Capstone_project

Author : Joy

Last Updated : 2022-09-02

Project Overview

This whole project is based on a bike sharing company , where we will analyze 12 months riding data set in order to answer some business questions and to provide some recommendations based on our analysis.

The Scenario

Cyclistic launched a bike sharing service at Chicago in 2016. Since then, the program has grown to a fleet of 5,824 bicycles that are geotracked and locked into a network of 692 stations across Chicago. Customers who purchase single-ride or full-day passes are referred to as casual riders. Customers who purchase annual memberships are Cyclistic members. Our stakeholder concluded that annual members are much more profitable than casual riders.Although the pricing flexibility helps Cyclistic attract more customers, our stakeholder believes that maximizing the number of annual members will be key to future growth. rather then creating marketing campeins to attract new customers , our stakeholder belives that maximizing annual members will be key to future growth.

Business Taks

How can we convert “Casual” riders to annual “Members”

Steps

We will follow the steps what we’ve learned so far .

The **Steps are** :

- **Ask**
- **Prepare**
- **Process**
- **Analyze**
- **Share**
- **Act**

Since we’ve completed **“Ask”** phase by knowing who is our stakeholder and what questions we have to answer, We will continue from our **“Prepare Phase”**

Prepare

- **In this phase we will know about our data.**
- **Then we’ll import the data from source.**
- **After import we will organize those data sets in order to keep everything concise and readable.**

Download the data from here (https://divvy-tripdata.s3.amazonaws.com/index.html)

After download

- **First create a seperate folder for only this project.**
- **Name the folder .**
- **Move downloaded data sets to the Folder.**
- **Rename data sets to identify clearly**

Since we have to work with 12 months data sets which are separated in 12 csv files

We will combine those 12 data sets into **“One” Data Frame!**

But first we need to import them

First we need to install required package to read our csv files.

```
install.packages('tidyverse')
```

After installing the package we will load it up.

```
library(tidyverse)  #Loading the package using library function.
```

Tidyverse is the most popular package to work with data related field . It comes with lots of functionality.

Now we will read those csv files using `read_csv` function included in “Tidyverse” package.

Reading a csv file

```
csv_data = read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202207-divvy-tripdata.csv") # Read csv
```

Since we have 12 csv files , we are going to read them as list items to combine them easily.

```
datas = list(read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202207-divvy-tripdata.csv"),
         read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202206-divvy-tripdata.csv"),
         read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202205-divvy-tripdata.csv"),
         read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202204-divvy-tripdata.csv"),
         read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202203-divvy-tripdata.csv"),
         read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202202-divvy-tripdata.csv"),
         read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202201-divvy-tripdata.csv"),
         read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202112-divvy-tripdata.csv"),
         read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202111-divvy-tripdata.csv"),
         read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202110-divvy-tripdata.csv"),
         read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202109-divvy-tripdata.csv"),
         read_csv("/Users/joy/Desktop/data_analysis/case_study_1_data/202108-divvy-tripdata.csv")
)
```

Now we will combine this 12 csv files into one using `bind_rows` included with “Tidyverse”

```
combined_df = bind_rows(datas) # Combining all csv files into one data frame
```

That’s it we are almost done with our **“Prepare”** phase . We have organized our data. Now we will start our **“Process”** part.

Process

In this phase of our data analysis we will clean our data and will add some new fields to work with Analyze part.

Now we will add `distance traveled` column to calculate how much a rider has traveled in one trip

To use distHaversine for calculation, first we need to install and load package called `geosphere`

```
install.packages('geosphere')
```

```
library(geosphere) # loading geosphere package
```

Now we can add our new column

```
cleaned_data = combined_df %>%
  mutate(distance_traveled = distHaversine(cbind(start_lng, start_lat), cbind(end_lng, end_lat)))
```

Haversine is a formula which calculates distance from one location to another location using longitude and latitude.

Notice that we are using a symbol `%>%` . This symbol enable us to combine multiple statements on a single block, which is pretty handy.

We can check if the column was added or not using `colnames(data_frame)` function

```
colnames(cleaned_data)
```

```
##  [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"      "distance_traveled"
```

New column was added successfully

Now let's check if there is any negative value in our new column, we will sort the data by distance traveled

```
head(select(arrange(cleaned_data, distance_traveled), distance_traveled), n=10)
```

```
## # A tibble: 10 × 1
##    distance_traveled
##                <dbl>
##  1                 0
##  2                 0
##  3                 0
##  4                 0
##  5                 0
##  6                 0
##  7                 0
##  8                 0
##  9                 0
## 10                 0
```

There is no negative values but we can see lot of 0 values , which indicates wrong data.

Hence we will filter them out for our analysis.

We can use filter function for this task

```
filter(distance_traveled > 0) # filtering by distance traveled  > 0
```

In code

```
cleaned_data = combined_df %>%
  mutate(distance_traveled = distHaversine(cbind(start_lng, start_lat), cbind(end_lng, end_lat))) %>%
  filter(distance_traveled > 0)
```

Now we'll check again for value with 0

```
head(select(arrange(cleaned_data, distance_traveled), distance_traveled), n=10)
```

```
## # A tibble: 10 × 1
##    distance_traveled
##                <dbl>
##  1            0.0231
##  2            0.0332
##  3            0.0333
##  4            0.0333
##  5            0.0333
##  6            0.0368
##  7            0.0396
##  8            0.0401
##  9            0.0408
## 10            0.0414
```

There we go. We've filtered values with 0.

Now that we know the distance traveled by a rider. We are also interested in figuring out how much time it took to complete a travel

For this we will `subtract` ended_at with started_at using `difftime` base function.

Example

```
  duration_in_min = as.double(difftime(ended_at, started_at, units = 'mins'))
```

We will calculate duration in minutes , using piping for this task

```
cleaned_data = combined_df %>%
  mutate(distance_traveled = distHaversine(cbind(start_lng, start_lat), cbind(end_lng, end_lat))) %>%
  filter(distance_traveled > 0) %>% # filtering distance traveled
  mutate(duration_in_min = as.double(difftime(ended_at, started_at, units = 'mins'))) # calculating and adding d
uration_in_min column into data frame
```

Now let us check if there is any wrong data. Which shouldn't be there

```
head(select(arrange(cleaned_data, duration_in_min), duration_in_min), n=10)
```

```
## # A tibble: 10 × 1
##    duration_in_min
##              <dbl>
##  1          -130.
##  2          -129.
##  3          -126.
##  4          -126.
##  5          -117.
##  6          -101.
##  7           -58.0
##  8           -55.9
##  9           -54.1
## 10           -53.8
```

It seems that we have lot of wrong `date data` which is not right. We will filter this negative values too.

We will use filter function again for this task

```
filter(duration_in_min > 0) # filtering by duration in min  > 0
```

In code

```
cleaned_data = combined_df %>%
  mutate(distance_traveled = distHaversine(cbind(start_lng, start_lat), cbind(end_lng, end_lat))) %>%
  filter(distance_traveled > 0) %>%
  mutate(duration_in_min = as.double(difftime(ended_at, started_at, units = 'mins'))) %>%
  filter(duration_in_min>0)  # Filtering data frame if duration in min has a negetive value
```

Now let us check again if there's still any wrong data

```
head(select(arrange(cleaned_data, duration_in_min), duration_in_min), n=10)
```

```
## # A tibble: 10 × 1
##    duration_in_min
##              <dbl>
##  1          0.0167
##  2          0.0167
##  3          0.0167
##  4          0.0167
##  5          0.0167
##  6          0.0167
##  7          0.0167
##  8          0.0167
##  9          0.0167
## 10          0.0167
```

Great we've removed wrong data.

Since our data set is too large there is a chance that lot of `Null values` will be in it

Let's check how many Null values are in our data set

```
sum(is.na(cleaned_data))
```

```
## [1] 3054590
```

Looks like we have tons of missing values. Although we will take care of this values later , but for now fill them with 'missing_value'

To fill out we will use `replace(is.na(dataframe), 'missing_value')`

In code

```
cleaned_data = combined_df %>%
  mutate(distance_traveled = distHaversine(cbind(start_lng, start_lat), cbind(end_lng, end_lat))) %>%
  filter(distance_traveled > 0) %>%
  mutate(duration_in_min = as.double(difftime(ended_at, started_at, units = 'mins'))) %>%
  filter(duration_in_min>0) %>%
  replace(is.na(.), 'missing_value') # Replacing null values with missing_value parameter
```

Now that we have a cleaned dataframe we're ready to analyze it. Before starting analyze we will arrange the whole data frame by starting date in ascending order

```
cleaned_data = combined_df %>%
  mutate(distance_traveled = distHaversine(cbind(start_lng, start_lat), cbind(end_lng, end_lat))) %>%
  filter(distance_traveled > 0) %>%
  mutate(duration_in_min = as.double(difftime(ended_at, started_at, units = 'mins'))) %>%
  filter(duration_in_min>0) %>%
  replace(is.na(.), 'missing_value') %>%
  arrange(started_at) # Arranging data by starting date in ascending order
```

According to our starting date , we need another column containing weekday name for further analysis

Adding `day_of_week` column to data frame according to starting date

```
cleaned_data$day_of_week <- weekdays(as.Date(cleaned_data$started_at))
cleaned_data$day_of_week <- factor(cleaned_data$day_of_week, levels=c('Saturday','Sunday', 'Monday', 'Tuesday',
'Wednesday', 'Thursday', 'Friday'))
```

Now let's take a sneak peek to our current data

```
head(cleaned_data, n=10)
```

```
## # A tibble: 10 × 16
##    ride_id       ridea…¹ started_at          ended_at            start…² start…³
##    <chr>         <chr>   <dttm>              <dttm>              <chr>   <chr>
##  1 D6AC43863387… classi… 2021-08-01 00:00:04 2021-08-01 00:13:28 Clark … TA1305…
##  2 00015E7CD37F… classi… 2021-08-01 00:00:17 2021-08-01 00:05:10 Racine… TA1306…
##  3 3B9554C39413… classi… 2021-08-01 00:00:19 2021-08-01 00:05:54 Wells … TA1306…
##  4 3F35D7C35A1D… classi… 2021-08-01 00:00:27 2021-08-01 00:12:17 Halste… 13192
##  5 A39DD0D9AD71… classi… 2021-08-01 00:00:32 2021-08-01 00:12:06 Broadw… 13325
##  6 46C654999C4B… classi… 2021-08-01 00:00:37 2021-08-01 00:21:03 Wells … KA1504…
##  7 0D6C68A027BA… classi… 2021-08-01 00:00:38 2021-08-01 00:18:06 Califo… 15642
##  8 105738879E10… electr… 2021-08-01 00:00:47 2021-08-01 00:06:34 missin… missin…
##  9 61DB513996B2… electr… 2021-08-01 00:00:47 2021-08-01 00:14:00 Halste… TA1309…
## 10 03D890D86770… electr… 2021-08-01 00:00:55 2021-08-01 00:24:34 Sherid… RP-009
## # … with 10 more variables: end_station_name <chr>, end_station_id <chr>,
## #   start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, end_lng <dbl>,
## #   member_casual <chr>, distance_traveled <dbl>, duration_in_min <dbl>,
## #   day_of_week <fct>, and abbreviated variable names ¹rideable_type,
## #   ²start_station_name, ³start_station_id
## # ℹ Use `colnames()` to see all variable names
```

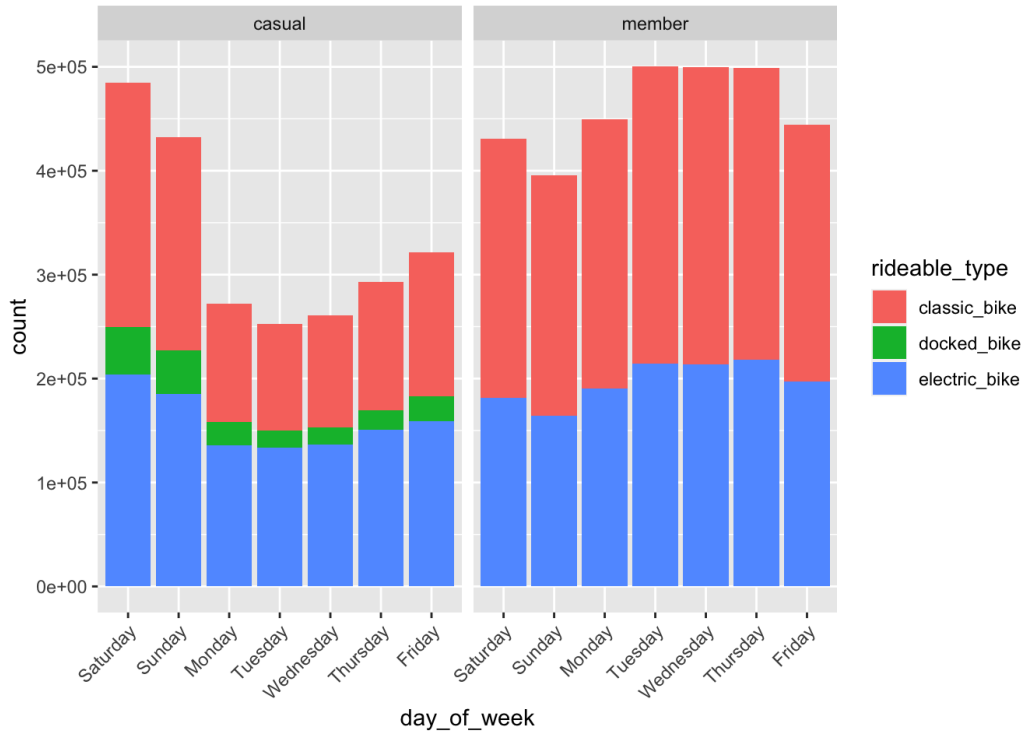
Heads up, we are ready for our "**Analyze**" part

Analyze

In this part of analysis we will discover and find some key findings to answer our business question and to provide recommendations.

We will use ggplot2 for our basic visualization

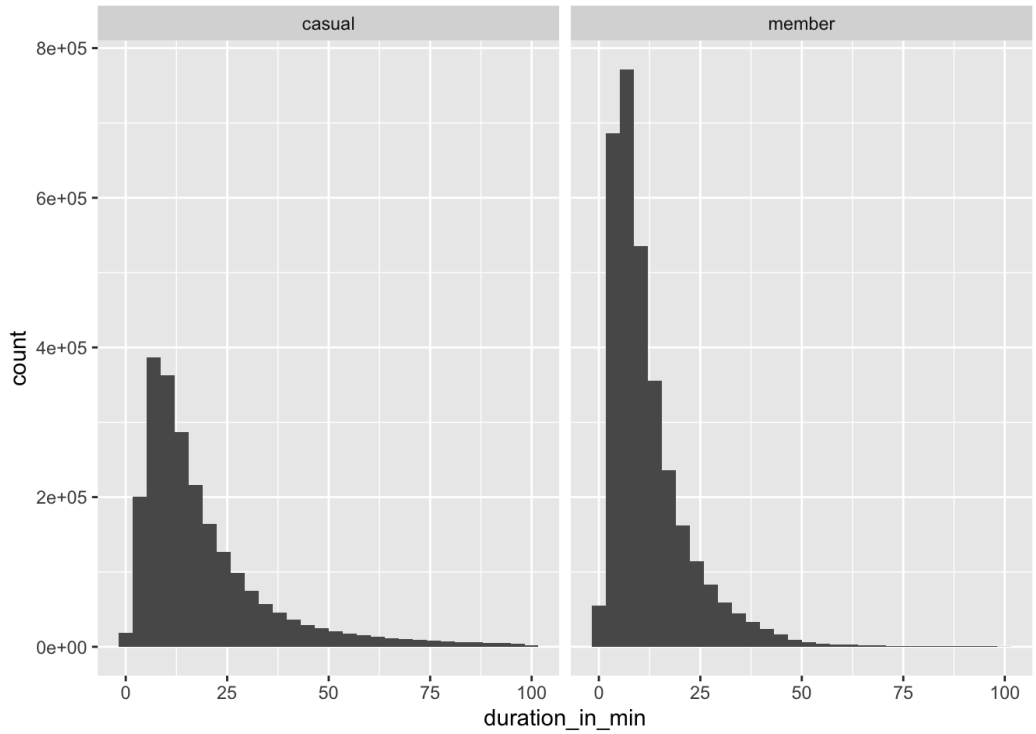
First we will check out weekly frequency distributions.



From this visualization we can observe some key points

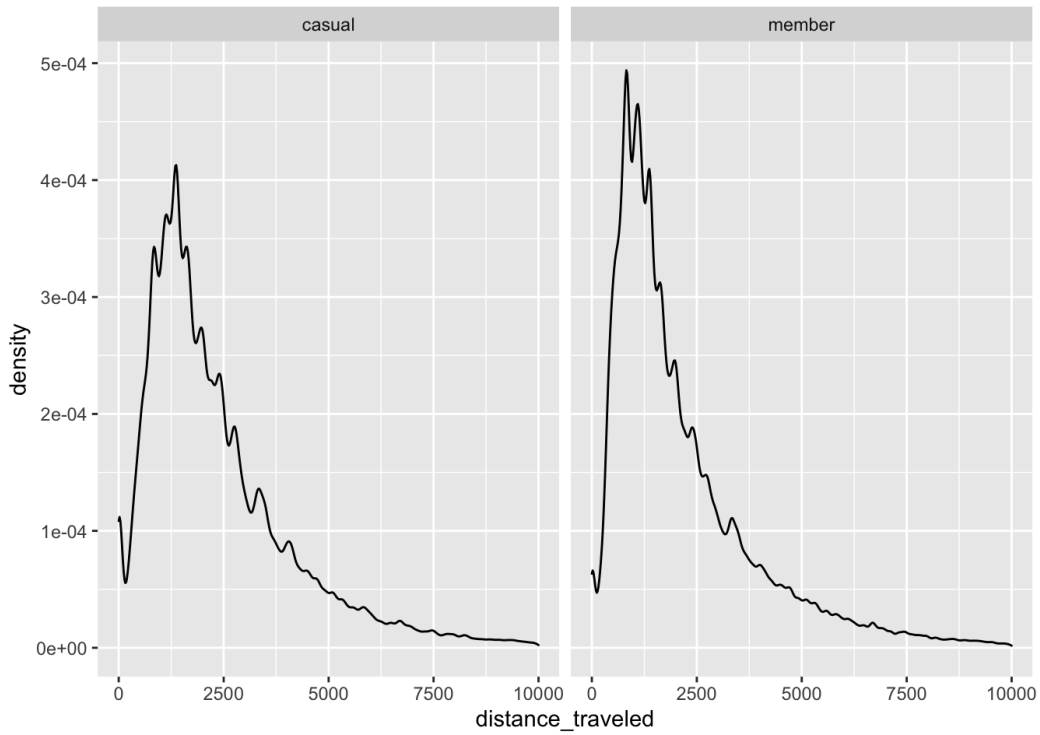
- **Members bike usage are quite similar throughout the week except Sunday , which bit slower. Which means most of the members are worker or employees**
- **For Casual riders, whole week frequency is bit slower except Saturday and Sunday , from this we can say that casual riders usage increases during weekends.**
-
- **Members likes using classic and electric bikes most same as Casual riders.**
- **Members are not interested in using docket bike**

Let's check out duration frequency too using histogram chart



We can see members tends to take shorter trip than Casual riders. Or casuals take longer trips than members. We will talk about the mean trip duration later using summary

Now lets check distance traveled by meters in density chart



It's pretty hard to observe anything from this charts, instead we will summarize the data.

Before summarizing we will divide member and casual separately

```
members = filter(cleaned_data, cleaned_data$member_casual == 'member')
casual = filter(cleaned_data, cleaned_data$member_casual == 'casual')
```

Summary data of distance traveled , and duration traveled

```
# member summary


summary(drop_na(select(members, c('day_of_week', 'distance_traveled', 'duration_in_min'))))
```

```
##     day_of_week     distance_traveled  duration_in_min
##  Saturday :430925   Min.   :    0.02   Min.   :   0.0167
##  Sunday   :395848   1st Qu.:  950.91   1st Qu.:   5.4333
##  Monday   :449590   Median : 1587.75   Median :   9.1500
##  Tuesday  :500336   Mean   : 2195.85   Mean   :  12.7033
##  Wednesday:499598   3rd Qu.: 2807.91   3rd Qu.:  15.6500
##  Thursday :499180   Max.   :42319.46   Max.   :1499.9333
##  Friday   :444595
```

```
# casual summary

summary(drop_na(select(casual, c('day_of_week', 'distance_traveled', 'duration_in_min'))))
```

```
##     day_of_week     distance_traveled duration_in_min
##  Saturday :484975   Min.   :      0   Min.   :    0.02
##  Sunday   :432557   1st Qu.:   1141   1st Qu.:    8.40
##  Monday   :272322   Median :   1891   Median :   14.23
##  Tuesday  :252547   Mean   :   2455   Mean   :   23.62
##  Wednesday:260604   3rd Qu.:   3162   3rd Qu.:   25.12
##  Thursday :292844   Max.   :1190854   Max.   :41629.17
##  Friday   :321754
```

Summary Takeaways

- **From above summary, we can observe that mean distance traveled by members and casuals are almost same.**
- **Members mean trip duration (~12 min), which is almost 1 time less than casual mean trip duration (~23 min)**
- **And maximum distance traveled by Members (`42319m`) is `100x` time less then Casual riders which is (`1190854m`)**

Now we will find out which are most popular start and end stations for Casual and Annual Members.

Member popular start stations

```
head(count(members, start_station_name, sort=T), n=10) # Start station count
```

```
## # A tibble: 10 × 2
##    start_station_name                n
##    <chr>                         <int>
##  1 missing_value                418019
##  2 Kingsbury St & Kinzie St      26040
##  3 Wells St & Concord Ln         23028
##  4 Clark St & Elm St             22937
##  5 Wells St & Elm St             20298
##  6 Ellis Ave & 60th St           19557
##  7 Clinton St & Washington Blvd  19235
##  8 Clinton St & Madison St       19216
##  9 University Ave & 57th St      19001
## 10 Dearborn St & Erie St         18301
```

Member popular end stations

```
head(count(members, end_station_name, sort=T), n=10) # End station count
```

```
## # A tibble: 10 × 2
##    end_station_name                 n
##    <chr>                        <int>
##  1 missing_value               414925
##  2 Kingsbury St & Kinzie St     25741
##  3 Wells St & Concord Ln        23727
##  4 Clark St & Elm St            23290
##  5 Wells St & Elm St            20332
##  6 Clinton St & Washington Blvd 20059
##  7 Clinton St & Madison St      19781
##  8 University Ave & 57th St     19728
##  9 Ellis Ave & 60th St          19066
## 10 Dearborn St & Erie St        18576
```

Casual popular start stations

```
head(count(casual, start_station_name, sort=T), n=10) # Start station count
```

```
## # A tibble: 10 × 2
##    start_station_name                   n
##    <chr>                            <int>
##  1 missing_value                   319482
##  2 Streeter Dr & Grand Ave          53806
##  3 DuSable Lake Shore Dr & Monroe St  27777
##  4 Millennium Park                  25586
##  5 DuSable Lake Shore Dr & North Blvd 24966
##  6 Michigan Ave & Oak St            24128
##  7 Shedd Aquarium                   19761
##  8 Wells St & Concord Ln            18375
##  9 Theater on the Lake              18316
## 10 Wells St & Elm St                14910
```

Casual popular end stations

```
head(count(casual, end_station_name, sort=T), n=10) # End station count
```

```
## # A tibble: 10 × 2
##    end_station_name                     n
##    <chr>                            <int>
##  1 missing_value                   374870
##  2 Streeter Dr & Grand Ave          55878
##  3 DuSable Lake Shore Dr & North Blvd 28733
##  4 Millennium Park                  26627
##  5 Michigan Ave & Oak St            25528
##  6 DuSable Lake Shore Dr & Monroe St  25427
##  7 Theater on the Lake              19402
##  8 Shedd Aquarium                   18618
##  9 Wells St & Concord Ln            17601
## 10 Clark St & Lincoln Ave           14643
```

From above summary we can see that members most used start and are same . And casual most used start and end stations are also same too.

We have to concatenate start station with end stations to find out most populart starting and ending point

Members

```
# member
members$route = paste(members$start_station_name, ' ------ ', members$end_station_name)
head(count(members, route, sort=T), n=10)
```

```
## # A tibble: 10 × 2
##    route                                                  n
##    <chr>                                              <int>
##  1 missing_value  ------  missing_value                209407
##  2 Ellis Ave & 60th St  ------  Ellis Ave & 55th St      5667
##  3 Ellis Ave & 60th St  ------  University Ave & 57th St  5646
##  4 University Ave & 57th St  ------  Ellis Ave & 60th St  5485
##  5 Ellis Ave & 55th St  ------  Ellis Ave & 60th St      4983
##  6 University Ave & 57th St  ------  missing_value       2955
##  7 Ellis Ave & 60th St  ------  missing_value            2645
##  8 missing_value  ------  University Ave & 57th St      2501
##  9 Calumet Ave & 33rd St  ------  State St & 33rd St     2494
## 10 missing_value  ------  Ellis Ave & 60th St           2407
```

Casual Riders

```
# casual
casual$route = paste(casual$start_station_name, ' ------ ', casual$end_station_name)
head(count(casual, route, sort=T), n=10)
```

```
## # A tibble: 10 × 2
##    route                                                        n
##    <chr>                                                    <int>
##  1 missing_value  ------  missing_value                    177697
##  2 DuSable Lake Shore Dr & Monroe St  ------  Streeter Dr & Grand Ave    5463
##  3 Streeter Dr & Grand Ave  ------  Millennium Park          3052
##  4 Streeter Dr & Grand Ave  ------  DuSable Lake Shore Dr & Monroe St    2921
##  5 Millennium Park  ------  Streeter Dr & Grand Ave          2740
##  6 Shedd Aquarium  ------  Streeter Dr & Grand Ave          2686
##  7 Streeter Dr & Grand Ave  ------  Michigan Ave & Oak St    2484
##  8 Streeter Dr & Grand Ave  ------  missing_value            2467
##  9 Streeter Dr & Grand Ave  ------  DuSable Lake Shore Dr & North Blvd   2394
## 10 Streeter Dr & Grand Ave  ------  Streeter Dr & Grand Ave  2105
```

Now we will summarize our Analysis key points using `date.frame` function from `formattable` package

First we need install and load `formattable` package.

```
install.packages("formattable")
```

```
library(formattable) # loading package
```

It was kind a boring to type out these raw , but the output wroth it

User.Type	Amount	Avg_and_median_trip_dur	Avg_and_median_trip_distance	Busiest.Day	Preferred.bike.type	Most.occaurd.route
Member	3220072 (58.14%)	12.70 min - 9.15 min	2195.85 meters - 1587.75 meters	Tuesday	classic & electric	Ellis Ave & 60th St - Ellis Ave & 55th St (5667)
Casual	2317603 (41.86%)	23.62 min - 14.23 min	2455 meters - 1891 meters	Saturday	electric	DuSable Lake Shore Dr & Monroe St - Streeter Dr & Grand Ave (5463)

Now will try to figure out why there is so many missing values in the date

```
missing_data = drop_na(filter(cleaned_data, start_station_name=='missing_value')) # Filtering missing values
summary(select(missing_data, c('day_of_week', 'distance_traveled', 'duration_in_min'))) # Summarizing missing va
lues
```

```
##     day_of_week     distance_traveled  duration_in_min
##  Saturday :117623   Min.   :   10.56   Min.   :  0.0167
##  Sunday   :107009   1st Qu.: 1113.19   1st Qu.:  6.1167
##  Monday   : 96016   Median : 1995.56   Median : 10.5167
##  Tuesday  : 99942   Mean   : 2674.73   Mean   : 14.9585
##  Wednesday:101040   3rd Qu.: 3441.19   3rd Qu.: 18.4833
##  Thursday :108259   Max.   :30725.77   Max.   :480.5167
##  Friday   :107612
```

Checking which bike type data is missing most

```
head(count(missing_data , member_casual, rideable_type, sor=T), n=10)
```

```
## # A tibble: 2 × 4
##   member_casual rideable_type sor        n
##   <chr>         <chr>         <lgl>  <int>
## 1 casual        electric_bike TRUE  319482
## 2 member        electric_bike TRUE  418019
```

We can see that electric bikes datas are only missing.

Let's see how much electric bike data is in the dataset

```
count(filter(cleaned_data, rideable_type=='electric_bike'))
```

```
## # A tibble: 1 × 1
##         n
##     <int>
## 1 2484562
```

Almost 2.5 million

From this 2.5 million electric bikes data, almost 29.68% of it is missing starting and ending station names, which is around 750k

Key Observations // Share

All though we can't answer the business question clearly because of data , but we will point out some analysis

- **Members bike usage is quite similar throughout the week except sunday which indicates that most of them are employees or workers . And the busiest days are Tuesday, Wednesday and Thursday.**
- **Casual riders bike usage is pretty slow except Saturday and Sunday which indicates casual riders usage increases during weekends.**
- **Among members most preferred bike is classic and electric bikes . And among casual riders most preferred bikes are same as members . This derives to a conclusion that both casual and members likes using electric and classic bikes most.**
- **Average distance traveled by members and causal are almost same . However members avg trip duration is 12 minutes, which is 2 time less then casual riders.**
- **In most cases both members and casual riders start and end stations are same .**
- **Most lengthy trips were taken by causal riders and they are abnormally long . For instance top five of them are respectively 28, 23, 22, 20 and 19 days long.**
- **All occurred missing data around (750k) of start and end station names occurred only for electric bikes . There are around (2.5 million) electric bikes and among them 29.68% are missing . which is almost 750k.**

Business Answer and Recommendations // Act

Considering the above observations and insights we can suggest the following:

- **We see that members take shorter trips to work with bikes during Monday to Saturday, since it is financially viable and fast transportation. However, casuals prefer longer trips especially Saturday and Sunday. Thus, We could increase the renting price of the bikes for the weekend to target casual users into having a membership especially for electric bikes, since they are preferred more by causal users.**
- **Providing a special service or perks for only members might motivate casual users to have a membership. Services might include free ice cream or lemonade, free tour guide, or fast line for renting without any line etc.**
- **Also, since we know the most popular start station names and routes for casual users, we can put banners or special discount advertisements in those areas or routes that would target casual users.**
- **Furthermore, all missing start and end station names occurred with electric bikes. We have to learn why that is the case and fix the infrastructure if necessary.**

Saving files and datas

We have a cleaned data set to save for this we will use `fwrite` from `data.table` package for faster file writing to save our csv file

First install `data.table` pacakge

```
  install.pacakges("data.table")
```

Then load it

```
library(data.table) # loading package
```

After loading the package , we can write the file like below

```
# writing csv file with fwrite from data.table
fwrite(cleaned_data, '/Users/joy/Desktop/data_analysis/case_study_1_c_data/cleaned_data.csv')
```

Conclusion

After completing Analyzing , it took lot of effort to make this whole R mark down report . The intersting part is that I've made this R Mark Down only for html produce able format. It's kinda crazy idea to embedded html inside a R Mark Down Report ;)

Thank You