, dec_20,
jan_21, feb_21, mar_21))
```

checking the number of rows match up

```
if (tot_rows == nrow(df_1)){ print("Binding complete, data verified.")
} else{ print("Error, please verify your data.")
}
```

```
## [1] "Binding complete, data verified."
```

changing datatypes of started_at, ended_at to datetime and ride_length to time for all dataframes

```
df_1 <- df_1 %>%
mutate(started_at = as_datetime(df_1$started_at, format = "%d/%m/%Y %H:%M")) %>% mutate(ended_at =
as_datetime(df_1$ended_at, format = "%d/%m/%Y %H:%M")) %>% mutate(ride_length = as.difftime(df_1$ride_length,
format = "%H:%M:%S"))
```

A quick analysis to find the mean of the ride_length column, and the max ride length

```
mean_r_length <- as.numeric(mean(df_1$ride_length))/60
cat("The average ride length over the year is:",mean_r_length,"minutes")
```

```
## The average ride length over the year is: 24.41373 minutes
```

```
max_r_length <- as.numeric(max(df_1$ride_length))/3600 cat("The longest ride
for the year was:",max_r_length,"hours")
```

```
## The longest ride for the year was: 23.99833 hours
```

Now, going to create a new dataframe with the data I want for a visualisation.

I will use sqldf to demonstrate some of my SQL abilities. creating two dataframes with top 5 start & end stations + no. of trips per mem/cas

####Top 5 starting geolocations for members

```
mem_start_geo <- sqldf("SELECT member_casual, start_station_name AS Start,
start_lat AS Starting_Latitude, start_lng As Starting_Longitude, count(start_station_name) AS
Num_Trips FROM df_1
                         WHERE start_station_name IS NOT ''
                         AND member_casual = 'member' GROUP
                         BY start_station_name
                         ORDER BY count(start_station_name) DESC
                         LIMIT 5", method = "auto")
```

####Top 5 starting geolocations for casuals

```
cas_start_geo <- sqldf("SELECT member_casual, start_station_name AS Start, start_lat AS
                        Starting_Latitude, start_lng As Starting_Longitude,
                        count(start_station_name) AS Num_Trips
                        FROM df_1
                        WHERE start_station_name IS NOT ''
                        AND member_casual = 'casual' GROUP
                        BY start_station_name
                        ORDER BY count(start_station_name) DESC
                        LIMIT 5", method = "auto")
```

###Binding the two tables into a dataframe, and viewing it

```
start_geo <- rbind(mem_start_geo, cas_start_geo) View(start_geo)
```

Changing the datatype of the coordinates to real numbers to use for plots

```
start_geo$Starting_Latitude = as.numeric(gsub(",",".",start_geo$Starting_Latitude,fixed=TRUE) start_geo$Starting_Longitude
= as.numeric(gsub(",",".",start_geo$Starting_Longitude,fixed=TRUE))
```

####Top 5 ending geolocations for members

```
mem_end_geo <- sqldf("SELECT member_casual, end_station_name AS End, end_lat AS
                      Ending_Latitude, end_lng As Ending_Longitude,
                      count(end_station_name) AS Num_Trips FROM df_1
                      WHERE end_station_name IS NOT ''
                      AND member_casual = 'member'
                      GROUP BY end_station_name
                      ORDER BY count(end_station_name) DESC
                      LIMIT 5", method = "auto")
```

####Top 5 ending geolocations for casuals

```
cas_end_geo <- sqldf("SELECT member_casual, end_station_name AS End, end_lat AS
                      Ending_Latitude, end_lng As
                      Ending_Longitude, count(end_station_name)
                      AS Num_Trips FROM df_1
                      WHERE end_station_name IS NOT ''
                      AND member_casual = 'casual' GROUP
                      BY end_station_name
                      ORDER BY count(end_station_name) DESC
                      LIMIT 5", method = "auto")
```

###Binding the two tables into a dataframe, and viewing it

```
end_geo <- rbind(mem_end_geo, cas_end_geo) View(end_geo)
```

Changing the datatype of the coordinates to real numbers to use for plots

```
end_geo$Ending_Latitude = as.numeric(gsub(",",".",end_geo$Ending_Latitude, fixed=TRUE))

end_geo$Ending_Longitude = as.numeric(gsub(",",".",end_geo$Ending_Longitude, fixed=TRUE))
```

Creating a geolocation map of the top 5 start and end stations

###Getting a shapefile of Chicago, and fortifying it into a dataframe

```
chi_map <- readOGR(dsn="C:/Users/quent/OneDrive/Documents/R/Projects/Google Capstone/Maps",
layer="geo_export_b9804a71-fc32-4cb9-ac05-b4cc9364243d")
```

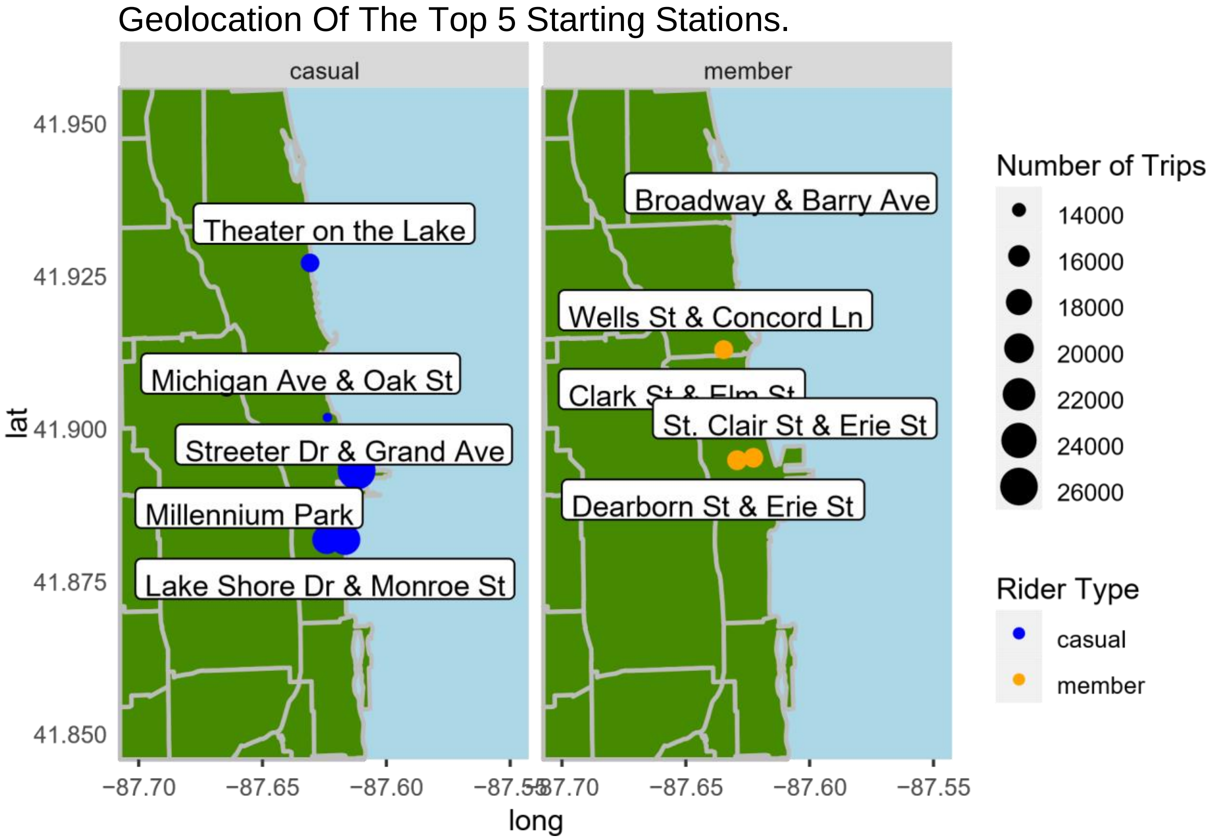
```
## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS =
## dumpSRS, : Discarded datum WGS84 in Proj4 definition: +proj=longlat +ellps=WGS84
## +no_defs

## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\quent\OneDrive\Documents\R\Projects\Google Capstone\Maps", layer:
"geo_export_b9804a71fc32-4cb9-ac05-b4cc9364243d"
## with 77 features
## It has 9 fields
```

```
chi_df = fortify(chi_map)
```

```
## Regions defined for each Polygons
```

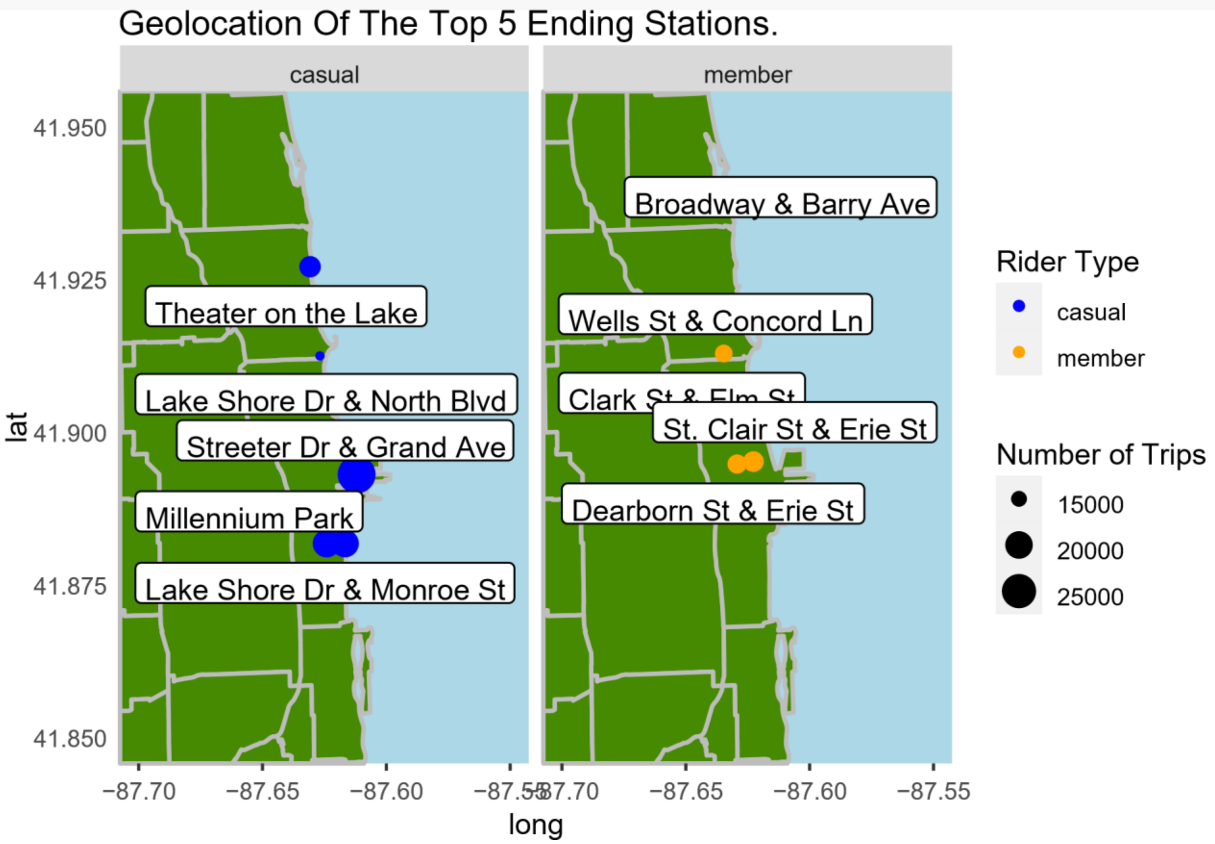
Plotting the start station geolocations.

```
ssgmap <-ggplot() +
geom_polygon(data = chi_df, aes(x = long, y=lat , group = group), colour = 'grey',
fill = 'chartreuse4', size = .7) + geom_point(data = start_geo,
aes(x = Starting_Longitude, y = Starting_Latitude, size = Num_Trips, color = member_casual), alpha
= 1) +
geom_label_repel(data = start_geo,
aes(x = Starting_Longitude, y = Starting_Latitude, label = Start),
box.padding = 0.4, point.padding = 0.65, segment.color =
'gray50') +
scale_colour_manual(values=c(member = 'orange', casual= 'blue'))+ facet_wrap(~member_casual)
+
labs(title = "Geolocation Of The Top 5 Starting Stations.", size = 'Number of
Trips', color = 'Rider Type') + coord_cartesian(xlim = c(-87.7, -87.55), ylim =
c(41.85, 41.95))+ theme(panel.background = element_rect(fill = "lightblue")) +
theme(panel.border = element_blank(), panel.grid.major = element_blank(),
panel.grid.minor = element_blank()) ssgmap
```



Plotting the end station geolocations.

```
esgmap <- ggplot() +
geom_polygon(data = chi_df, aes(x = long, y=lat , group = group), colour =
'grey', fill = 'chartreuse4', size = .7) + geom_point(data = end_geo,
aes(x = Ending_Longitude, y = Ending_Latitude, size = Num_Trips, color = member_casual),
alpha = 1) +
geom_label_repel(data = end_geo,
aes(x = Ending_Longitude, y = Ending_Latitude, label =
End), box.padding = 0.4, point.padding = 0.65,
segment.color = 'gray50') +
scale_colour_manual(values=c(member = 'orange', casual= 'blue')) +
facet_wrap(~member_casual) +
labs(title = "Geolocation Of The Top 5 Ending Stations.", size = 'Number of
Trips', color = 'Rider Type') + coord_cartesian(xlim = c(-87.7, -87.55), ylim =
c(41.85, 41.95)) + theme(panel.background = element_rect(fill = "lightblue")) +
theme(panel.border = element_blank(), panel.grid.major = element_blank(),
panel.grid.minor = element_blank()) esgmap
```



SQL Queries for the yearly Mode of day_of_week (total, members, casuals)

```
mode_t <- sqldf("SELECT day_of_week, member_casual, COUNT(day_of_week) AS Total
                 FROM df_1
                 GROUP BY member_casual, day_of_week
                 ORDER BY day_of_week DESC", method = "auto")
```

Replacing the numerical values with names of weekdays

```
mode_t$day_of_week[mode_t$day_of_week == "1"] <- "Sunday"

mode_t$day_of_week[mode_t$day_of_week == "2"] <- "Monday"

mode_t$day_of_week[mode_t$day_of_week == "3"] <- "Tuesday"

mode_t$day_of_week[mode_t$day_of_week == "4"] <- "Wednesday"

mode_t$day_of_week[mode_t$day_of_week == "5"] <- "Thursday"

mode_t$day_of_week[mode_t$day_of_week == "6"] <- "Friday"

mode_t$day_of_week[mode_t$day_of_week == "7"] <- "Saturday"
```

##Plotting the Modes

This function locks in the order I established so that x axis isn't sorted

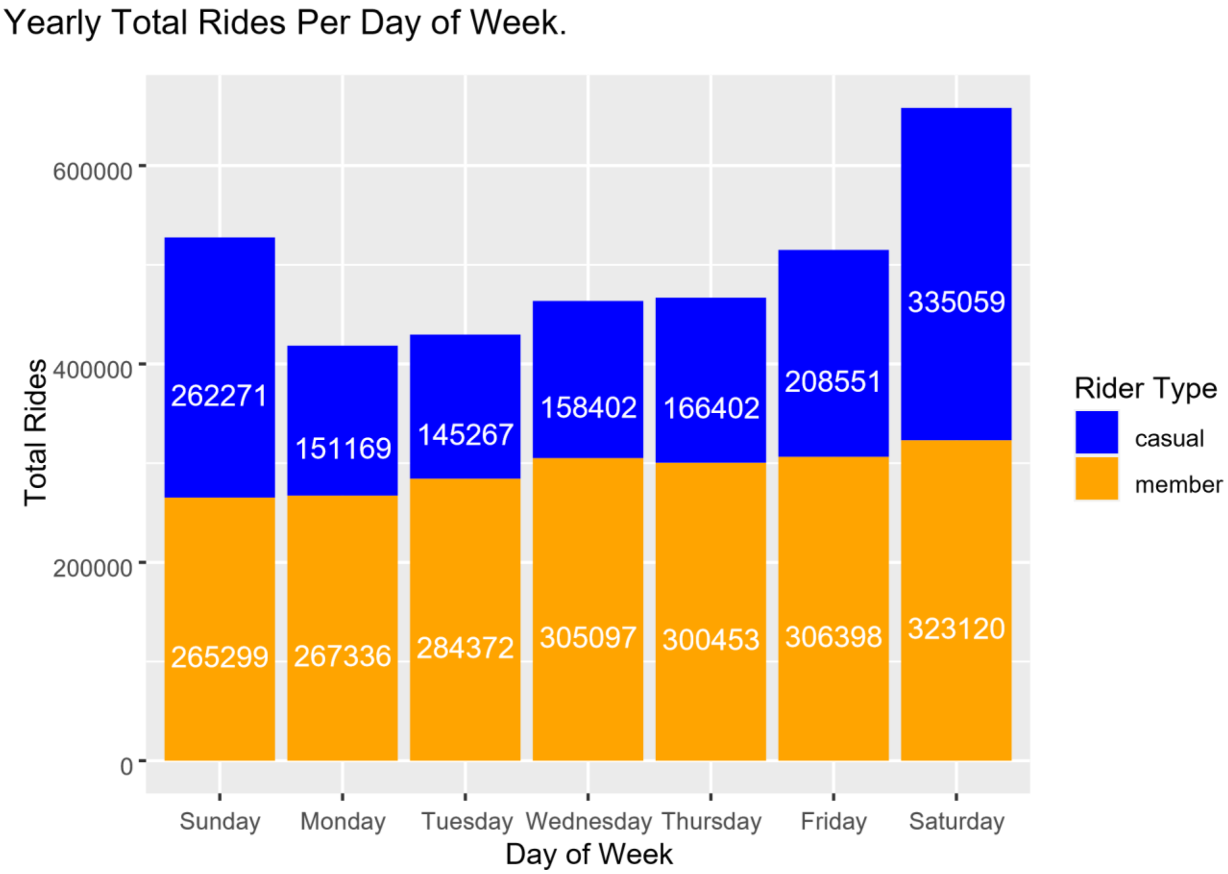
```
mode_t$day_of_week <- factor(mode_t$day_of_week, levels = rev(unique(mode_t$day_of_week)), ordered=TRUE)
```

This function finds the sum of casual and member riders, to be used to plot labels in the middle of each bar.

```
mode_t <- mode_t %>%
arrange(day_of_week, rev(member_casual)) %>% group_by(day_of_week) %>%
mutate(GTotal = cumsum(Total) - 0.5 * Total)
```

A stacked bar plot with the yearly modes for all riders

```
Mode_plot <- ggplot(data = mode_t, aes(x = day_of_week, y = Total, fill = member_casual))
+ scale_fill_manual(values=c(member = 'orange', casual= 'blue')) + geom_col() +
geom_text(aes(y = GTotal, label = Total), vjust = 1.5, colour = "white") + labs(title = "Yearly
Total Rides Per Day of Week.", x = "Day of Week", y = "Total Rides", fill = "Rider Type") +
scale_y_continuous(labels = function(x) format(x, scientific = FALSE))
Mode_plot
```



A query to return results related to rideble types used by members

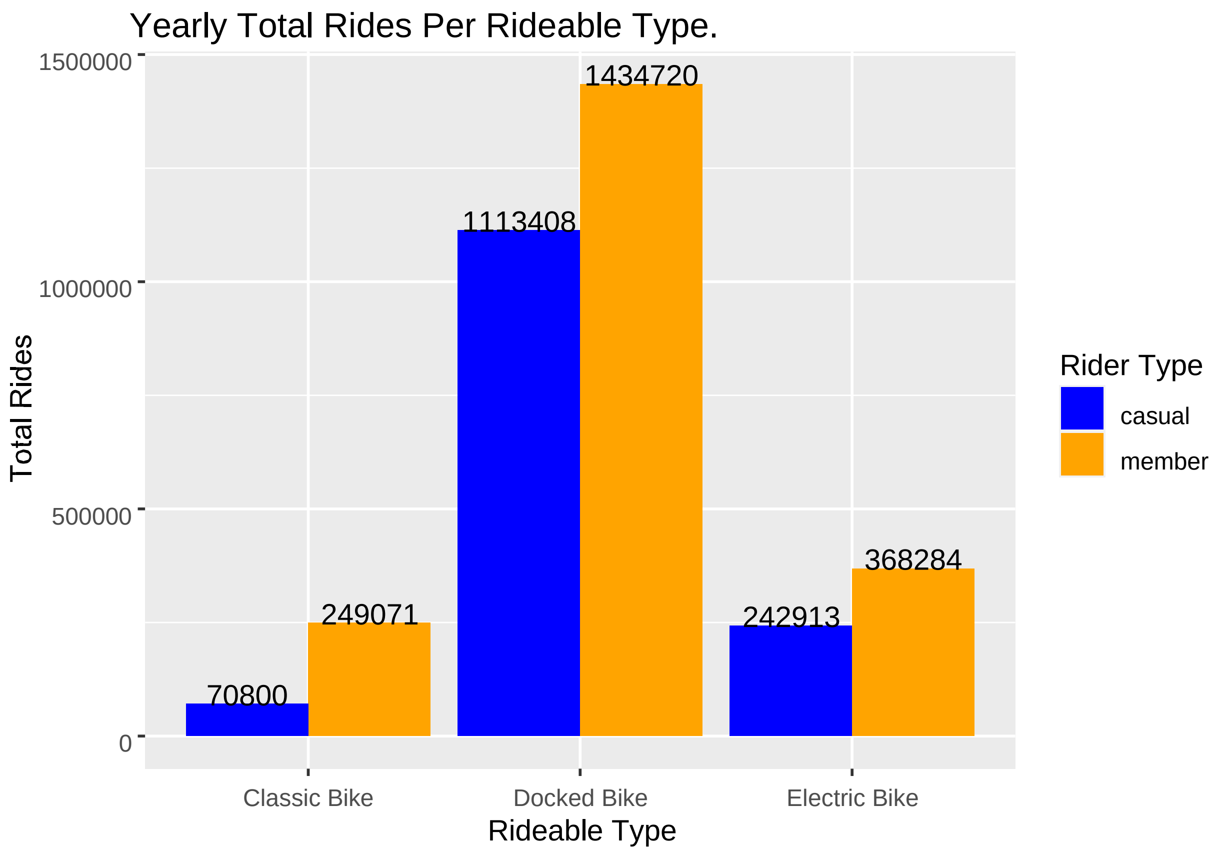
```
bike_df <- sqldf("SELECT rideable_type, member_casual, count(rideable_type) as number_of_uses
                           FROM df_1
                           GROUP BY member_casual, rideable_type
                           ORDER BY count(rideable_type) DESC", method = "auto" )
```

Changing the names of the rideable type to remove the underscore

```
bike_df$rideable_type[bike_df$rideable_type == "classic_bike"] <- "Classic Bike"
bike_df$rideable_type[bike_df$rideable_type == "docked_bike"] <- "Docked Bike"
bike_df$rideable_type[bike_df$rideable_type == "electric_bike"] <- "Electric Bike"
```

A side by side bar plot with the yearly count of rideablet for all riders

```
bike_plot <- ggplot(data = bike_df, aes(x = rideable_type, y = number_of_uses, fill = member_casual))
+ scale_fill_manual(values=c(member = 'orange', casual= 'blue')) + geom_col(position = "dodge") +
geom_text(aes(label = number_of_uses), vjust = -0.3 ,colour = "black", position
= position_dodge(.9)) +
10
labs(title = "Yearly Total Rides Per Rideable Type.", x = "Rideable Type", y
= "Total Rides", fill = "Rider Type") +
scale_y_continuous(labels = function(x) format(x, scientific = FALSE))
bike_plot
```



#END