Google Data Analytics Certificate - Case Study

By: Alem Haddush Fitwi

91 Endicott Ave, Johnson City, NY 13790
Mobile Phone Number: (607) 232-1056
E-mail: ahfitwi@gmail.com/afitwi1@binghamton.edu

Summary:

A seasoned Engineer with a wealth of experience in Electrical & Computer Engineering projects, Computer Networking and Security, Software Development and management, Business Intelligence, Data Analytics, Data Science and Machine Learning/Deep Learning. I tend more toward Data Science or Data Analytics using Python3, R, Machine Learning and Apache Spark for cluster or parallel computing! Besides, I am very good at fullstack web and Dashboard development using HTML, CSS, & JavaScript for frontend coding, Python3 for backend coding, & MySQL/ORacleDB/PSQL for database management.

Professional & Project Experience:

Project Research Assistant (May 2021 - Todate)

@ SUNY Research Foundation, Binghamton University

- **Major Tasks Performed**:
  - Developed and implemented a Distributed Network Communication Algorithm for the implementation of the "Distributed Collaborative Orthogonal Matching Pursuit Thresholding Algorithm (DC-OMP-TA)" based on a hybrid of peer-to-peer and client-server architectures. It was developed through the seamless integration of Python3 based communication application and a MATLAB implementation of the DC-OMP-TA algorithm. It was successfully tested on a single PC with multiple threads of CLI running on it to emulate multiple machines and

on more than three distributed machines. A report/manual comprising more than 60 pages was generated, as well.
    - Developed social distance and crowd density estimator using Deep Learning and Python3 for Technergetics. Human Detection, Distance and Area estimation algorithms were designed, developed and implemented. Out of it, a 12-page article was submitted for publication. The entire documentation is provided on my Github: https://github.com/ahfitwi/SDProject2021
- **Tools Employed**:
    - Python for video dataset processing, for building Deep-learning Models, for Implementing Algorithms, for network programming, and file managemnet.
    - MATLAB: for the implementation of DC-OMP-TA

Graduate Assistant-Data Analyst (August, 2017 - May, 2021):

@ Dean's Office of Watson College of Engineering & Applied Science, Binghamton University

- **Major Tasks Performed**:
    - Industry grade dashboards and database designs and developments using Python and MySQL
    - Business Intelligence (BI), and Content Analysis
    - Data Science and Analysis on such input datasets as 15-year and semester-wise Student Opinion of Teaching (SOOT), Historical and current Admission Data, and 25 years research data.
    - The size of datasets processed ranges from 1MB to 5GB.
    - The outputs of the analysis were mainly actionable data and insightful reports (in CSV, EXCEL, SAS, PDF, HTML, and RTF/WORD formats as requested) and graphical plots (line graphs, bar charts, pie charts, and scatter plots) generated using SAS, R, SQL, Python, and Machine Learning/Deep Learning algorithms.
    - Preparing Surveys on Qualtrics, Performing Descriptive and Inferential Statistical Analysis using Python3 and R programming Languages, Testing Hypothesis, and Determining confidence Intervals.
- **Tools employed**:
    - SAS for data analysis, visualization, generating aesthetic reports, and statistical analysis

- Python3 for backend development, dashboard development using dash and plotly, data analysis, machine learning based model building, & content analysis
- SQLite/MySQL/ Oracle SQL for designing, creating, & developing Databases and for creating queries and subqueries.
- CSS/HTML/js for the frond development of a Dashboard
- Microsoft Excel: For cleaning and analyzing small datasets and generating visualization
- Google Sheets: For cleaning and analyzing small datasets and creating visualization.
- R for statistical analysis and visualization

Director of Information & Communication Technology Directorate (October 2012 - October 2015):

`@ Mekelle University, Mekelle, Tigrai, Ethiopia`

- Datacenter: managed to build modern, tier-III datacenter for Mekelle University
  - Expert at Designing Datacenter infrastructures, ERP/CAN Networks, & Configurations
- University ERP: managed to get the Mekelle University business processes automated
  - Backend: Ruby On Rails, DB: MySQL, & Frondend: HTML, CSS, & Java Script

Mega Wind Farm Projects (Ashegoda and Adama-II):

- successfully took part in two mega wind farm projects as Electrical & SCADA Engineer in the design, supervision, and commissioning processes.
- successfully worked as Electrical Engineer (September, 2012 to June, 2014) in a 120 MW Wind Farm project (Ashegoda, Mekelle, Ethiopia) and as a SCADA Engineer (July 01, 2013 to December 15, 2015) in another 153 MW Wind Farm Project (Adama, Ethiopia).

Teaching Career (September 2006 - August 2017):

`@ Mekelle University, Mekelle, Tigrai, Ethiopia`

- Worked as Graduate Assistant II, Assistant Lecturer, Lecturer, & Assistant professor.

- Major Tasks: Teaching Computer Engineering Courses, Conducting Lab activitis, carrying out research, giving community services, and software development.

Research:

- Areas: Privacy and Cyber Security, Video Surveillance, Lightweight Machine Learning/Deep-learnig, Edge/Fog/Cloud Computing, Network Architecture, Distributed Computing, and IoTs
- Tools & Hardwares: Statistical Methods, Machine/Deep Learning Models, Scrambling/Encryption Schemes, Python3, C++, SQLite, SQL, Edge and cloud servers, vSphere-based virtual machines, and Edge Devices (Raspberry PI 4, Cameras, Sensors, ...)
- Publications: 20 articles and conference papers and a bookchapter have been published to date, not to mention th emany unplished teaching, training, and project manuals, reports and documents.

Leadership:

- **Information & Communication Technology Director** (October, 2012 - October 26, 2015): managed the ICT directorate of Mekelle University (Ethiopia) for 3 years, responsible for running the entire university (has a population of more than 35, 000 students and staff) campus area network and ERP. The directorate had 6 departments and 106 IT staff members of various professions.
- **Head and Program Coordinator** (2010 - 2012): Head of the Computer Engineering chair (11 staff members), and program coordinator of the department of Electrical and Computer Engineering (79 staff members), Mekelle University, for one year each.

Education:

- Watson School of Engineering, Binghamton University, State University of New York (Binghamton, NY, USA)
  - PhD in Computer Engineering, Expected: December 2021
- Addis Ababa University (Addis Ababa, Ethiopia; P.O.Box 1176)
  - M.Sc. in Computer Engigneering, Earned: September 2010
- Bahir Dar University (Bahir Dar, Ethiopia; P.O.Box 79)
  - B.Sc in Electrical & Computer Engineering, Earned: July 06, 2006

Technical and Programming Skills:

1. **Excellent Data Processing Skill Set**: Super good in creating surveys on Qualtrics, Data Processing, Analyzing, Hypothesis Testing, and genrating repots helpful for informed decision making.
2. **Advanced Python Programming + SQL + Machine Learning**: For both academic and professional purposes since 2014, mainly for Data Science and Analysis (Numpy, Pandas, Scipy, Scikit Learn, Seaborn, Matplotlib, Plotly and Dash), for Deep Learning (Lightweight DNN for Conserving Privacy in Edge Devices as part of my PhD research), Network and Network Security Programming (Device Configurations, Extensive String Manipulations using RE, DNS Activity Monitoring, and Network Scan Detection projects), and industry grade dashboard development.
3. **R Programming**: Very good in R for data cleaning, organizing, analyzing, and static and dynamic(dashboard) visualizing using a bunch of packages (tidyverse, dplyr, ggplot2, lubridate, data.table, tidyr, shiny, plotly, mlr3, ...)
4. **Fullstack Web Development**: Very good at fullstack web development using HTML, CSS, & JavaScript for frontend coding, Python for backend coding, and MySQL (or Oracle DB, MangoDB, etc) for databases.
5. **SAS Programming + SQL**: For data analysis (data preprocessing, transforming, mining, visualization, and reporting in xlsx, pdf, & rtf formats). I have used it since August 2017.
6. **Spreadsheet or googlesheets**: for data cleaning, analysis, visualization, and reporting
7. **Network Administration & Security**: Very good expertise and experience in IT Networks and Operational Networks (SCADA) design, installation, and administration. I had worked as Network Admin, SCADA Engineer, & ICT Director, and was trained and certified in Network Administration, Data Center Infrastructures design and construction, and Network Security. I am also very good in network programming using Python.
8. **Other Programming Skills**:
    - **Java**: I used to be an expert in it and I developed many desktop and web apps using it. But I haven't used it since 2013.
    - **PHP**: for backend web app development. I last used it in 2015.
    - **C++**: I learnt this languange long ago. I still occasionally use it for research, in areas where speed is of great essence.
    - **VHDL**: I had frequently used it for academic and research purpose until 2015. Most recent use was in 2018 for a VLSI class.

- **C**: the first programming language I learnt as a young boy. I was very good at it but I have n't used it for a while.

Case Study: How Does a Bike-Share Navigate Speedy Success?

Introduction

Welcome to the Cyclistic bike-share analysis case study! In this case study, I will perform many real-world tasks of a data analyst. I will work for a fictional company, Cyclistic, and meet different characters and team members. In order to answer the key business questions, I will follow the steps of the data analysis process: **ask, prepare, process, analyze, share, and act**. I will borrow Google Analytics Certificate Course's icons/pctures to represent each of these steps in what ensues.

Scenario

I am data analyst working in the marketing analyst team at Cyclistic, a bike-share company in Chicago. The director of marketing believes the company’s future success depends on maximizing the number of annual memberships. Therefore, my team wants to understand how **casual riders and annual members** use Cyclistic bikes differently. From these insights, my team will design a new marketing strategy to convert casual riders into annual members. But first, Cyclistic executives must approve our recommendations, so they must be backed up with compelling data insights and professional data visualizations.

Characters and teams

- **Cyclistic**: A bike-share program that features more than 5,800 bicycles and 600 docking stations. Cyclistic sets itself apart by also offering reclining bikes, hand tricycles, and cargo bikes, making bike-share more inclusive to people with disabilities and riders who can’t use a standard two-wheeled bike. The majority of riders opt for traditional bikes; about 8% of riders use the

assistive options. Cyclistic users are more likely to ride for leisure, but about 30% use them to commute to work each day.

- **Lily Moreno**: The director of marketing and my manager. Moreno is responsible for the development of campaigns and initiatives to promote the bike-share program. These may include email, social media, and other channels.
- **Cyclistic marketing analytics team**: A team of data analysts who are responsible for collecting, analyzing, and reporting data that helps guide Cyclistic marketing strategy. I joined this team six months ago and have been busy learning about Cyclistic's mission and business goals — as well as how I, as a data analyst, can help Cyclistic achieve them.
- **Cyclistic executive team**: The notoriously detail-oriented executive team will decide whether to approve the recommended marketing program.

About the company

In 2016, Cyclistic launched a successful bike-share offering. 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. The bikes can be unlocked from one station and returned to any other station in the system anytime.

Until now, Cyclistic's marketing strategy relied on building general awareness and appealing to broad consumer segments. One approach that helped make these things possible was the flexibility of its pricing plans: single-ride passes, full-day passes, and annual memberships. Customers who purchase single-ride or full-day passes are referred to as casual riders. Customerswho purchase annual memberships are Cyclistic members.

Cyclistic's finance analysts have concluded that annual members are much more profitable than casual riders. Although the pricing flexibility helps Cyclistic attract more customers, Moreno believes that maximizing the number of annual members will be key to future growth. Rather than creating a marketing campaign that targets all-new customers, Moreno believes there is a very good chance to convert casual riders into members. She notes that casual riders are already aware of the Cyclistic program and have chosen Cyclistic for their mobility needs.

Moreno has set a clear goal: Design marketing strategies aimed at converting casual riders into annual members. In order to do that, however, the marketing analyst team needs to better understand how annual members and casual riders

differ, why casual riders would buy a membership, and how digital media could affect their marketing tactics. Moreno and her team are interested in analyzing the Cyclistic historical bike trip data to identify trends.

Step-1: ASK

Case Study Roadmap - Ask

Guiding questions

Three questions will guide the future marketing program:

1. How do annual members and casual riders use Cyclistic bikes differently?
2. Why would casual riders buy Cyclistic annual memberships?
3. How can Cyclistic use digital media to influence casual riders to become members?

My Assignment

Moreno has assigned me the first question to answer:

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

Hypothesis

- Annual members are much more profitable than casual riders; hence, converting casual riders into annual members would increase the company's profit.

Deliverable

Then, I will produce a report with the following deliverables:

1. A clear statement of the business task
    - How do annual members and casual riders use Cyclistic bikes differently?
        - Number of casual vs annual member riders?

    - Averages of casual and member riders in terms of daily riding duration?
    - What is the monthly average of riding durations for casual and annual-member riders?
    - What are the correlations between seasons and number of rides of each group?
    - what are the correlations between seasons and riding duration of each group?
    - Overall correlations between seasons and riding durations?
2. A description of all data sources used
3. Documentation of any cleaning or manipulation of data
4. A summary of my analysis
5. Supporting visualizations and key findings
6. My top three recommendations based on my analysis

Step-2: PREPARE

Case Study Roadmap - Prepare

Guiding questions and answers

- Where is my data located?
    - Source URL: https://divvy-tripdata.s3.amazonaws.com/index.html
- How is the data organized?
    - I am using the last 12 months data of biking and they are stored on the site month wise. After downloading the most recent 12 month data (October, 2020 to September, 2021), they are mereged into into a single file, named "bike_data_Oct2021_to_Sept2021_v01.csv". It is a structured data that comprises 5,136,261 rows and 14 columns.
- Are there issues with bias or credibility in this data? Does your data ROCCC?
    - There is sufficiently large data which is reliable (R) (proven fit for use and unbiased), original (O), comprehensive (C) (contains all critical information needed to answer the questions raised), current (C) and relevant to the task, and cited ()C by many.
    - Hence, yes the data I have at hand ROCCCs!
- How am I addressing licensing, privacy, security, and accessibility?

    - Accessible with clearly declared use policy.
    - The data has been made available by Motivate International Inc. under the license described in
        - https://www.divvybikes.com/data-license-agreement
- How did I verify the data's integrity?
    - By verifying its overall accuracy, completeness, and reliability
- How does it help me answer my question?
    - It contains information about the riding behavior of casual and annual member riders and it definitely helps us answer the question of how casual riders can be converted into member riders to make the company more profitable.
- Are there any problems with the data?
    - On the dataset, the "start_station_id" and "end_station_id" values are deliberately anonymized for the privacy of riders; hence, analysis based on this attribute cannot be made.
    - I also observed inconsistent station names and data-types (especially dates), missing stations, and inscosistent naming under the rideable_type ('electric_bike', 'classic_bike', and 'docked_bike'). The 'classic_bike', and 'docked_bike' mean the same thing. In older data (beyond December, 2020), it is writtent as "docked_bike; however, this has been changed to "classic_bike" starting December, 2020.
    - Cleaning, filtering, and column renaming are performed at the "Process" phase, just in the next step.

Key tasks

1. Download data and store it appropriately.
    - 12 datasets (October 2020 to September 2021) were downloaded and merged into a single big dataset, named "bike_data_Oct2021_to_Sept2021_v01.csv"
2. Identify how it's organized.
    - Structured and comprises 5,136,261 rows and 14 columns
    - Renamed as "bike_data_Oct2021_to_Sept2021_v01.csv"
3. Sort and filter the data.
    - Unnecessary columns like start_lat, start_lng, end lat, and end_lng were filtered out!
4. Determine the credibility of the data.
    - Overall reliable, original, comprhensive, consistent, and cited datasets they are

Deliverable

- A description of all data sources used
  - Done

R packages Neccessary for this project are Imported here

- install.packages("package_name")
- packageVersion("package_name")
- remove.packages("package_name")

```
options(warn = - 1)

library(tidyverse)
library(lubridate)
library(ggplot2)
library(dplyr)
library(here)
library(skimr)
library(janitor)
library(Dict)
library(egg)
library("IRdisplay")

print(paste("tidyverse package version: ",
        packageVersion("tidyverse")))
print(paste("lubridate package version: ",
        packageVersion("lubridate")))
print(paste("ggplot2 package version: ",  packageVersion("ggplot2")))
print(paste("dplyr package version: ",  packageVersion("dplyr")))
print(paste("here package version: ",  packageVersion("here")))
print(paste("skimr package version: ",  packageVersion("skimr")))
print(paste("janitor package version: ",  packageVersion("janitor")))
print(paste("Dict package version: ",  packageVersion("Dict")))
print(paste("egg package version: ",  packageVersion("egg")))
```

```
[1] "tidyverse package version:  1.2.1"
[1] "lubridate package version:  1.7.4"
[1] "ggplot2 package version:  3.1.1"
[1] "dplyr package version:  1.0.6"
[1] "here package version:  1.0.1"
[1] "skimr package version:  2.1.3"
[1] "janitor package version:  2.1.0"
```

```
[1] "Dict package version:  0.1.0"
[1] "egg package version:  0.4.5"
```

A python code for putting the last 12 months' bike datasets together

- Only the data downloading and merging steps were done using python3.9
- The rest of the project was processed using R-programming

```
def get_yearMonth(path, str1):
    """ Grabs Year & Month from
        csv names"""
    str1 = str1.replace(path+'/','')
    str1 = str1.split("-")
    str1 = str1[0]
    str1 = str1[:4] + '-' + str1[4:]
    dt = datetime.datetime.strptime(str1, "%Y-%m")
    if len(str(dt.month)) == 1:
        month = '0'+str(dt.month)
    else:
        month = str(dt.month)
    return str(dt.year)+'-'+ month


def merge_bikeDatasets(path):
    """Merges bike data of the
    last 12 mongths into one."""
    try:
        path1 = os.path.join(path, "*.csv")
        files =  glob.glob(path1)
        df = pd.read_csv(files[0])
        df["month"] = get_yearMonth(files[0])
        files1 = files[1:]
        for file in files1:
            tmp = pd.read_csv(file)
            tmp["month"] = get_yearMonth(file)
            df = df.append(tmp, ignore_index=True)
    except:
        return False
```

```
    else:
        return df
```

Merged bike_dataset, read it in using R

- 5,136,261 rows and 14 columns
- The column named "month" was added during the merging process
- Before merging, the datasets appeared like what ensues:

```
['202101-divvy-tripdata.csv',
 '202010-divvy-tripdata.csv',
 '202109-divvy-tripdata.csv',
 '202012-divvy-tripdata.csv',
 '202108-divvy-tripdata.csv',
 '202106-divvy-tripdata.csv',
 '202102-divvy-tripdata.csv',
 '202104-divvy-tripdata.csv',
 '202011-divvy-tripdata.csv',
 '202107-divvy-tripdata.csv',
 '202103-divvy-tripdata.csv',
 '202105-divvy-tripdata.csv']
```

Reading Dataset

```
# Path
path <- "C:\\Users\\afitwi1\\Desktop\\googleproj\\dataset\\"
file <- "bike_data_Oct2021_to_Sept2021_v01.csv"
path <- paste(path, file, sep='')

# Dataframe Created from csv file just read in!
bike_df <- read_csv(path)

Parsed with column specification:
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_character(),
```

```
  end_station_name = col_character(),
  end_station_id = col_character(),
  start_lat = col_double(),
  start_lng = col_double(),
  end_lat = col_double(),
  end_lng = col_double(),
  member_casual = col_character(),
  month = col_character()
)
```

Dimensions: Rows and Columns

```
dim(bike_df)
```

1. 5136261
2. 14

Sample Displays

```
head(bike_df)
```

ride_id	rideable_type	started_at	ended_at	start_st
E19E6F1B8D4C42ED	electric_bike	2021-01-23 16:14:19	2021-01-23 16:24:44	California Cortez St
DC88F20C2C55F27F	electric_bike	2021-01-27 18:43:08	2021-01-27 18:47:12	California Cortez St
EC45C94683FE3F27	electric_bike	2021-01-21 22:35:54	2021-01-21 22:37:14	California Cortez St
4FA453A75AE377DB	electric_bike	2021-01-07 13:31:13	2021-01-07 13:42:55	California Cortez St
BE5E8EB4E7263A0B	electric_bike	2021-01-23 02:24:02	2021-01-23 02:24:45	California Cortez St

ride_id	rideable_type	started_at	ended_at	start_st
5D8969F88C773979	electric_bike	2021-01-09 14:24:07	2021-01-09 15:17:54	California Cortez St

Columns of the Dataset

```
colnames(bike_df)
```

1. 'ride_id'
2. 'rideable_type'
3. 'started_at'
4. 'ended_at'
5. 'start_station_name'
6. 'start_station_id'
7. 'end_station_name'
8. 'end_station_id'
9. 'start_lat'
10. 'start_lng'
11. 'end_lat'
12. 'end_lng'
13. 'member_casual'
14. 'month'

Structure of the Dataset

```
str(bike_df)
```

```
spec_tbl_df [5,136,261 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ ride_id           : chr [1:5136261] "E19E6F1B8D4C42ED"
"DC88F20C2C55F27F" "EC45C94683FE3F27" "4FA453A75AE377DB" ...
 $ rideable_type     : chr [1:5136261] "electric_bike" "electric_bike"
"electric_bike" "electric_bike" ...
 $ started_at        : POSIXct[1:5136261], format: "2021-01-23
16:14:19" "2021-01-27 18:43:08" ...
 $ ended_at          : POSIXct[1:5136261], format: "2021-01-23
16:24:44" "2021-01-27 18:47:12" ...
 $ start_station_name: chr [1:5136261] "California Ave & Cortez St"
"California Ave & Cortez St" "California Ave & Cortez St" "California
Ave & Cortez St" ...
```

```
 $ start_station_id  : chr [1:5136261] "17660" "17660" "17660" "17660"
...
 $ end_station_name  : chr [1:5136261] NA NA NA NA ...
 $ end_station_id    : chr [1:5136261] NA NA NA NA ...
 $ start_lat         : num [1:5136261] 41.9 41.9 41.9 41.9 41.9 ...
 $ start_lng         : num [1:5136261] -87.7 -87.7 -87.7 -87.7 -87.7
...
 $ end_lat           : num [1:5136261] 41.9 41.9 41.9 41.9 41.9 ...
 $ end_lng           : num [1:5136261] -87.7 -87.7 -87.7 -87.7 -87.7
...
 $ member_casual     : chr [1:5136261] "member" "member" "member"
"member" ...
 $ month             : chr [1:5136261] "2021-01" "2021-01" "2021-01"
"2021-01" ...
 - 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_character(),
  ..   end_station_name = col_character(),
  ..   end_station_id = col_character(),
  ..   start_lat = col_double(),
  ..   start_lng = col_double(),
  ..   end_lat = col_double(),
  ..   end_lng = col_double(),
  ..   member_casual = col_character(),
  ..   month = col_character()
  .. )
```

```
glimpse(bike_df)
```

```
Rows: 5,136,261
Columns: 14
$ ride_id           <chr> "E19E6F1B8D4C42ED", "DC88F20C2C55F27F",
"EC45C94683~
$ rideable_type     <chr> "electric_bike", "electric_bike",
"electric_bike", ~
$ started_at        <dttm> 2021-01-23 16:14:19, 2021-01-27 18:43:08,
```

```
2021-01-~
$ ended_at           <dttm> 2021-01-23 16:24:44, 2021-01-27 18:47:12,
2021-01-~
$ start_station_name <chr> "California Ave & Cortez St", "California
Ave & Cor~
$ start_station_id   <chr> "17660", "17660", "17660", "17660",
"17660", "17660~
$ end_station_name   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, "Wood
St & Augu~
$ end_station_id     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, "657",
"13258",~
$ start_lat          <dbl> 41.90034, 41.90033, 41.90031, 41.90040,
41.90033, 4~
$ start_lng          <dbl> -87.69674, -87.69671, -87.69664, -87.69666,
-87.696~
$ end_lat            <dbl> 41.89000, 41.90000, 41.90000, 41.92000,
41.90000, 4~
$ end_lng            <dbl> -87.72000, -87.69000, -87.70000, -87.69000,
-87.700~
$ member_casual      <chr> "member", "member", "member", "member",
"casual", "~
$ month              <chr> "2021-01", "2021-01", "2021-01", "2021-01",
"2021-0~
```

```
summary(bike_df)
```

```
   ride_id          rideable_type        started_at
 Length:5136261     Length:5136261     Min.   :2020-10-01 00:00:06
 Class :character   Class :character   1st Qu.:2021-04-11 18:50:57
 Mode  :character   Mode  :character   Median :2021-06-21 18:01:31
                                       Mean   :2021-05-25 22:30:57
                                       3rd Qu.:2021-08-11 21:13:51
                                       Max.   :2021-09-30 23:59:48

    ended_at                   start_station_name start_station_id
 Min.   :2020-10-01 00:05:09   Length:5136261     Length:5136261
 1st Qu.:2021-04-11 19:15:05   Class :character   Class :character
 Median :2021-06-21 18:20:59   Mode  :character   Mode  :character
 Mean   :2021-05-25 22:51:34
 3rd Qu.:2021-08-11 21:33:57
 Max.   :2021-10-01 22:55:35
```

```
end_station_name   end_station_id       start_lat       start_lng
Length:5136261     Length:5136261     Min.   :41.64   Min.   :-87.84
Class :character   Class :character   1st Qu.:41.88   1st Qu.:-87.66
Mode  :character   Mode  :character   Median :41.90   Median :-87.64
                                      Mean   :41.90   Mean   :-87.65
                                      3rd Qu.:41.93   3rd Qu.:-87.63
                                      Max.   :42.08   Max.   :-87.52

   end_lat         end_lng       member_casual         month
Min.   :41.51   Min.   :-88.07   Length:5136261     Length:5136261
1st Qu.:41.88   1st Qu.:-87.66   Class :character   Class :character
Median :41.90   Median :-87.64   Mode  :character   Mode  :character
Mean   :41.90   Mean   :-87.65
3rd Qu.:41.93   3rd Qu.:-87.63
Max.   :42.17   Max.   :-87.44
NA's   :4821    NA's   :4821
```

Step-3: PROCESS

Case Study Roadmap - Process

Guiding questions

- What tools are you choosing and why?
  - To clean, analyze, and visualize this big dataset with ease, I have chosen to use R-programming which is an ideal tool when it comes to data wrangling, analysis, and visualization. It provides numerous packages that make data wrangling, analysis, and visualization so simple. The ggplot2 package is one of the most popular packages for its visualizations.
- Have you ensured your data's integrity?
  - Yes, the overall accuracy, completeness, and reliability of the dataset were verified
- What steps have you taken to ensure that your data is clean?

    - Through thorough investigation of the data, I identified the inconsistencies that exist in it and made all necessary cleaning on a copy of my dataset. Eventually, I made comparison between the editted copy and the original dataset to verify the effected changes.
- How can you verify that your data is clean and ready to analyze?
    - I made a manual and script-based comparison between the editted copy and the original dataset to verify the effected changes.
    - All attributes of both datasets were compared and analyzed to make sure that only desirable changes were effected.
- Have you documented your cleaning process so that you can review and share those results?
    - Yes, I logged every cleaning made on the data. The cleaning process is described in ensuing subsections.

Key tasks

1. Check the data for errors.
    - Done! Erroneous parts were identified and cleaned appropriately
2. Choose your tools.
    - I chose R-programming
3. Transform the data so you can work with it effectively.
    - Data was transformed into a form helpful for this analysis
4. Document the cleaning process.
    - Neatly documented

Deliverable

- Documentation of any cleaning or manipulation of data
    - Done

A. Dataset Cleaning Log

- As observed from the structures of the dataset displayed in the previous section and thorough investigation, the dataset has some inconsistencies and it has to be cleaned before it can be employed for anlysis to answer the main question of this project.

A0: Column "month" was Added During the Merging Process in previous step

- While merging the 12 datasets, their respective month values were added as a new column.

A1: Filtering

- Filter out unnecessary columns and create a new dataframe with columns of interest only!
  1. Delete the anonymized attributes/columns, namely "start_station_id", and "end_station_id". They are of no use in this anlysis.
  2. The columns named start_lat, start_lng, end lat, and end_lng offer no information useful for the analysis; as a result, they were filtered out.
  3. The column named ride_id is also of no importance and removed from the dataset (done later on)

A2: Fixing the Values of column named "rideable_type"

- This column has three values: 'electric_bike', 'classic_bike', and 'docked_bike'. However, they are supposed to be two. The 'classic_bike', and 'docked_bike' mean the same thing. The 'classic_bike' was introduced starting December, 2020; whereas the 'docked_bike' was in use before December 2020.
- Hence, 'docked_bike' is changed to 'classic_bike' under this column.

A3: Correcting Inconsistent Start Station Names (column: 'start_station_name') and Handling Missing Names

- Under the 'start_station_name column, the following station names contain " (Temp)".

```
{'California Ave & Francis Pl (Temp)',
 'Franklin St & Adams St (Temp)',
 'Halsted St & 18th St (Temp)', --> Problem noticed here!
 'Pulaski Rd & Eddy St (Temp)',
 'Wentworth Ave & 24th St (Temp)',
 'Wood St & Taylor St (Temp)'}
```

- It causes inconsistency in 'Halsted St & 18th St'. The same station is named differently; hence, the substring "(Temp)" and trailing white spaces are cleaned for consistency's sake.

```
 'Halsted St & 18th St',
 'Halsted St & 18th St (Temp)' --> 'Halsted St & 18th St'
```

- Twenty (20) rows of the dataset contain {'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)'}. I feel like the staff of Divvy performed some testing. Even the name implies so. Hence, I opted to remove these 20 rows from the dataset.

```
['HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)',
 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)']
```

- Missing station names are filled with "not_provided"

A4: Add a New column named "riding_duration"

- No riding duration of each client (casual or member) is provided in the dataset. Hence, I opted to add this column for it is vital in the analysis process. It is the difference of the start and end times of every ride in the dataset.

```
riding_duration <- ended_at - started_at
```

A5: Changing column names to something that makes more sense

- Some of the attributes' names of the dataset don't make sense. I had struggled to make sense of the 'rideable_type' and 'member_casual' columns until I checked their contents.

- Hence, for better and prompt understanding of the values they represent, I have changed these column names to bicycle_type and client_type, respectively, as depicted in what ensues.

```
'rideable_type' --> 'bicycle_type',  and
'member_casual' --> 'client_type'
```

A6: Cleaning was performed on the copied version of the original dataset

- I made sure that the original and copy don't have the same memory addresses. They shouldn't indicate to the same memory location!

```
bike_cleaned_df <- bike_df
```

B. Cleaning of dataframe "bike_cleaned_df" using R-Tools

B1: Create a dtaframe, which is a separate copy of the oginal one

- Make deep copy of bike_df to bike_cleaned_df as ensues:

```
bike_cleaned_df <- data.frame(bike_df), or
bike_cleaned_df <- cbind(bike_df)
```

```
dim(bike_df)
```

1. 5136261
2. 14

```
bike_cleaned_df <- data.frame(bike_df)
```

- Make sure they are stored on different memory locations!
    - The following comparison has to produce a False result

```
tracemem(bike_cleaned_df) == tracemem(bike_df)
```

```
tracemem(bike_cleaned_df) == tracemem(bike_df)
```

FALSE

B2: Check if column names are named correctly

- Ensure that column names are as per the rules: letters, numberss, _, & .

- Start with a letter or with a period followed by a non-numeric char.

```
bike_cleaned_df <- clean_names(bike_cleaned_df)
```

```
tracemem[0x00000000ea35dbf8 -> 0x00000000e63d75f0]: <Anonymous>
clean_names.data.frame clean_names eval eval withVisible
withCallingHandlers doTryCatch tryCatchOne tryCatchList tryCatch try
handle timing_fn evaluate_call evaluate doTryCatch tryCatchOne
tryCatchList doTryCatch tryCatchOne tryCatchList tryCatch <Anonymous>
handle_shell <Anonymous> <Anonymous>
tracemem[0x00000000e63d75f0 -> 0x00000000e63d7540]: <Anonymous>
clean_names.data.frame clean_names eval eval withVisible
withCallingHandlers doTryCatch tryCatchOne tryCatchList tryCatch try
handle timing_fn evaluate_call evaluate doTryCatch tryCatchOne
tryCatchList doTryCatch tryCatchOne tryCatchList tryCatch <Anonymous>
handle_shell <Anonymous> <Anonymous>
```

```
colnames(bike_cleaned_df)
```

1. 'ride_id'
2. 'rideable_type'
3. 'started_at'
4. 'ended_at'
5. 'start_station_name'
6. 'start_station_id'
7. 'end_station_name'
8. 'end_station_id'
9. 'start_lat'
10. 'start_lng'
11. 'end_lat'
12. 'end_lng'
13. 'member_casual'
14. 'month'

B3: Perform Filtering

- Columns start_station_id, end_station_id, start_lat, start_lng, end_lat, end_lng, and ride_id are filtered out as follows!

```
bike_cleaned_df <- bike_cleaned_df %>%
    select(ride_id, rideable_type, started_at, ended_at,
```

```
            start_station_name, end_station_name,
            member_casual, month)
```

```
colnames(bike_cleaned_df)
```

1. 'ride_id'
2. 'rideable_type'
3. 'started_at'
4. 'ended_at'
5. 'start_station_name'
6. 'end_station_name'
7. 'member_casual'
8. 'month'

```
head(bike_cleaned_df)
```

ride_id	rideable_type	started_at	ended_at	start_st
E19E6F1B8D4C42ED	electric_bike	2021-01-23 16:14:19	2021-01-23 16:24:44	California Cortez St
DC88F20C2C55F27F	electric_bike	2021-01-27 18:43:08	2021-01-27 18:47:12	California Cortez St
EC45C94683FE3F27	electric_bike	2021-01-21 22:35:54	2021-01-21 22:37:14	California Cortez St
4FA453A75AE377DB	electric_bike	2021-01-07 13:31:13	2021-01-07 13:42:55	California Cortez St
BE5E8EB4E7263A0B	electric_bike	2021-01-23 02:24:02	2021-01-23 02:24:45	California Cortez St
5D8969F88C773979	electric_bike	2021-01-09 14:24:07	2021-01-09 15:17:54	California Cortez St

B4: Under "rideable_type" column, change 'docked_bike' to 'classic_bike'

- Effectively changed to "classic_bike". As shown below, there are only 'electric_bike' and 'classic_bike'.

```
bike_cleaned_df$rideable_type[
    bike_cleaned_df$rideable_type == 'docked_bike'] <- 'classic_bike'

values <- bike_cleaned_df %>%
    select(rideable_type)

list_values <- list(values$rideable_type)

sapply(list_values, unique)
```

electric_bike
classic_bike

B5: Fix values of the 'start_station_name' column

- Make the following change:

```
    'Halsted St & 18th St (Temp)' --> 'Halsted St & 18th St'

bike_cleaned_df$start_station_name[
    bike_cleaned_df$start_station_name == 'Halsted St & 18th St
        (Temp)'] <- 'Halsted St & 18th St'
```

B6: Replace missing names under "start/end_station_name" with "not_provided"

```
    bike_cleaned_df <- bike_cleaned_df %>%
        mutate(mpg = ifelse(is.na(start_station_name), 'not_provided',
mpg))
```

- Before cleaning, there are 523467 cells with missing value under start_station_name column

```
sum(!complete.cases(bike_cleaned_df$start_station_name))
```

523467

```
sum(!complete.cases(bike_cleaned_df$end_station_name))
```

567268

```
bike_cleaned_df <- bike_cleaned_df %>%
    replace_na(list(start_station_name = 'not_provided'))

bike_cleaned_df <- bike_cleaned_df %>%
    replace_na(list(end_station_name = 'not_provided'))
```

- After Cleaning, no cell with missing value under start_station_name column

```
sum(!complete.cases(bike_cleaned_df$start_station_name))
```

0

```
sum(!complete.cases(bike_cleaned_df$end_station_name))
```

0

B7: Remove Rows containing 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)' under column name "start_station_name"

- Remove each row from bike_cleaned_df, which satisfies the following condition:

  `bike_cleaned_df$start_station_name == 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)'`

```
# 5136261
dim((bike_cleaned_df))
```

1. 5136261
2. 8

```
condition <- 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)'
bike_cleaned_df <- bike_cleaned_df[bike_cleaned_df$start_station_name
        != condition,]

dim((bike_cleaned_df))
```

1. 5136241
2. 8

```
head(bike_cleaned_df)
```

ride_id	rideable_type	started_at	ended_at	start_st

ride_id	rideable_type	started_at	ended_at	start_st
E19E6F1B8D4C42ED	electric_bike	2021-01-23 16:14:19	2021-01-23 16:24:44	California Cortez St
DC88F20C2C55F27F	electric_bike	2021-01-27 18:43:08	2021-01-27 18:47:12	California Cortez St
EC45C94683FE3F27	electric_bike	2021-01-21 22:35:54	2021-01-21 22:37:14	California Cortez St
4FA453A75AE377DB	electric_bike	2021-01-07 13:31:13	2021-01-07 13:42:55	California Cortez St
BE5E8EB4E7263A0B	electric_bike	2021-01-23 02:24:02	2021-01-23 02:24:45	California Cortez St
5D8969F88C773979	electric_bike	2021-01-09 14:24:07	2021-01-09 15:17:54	California Cortez St

B8: Add a New column named "riding_duration" using the mutate function

- The unit of the new column "riding_duration" is minutes

```
        riding_duration <- (ended_at - started_at)
```

```
bike_cleaned_df <- mutate(bike_cleaned_df,
     riding_duration = (ended_at - started_at))
```

```
bike_cleaned_df$riding_duration <- as.integer(
    bike_cleaned_df$riding_duration)/60
```

```
bike_cleaned_df$riding_duration <-
        round(bike_cleaned_df$riding_duration, digit=2)
```

```
dim((bike_cleaned_df))
```

1. 5136241
2. 9

```
head(bike_cleaned_df)
```

ride_id	rideable_type	started_at	ended_at	start_st
E19E6F1B8D4C42ED	electric_bike	2021-01-23 16:14:19	2021-01-23 16:24:44	California Cortez St
DC88F20C2C55F27F	electric_bike	2021-01-27 18:43:08	2021-01-27 18:47:12	California Cortez St
EC45C94683FE3F27	electric_bike	2021-01-21 22:35:54	2021-01-21 22:37:14	California Cortez St
4FA453A75AE377DB	electric_bike	2021-01-07 13:31:13	2021-01-07 13:42:55	California Cortez St
BE5E8EB4E7263A0B	electric_bike	2021-01-23 02:24:02	2021-01-23 02:24:45	California Cortez St
5D8969F88C773979	electric_bike	2021-01-09 14:24:07	2021-01-09 15:17:54	California Cortez St

B9: Remove the values of the new column "riding_duration" which are negative

- Under normal conditions, the start time can never be less than the end time; hence, any negative when (ended_at - started_at) is computed is abnormal. It should be the result of a system fault.
- Under normal scenario, the following condition must be met.

```
riding_duration <- (ended_at - started_at)
riding_duration >= 0
```

- A shown in the cell below, there are 3304 rows where the value of riding_duration is negative!
- All these rows were removed from the dataframe.

```
dim(bike_cleaned_df[bike_cleaned_df$riding_duration < 0,])
```

1. 3304
2. 9

```
dim(bike_cleaned_df)
```

1. 5136241
2. 9

```
bike_cleaned_df <- bike_cleaned_df[
    !(bike_cleaned_df$riding_duration < 0),]
```

```
dim(bike_cleaned_df)
```

1. 5132937
2. 9

```
sum(is.na(bike_cleaned_df$riding_duration))
```

0

B10: Remove the rows where the 'start_station_name' and 'end_station_name' indicate to the same place and the "riding_duration" is significantly small.

- Probably a duration of one minute or less may mean that the rider changed their mind and dock the bicycle back. As a result, if the 'start_station_name' and 'end_station_name' indicate to the same place and the "riding_duration" is less than or equal to one minute (60 seconds), I opted to drop the corresponding data-point or observation.
- Example:

```
      classic_bike 2021-01-17 10:50:25 2021-01-17 10:50:33
State St & Pearson St           State St & Pearson St
member  2021-01 0.13
```

- 54345 rows or data-points/observations were removed in this section, as illustrated in what ensues.

```
ids <-  list(bike_cleaned_df$ride_id)
ids <- sapply(ids, unique)
```

```
length(ids)
```

5132937

```
dim(bike_cleaned_df)
```

1. 5132937
2. 9

```
sum(is.na(bike_cleaned_df$riding_duration))
```

0

```
sameplace <- bike_cleaned_df[((bike_cleaned_df$start_station_name ==
     bike_cleaned_df$end_station_name) &
        (bike_cleaned_df$riding_duration <=1)),]


sameplace <- sameplace %>% drop_na(riding_duration)
ids_sub <-  list(sameplace$ride_id)
ids_sub <- sapply(ids_sub, unique)
dim(sameplace)
```

1. 70551
2. 9

```
sum(is.na(sameplace$riding_duration))
```

0

```
sameplace2 <- bike_cleaned_df[!(bike_cleaned_df$ride_id %in%
        ids_sub),]
dim(sameplace2)
```

1. 5062386
2. 9

```
sum(is.na(sameplace2$riding_duration))
```

0

```
bike_cleaned_df <- bike_cleaned_df[!(bike_cleaned_df$ride_id %in%
        ids_sub),]
dim(bike_cleaned_df)
```

1. 5062386
2. 9

B11: Remove the rows that contain NA under column name "riding_duration" if any.

```
sum(is.na(bike_cleaned_df$riding_duration))
```

0

```
bike_cleaned_df <- bike_cleaned_df[!
        (is.na(bike_cleaned_df$riding_duration)),]
dim(bike_cleaned_df)
```

1. 5062386
2. 9

B12: Rename 'rideable_type' and 'member_casual' columns as follows:

```
        'rideable_type' --> 'bicycle_type',  and
        'member_casual' --> 'client_type'
```

```
bike_cleaned_df <- bike_cleaned_df %>%
    rename(bicycle_type = rideable_type, client_type = member_casual)

colnames(bike_cleaned_df)
```

1. 'ride_id'
2. 'bicycle_type'
3. 'started_at'
4. 'ended_at'
5. 'start_station_name'
6. 'end_station_name'
7. 'client_type'
8. 'month'
9. 'riding_duration'

B13: For better readability, columns are reordered as ensues:

```
col_order <- c('ride_id', 'bicycle_type', 'client_type', 'month',
        'start_station_name',
              'end_station_name','started_at',
        'ended_at','riding_duration')
bike_cleaned_df <- bike_cleaned_df[, col_order]
```

- Partial view of the cleaned dataframe

```
head(bike_cleaned_df)
```

ride_id	bicycle_type	client_type	month	start_statio

ride_id	bicycle_type	client_type	month	start_statio
E19E6F1B8D4C42ED	electric_bike	member	2021-01	California Ave Cortez St
DC88F20C2C55F27F	electric_bike	member	2021-01	California Ave Cortez St
EC45C94683FE3F27	electric_bike	member	2021-01	California Ave Cortez St
4FA453A75AE377DB	electric_bike	member	2021-01	California Ave Cortez St
BE5E8EB4E7263A0B	electric_bike	casual	2021-01	California Ave Cortez St
5D8969F88C773979	electric_bike	casual	2021-01	California Ave Cortez St

```
dim(bike_cleaned_df)
```

1. 5062386
2. 9

B14: Outliers Detection in the "riding_duration" column

- 378674 outliers, they lie above the upper boundary, beyond $Q_3 + 1.5 \times IQR$
- These outliers have the potential to skew the analysis result; hence, they are removed!

1) Descriptive Statistics: using min(), mean(), and max()

- As illustrated below, the maximum difference between start time and end time is way much farther from the average value than minimum difference is.

```
min(bike_cleaned_df$riding_duration)
```

0

```
round(mean(bike_cleaned_df$riding_duration), digit = 2)
```

23.06

```
max(bike_cleaned_df$riding_duration)
```

55944.15

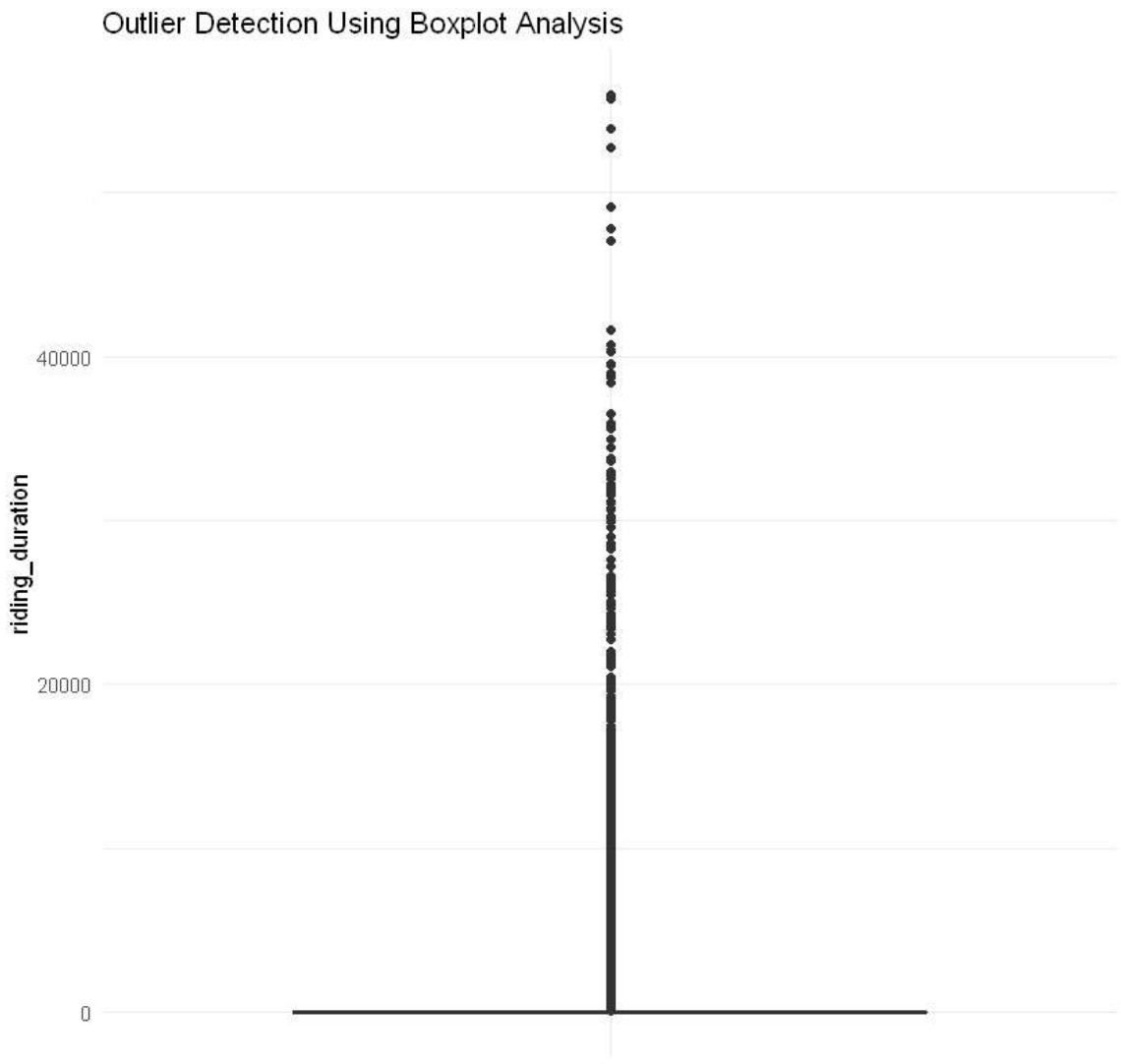
2) Using Histogram

- Another basic way to detect outliers is to draw a histogram of the data.
  - However, it gives no clear clue in this case

3) Boxplot

- Boxplot is useful in detecting potential outliers.
- A boxplot helps to visualize a quantitative variable by displaying five common location summary (minimum, median, first and third quartiles and maximum)

```
ggplot(bike_cleaned_df) +
  aes(x = "", y = riding_duration) +
  geom_boxplot(fill = "#0c4c8a") +
  labs(title = "Outlier Detection Using Boxplot Analysis")+
  theme_minimal()
```



3) Determination of outliers using minimum and maximum bounds

- Lower and upper bounds are computed by using the first and thrid quartiles along with 1.5 times of the inter quartile range (*IQR*).
  - $IQR = Q3 - Q1$
  - $lower\_bound = Q1 - 1.5 \times IQR$
  - $upper\_bound = Q3 + 1.5 \times IQR$
- In other words, all observations outside of the following interval (I) will be considered as potential outliers:

$$I = [Q_1 - 1.5 \times IQR : Q_3 + 1.5 \times IQR]$$

```
quantiles = quantile(bike_cleaned_df$riding_duration)
quantiles
```

0%

0

25%
7.3
50%
12.8
75%
23.03
100%
55944.15

```
Q1 = 7.25
Q2 = 12.75
Q3 = 22.98
IQR = Q3-Q1
lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR
```

- Number of observations whose riding_duration is less than lower_bound:

```
lesstn <-  bike_cleaned_df[
    bike_cleaned_df$riding_duration <lower_bound,]

dim(lesstn)
```

1. 0
2. 9

- Number of observations whose riding_duration is greater than upper_bound:

```
greatertn <-  bike_cleaned_df[
    bike_cleaned_df$riding_duration > upper_bound,]


dim(greatertn)
```

1. 378674
2. 9

```
head(greatertn)
```

	ride_id	bicycle_type	client_type	month	start_
6	5D8969F88C773979	electric_bike	casual	2021-01	Califor Cortez

	ride_id	bicycle_type	client_type	month	start_
133	0EDD906628861314	classic_bike	member	2021-01	Indiana Roosev
150	33A754F67BD337A7	classic_bike	member	2021-01	Kedzie Milwau
176	2CE7583737A818C4	classic_bike	casual	2021-01	Rush S
343	795E03AF09CE93F8	classic_bike	member	2021-01	Univers St
344	E9B4DB03F6F7226C	electric_bike	member	2021-01	Univers St

```
ids2 <-  list(greatertn$ride_id)
ids2 <- sapply(ids2, unique)

dim(bike_cleaned_df)
```

1. 5062386
2. 9

```
bike_cleaned_df <- bike_cleaned_df[!(bike_cleaned_df$ride_id %in%
        ids2),]
dim(bike_cleaned_df)
```

1. 4683712
2. 9

```
5078592-4699918
```

378674

B15: Remove the ride_id column, it is no longer useful

```
col_order2 <- c('bicycle_type', 'client_type', 'month',
        'start_station_name',
              'end_station_name','started_at',
        'ended_at','riding_duration')
```

```
bike_cleaned_df <- bike_cleaned_df[, col_order2]

colnames(bike_cleaned_df)
```

1. 'bicycle_type'
2. 'client_type'
3. 'month'
4. 'start_station_name'
5. 'end_station_name'
6. 'started_at'
7. 'ended_at'
8. 'riding_duration'

B16: Create another new column called day_of_week

- It stores the day of the week that each ride started

```
bike_cleaned_df$day_of_week <-
        weekdays(as.Date(bike_cleaned_df$started_at))

col_order3 <- c('bicycle_type', 'client_type', 'month',
        'start_station_name',
               'end_station_name','started_at',
        'day_of_week','ended_at','riding_duration')
bike_cleaned_df <- bike_cleaned_df[, col_order3]

colnames(bike_cleaned_df)
```

1. 'bicycle_type'
2. 'client_type'
3. 'month'
4. 'start_station_name'
5. 'end_station_name'
6. 'started_at'
7. 'day_of_week'
8. 'ended_at'
9. 'riding_duration'

```
head(bike_cleaned_df)
```

	bicycle_type	client_type	month	start_station_name	end_sta
1	electric_bike	member	2021-01	California Ave & Cortez St	not_prov

	bicycle_type	client_type	month	start_station_name	end_sta
2	electric_bike	member	2021-01	California Ave & Cortez St	not_prov
3	electric_bike	member	2021-01	California Ave & Cortez St	not_prov
4	electric_bike	member	2021-01	California Ave & Cortez St	not_prov
5	electric_bike	casual	2021-01	California Ave & Cortez St	not_prov
7	electric_bike	member	2021-01	California Ave & Cortez St	not_prov

```
dim(bike_cleaned_df)
```

1. 4683712
2. 9

B17: Save the cleaned Data to disk

```
# Path
path1 <- "C:\\Users\\afitwi1\\Desktop\\googleproj\\dataset\\"
file1 <- "cleaned_bike_data_20211017.csv"
path1 <- paste(path1, file1, sep='')

# write
write.csv(bike_cleaned_df,path1, row.names = FALSE)
```

Summary of Data Preprocessing

- Dimensions of the cleaned dataframe
  - 20 rows werer removed due to 'HUBBARD ST BIKE CHECKING (LBS-WH-TEST)' under column name "start_station_name" and
  - 3304 Data-points/obseravtions were removed because they have negative values under column name "riding_duration"

    - 54345 datapoints were removed because they have the same starting and ending stations and a riding duration of less than or equal to 60 seocnds
    - 378,674 data-points were removed because they fall on the outlier region. They have the potential to skew the result.
    - 7 columns were removed in the process step
    - 2 new columns were added in the process step: riding_duration & day_of_week
    - Ended up with a cleaned dataframe of dimensions

$$Rows = 4,683,712 \; and \; Columns = 9$$

- 2 columns renamed: 'rideable_type' --> 'bicycle_type', and 'member_casual' --> 'client_type'

Step-4: ANALYZE

Now that my data is stored appropriately and has been prepared for analysis, it is time I start putting it to work. I have used the following Case Study Roadmap as a guide to successfully perform the analysis phase.

Case Study Roadmap - Analyze

Guiding questions

- How should the data be organized to perform analysis on it?
- Has the data been properly formatted?
- What surprises were discovered in the data?
- What trends or relationships were found in the data?
- How will these insights help answer the company's business questions?

Key tasks

1. Aggregate the data so it’s useful and accessible.
2. Organize and format the data.
3. Perform calculations.
4. Identify trends and relationships.

Deliverable

- A summary of my analysis

1. Cleaned Dataframe overview

- The unit of column "riding_duration" is minutes

```
cleaned_bike_df <-
        read_csv("./dataset/cleaned_bike_data_20211017.csv")
```

```
Parsed with column specification:
cols(
  bicycle_type = col_character(),
  client_type = col_character(),
  month = col_character(),
  start_station_name = col_character(),
  end_station_name = col_character(),
  started_at = col_datetime(format = ""),
  day_of_week = col_character(),
  ended_at = col_datetime(format = ""),
  riding_duration = col_double()
)
```

```
colnames(cleaned_bike_df)
```

1. 'bicycle_type'
2. 'client_type'
3. 'month'
4. 'start_station_name'
5. 'end_station_name'
6. 'started_at'
7. 'day_of_week'
8. 'ended_at'
9. 'riding_duration'

```
glimpse(cleaned_bike_df)
```

```
Rows: 4,683,712
Columns: 9
$ bicycle_type       <chr> "electric_bike", "electric_bike",
"electric_bike", ~
$ client_type        <chr> "member", "member", "member", "member",
```

```
"casual", "~
$ month              <chr> "2021-01", "2021-01", "2021-01", "2021-01",
"2021-0~
$ start_station_name <chr> "California Ave & Cortez St", "California
Ave & Cor~
$ end_station_name   <chr> "not_provided", "not_provided",
"not_provided", "no~
$ started_at         <dttm> 2021-01-23 16:14:19, 2021-01-27 18:43:08,
2021-01-~
$ day_of_week        <chr> "Saturday", "Wednesday", "Thursday",
"Thursday", "S~
$ ended_at           <dttm> 2021-01-23 16:24:44, 2021-01-27 18:47:12,
2021-01-~
$ riding_duration    <dbl> 10.42, 4.07, 1.33, 11.70, 0.72, 5.58, 6.67,
2.52, 7~
```

```
sum(is.na(cleaned_bike_df$riding_duration))
```

0

2. Number of Anual-member riders vs Number of casual riders

- Of all the riders in this data, 42.7% are casual members and 57.3% are annual members

```
sum(is.na(cleaned_bike_df$client_type))
```

0

```
vec <- c(dim(cleaned_bike_df))
count <- list(rep(1, vec[1]))
groups <- data.frame(client_type = cleaned_bike_df$client_type, cnt =
        count)
names(groups)[2] <- 'count'
aggregates <- groups %>% group_by(client_type) %>% summarize(Counts =
        sum(count))
aggregates
```

client_type	Counts
casual	1998061
member	2685651

3. Total rides in minutes for each group

- Member riders rode for **34, 245, 797** minutes over the period of the last 12 months
- Casual riders rode for **34, 064, 954** minutes over the period of the last 12 months

```
sum_mins <- cleaned_bike_df %>%
            group_by(client_type) %>% summarize(sum_minutes =
        sum(riding_duration))
```

```
sum_mins
```

client_type	sum_minutes
casual	34064954
member	34245797

4. Daily Average of the rider groups

```
daily_gp <- cleaned_bike_df[,c('client_type','riding_duration')]
daily_gp1 <- daily_gp  %>%  group_by(client_type) %>%
                  summarize(daily_mean = mean(riding_duration))
daily_gp1
```

client_type	daily_mean
casual	17.04901
member	12.75140

5. Monthly Average Rides of the Rider Groups

```
mnth_gp <-
        cleaned_bike_df[,c('client_type','month','riding_duration')]
mnth_gp <- mnth_gp  %>%  group_by(client_type, month) %>%
                  summarize(montly_avg_minutes =
        mean(riding_duration))
head(mnth_gp)
```

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

client_type	month	montly_avg_minutes
casual	2020-10	16.72820
casual	2020-11	16.78183

client_type	month	montly_avg_minutes
casual	2020-12	15.13311
casual	2021-01	14.41938
casual	2021-02	16.63120
casual	2021-03	17.89560

6. Average Minutes of rides on each day of the week for members and casual riders

```
# Average
day_minuntes <-
        cleaned_bike_df[,c('client_type','day_of_week','riding_duration')]

day_minuntes <- day_minuntes  %>%  group_by(client_type, day_of_week)
        %>%
                  summarize(Weekdays_avg_minutes =
        mean(riding_duration))


# Round to two decimal places
day_minuntes <- day_minuntes %>% mutate_if(is.numeric, ~round(.,2))
head(day_minuntes)
```

```
`summarise()` has grouped output by 'client_type'. You can override
using the `.groups` argument.
`mutate_if()` ignored the following grouping variables:
Column `client_type`
```

client_type	day_of_week	Weekdays_avg_minutes
casual	Friday	16.55
casual	Monday	16.95
casual	Saturday	18.19
casual	Sunday	18.37
casual	Thursday	15.73
casual	Tuesday	16.07

7. Correlation between Weekdays and Number of Rides

- Found no correlation!

```
vec7 <- c(dim(cleaned_bike_df))
number <- list(rep(1, vec7[1]))
cor7_df <- data.frame(ymd_val = cleaned_bike_df$started_at,
                      cnt = number)
names(cor7_df)[2] <- 'number'

cor7_df$ymd_val <- as.Date(cor7_df$ymd_val, format = "%Y-%m-%d")
cor7_df <- cor7_df%>% group_by(ymd_val) %>% summarize(number_of_rides
        = sum(number))
cor7_df$month <- format(cor7_df$ymd_val,"%Y-%m")
cor7_df$day_of_week <- weekdays(as.Date(cor7_df$ymd_val))

cor7_df <- mutate(cor7_df, day_rep = ifelse(day_of_week == "Sunday",
        1,
                    ifelse(day_of_week == "Monday", 2,
                    ifelse(day_of_week == "Tuesday", 3,
                    ifelse(day_of_week == "Wednesday", 4,
                    ifelse(day_of_week == "Thursday", 5,
                    ifelse(day_of_week == "Friday", 6,
                    ifelse(day_of_week == "Saturday", 7,
                           "no")))))))
cor7_df$day_rep <- as.integer(cor7_df$day_rep)
cor(select(cor7_df,  number_of_rides, day_rep))
```

	number_of_rides	day_rep
number_of_rides	1.00000000	0.09362992
day_rep	0.09362992	1.00000000

8. Correlation between Season and Number of Rides

- Seasons in relation to North America, specifically Chicago:
  - Winter := {December, January, February}, season_code := 1
  - Spring := {March, April, May}, season_code := 2
  - Summer := {June, July, August}, season_code := 3
  - Fall := {September, October, November}, season_code := 4
- There exist a strong correlation between the seasons and the number of rides. It is computed to be **0.58** (illustrated below). As a result, there is an uptrend correlation as we move from the end of winter to summer, and conversely

```
vec8 <- c(dim(cleaned_bike_df))
number8 <- list(rep(1, vec8[1]))
```

```
cor8_df <- data.frame(ymd_val = cleaned_bike_df$started_at,
                      cnt = number8)
names(cor8_df)[2] <- 'number'

cor8_df$ymd_val <- as.Date(cor8_df$ymd_val, format = "%Y-%m-%d")
cor8_df <- cor8_df%>% group_by(ymd_val) %>% summarize(number_of_rides
       = sum(number))
cor8_df$month <- format(cor8_df$ymd_val,"%Y-%m")
cor8_df$month2 <- format(cor8_df$ymd_val,"%m")
cor8_df$month2 <- as.integer(cor8_df$month2)

cor8_df <- mutate(cor8_df, season_code = ifelse(month2 == 1, 1,
                   ifelse(month2 == 2, 1,
                   ifelse(month2 == 3, 2,
                   ifelse(month2 == 4, 2,
                   ifelse(month2 == 5, 2,
                   ifelse(month2 == 6, 3,
                   ifelse(month2 == 7, 3,
                   ifelse(month2 == 8, 3,
                   ifelse(month2 == 9, 4,
                   ifelse(month2 == 10, 4,
                   ifelse(month2 == 11, 4,
                   ifelse(month2 == 12, 1,
                          "no"))))))))))))

cor8_df$season_code <- as.integer(cor8_df$season_code)
head(cor8_df)
```

ymd_val	number_of_rides	month	month2	season_code
2020-10-01	10728	2020-10	10	4
2020-10-02	13604	2020-10	10	4
2020-10-03	13296	2020-10	10	4
2020-10-04	11629	2020-10	10	4
2020-10-05	11797	2020-10	10	4
2020-10-06	14988	2020-10	10	4

```
cor(select(cor8_df,  season_code, number_of_rides))
```

	season_code	number_of_rides

	season_code	number_of_rides
season_code	1.0000000	0.5815191
number_of_rides	0.5815191	1.0000000

9. Correlation between Seasons and Riding durations

- As demostrated below, the correlation between seasons and riding durations is 0.004, which is almost zero. Hence, there is no correlation between the seasons and riding durations.

```
cor9_df <- cleaned_bike_df[,c("started_at","riding_duration")]
cor9_df$started_at <- as.Date(cor9_df$started_at, format = "%Y-%m-%d")
cor9_df$month <- format(cor9_df$started_at,"%m")
cor9_df$month <- as.integer(cor9_df$month)

cor9_df <- mutate(cor9_df, season_code = ifelse(month == 1, 1,
                    ifelse(month == 2, 1,
                    ifelse(month == 3, 2,
                    ifelse(month == 4, 2,
                    ifelse(month == 5, 2,
                    ifelse(month == 6, 3,
                    ifelse(month == 7, 3,
                    ifelse(month == 8, 3,
                    ifelse(month == 9, 4,
                    ifelse(month == 10, 4,
                    ifelse(month == 11, 4,
                    ifelse(month == 12, 1,
                           "no"))))))))))))

cor9_df$season_code <- as.integer(cor9_df$season_code)
head(cor9_df)
```

started_at	riding_duration	month	season_code
2021-01-23	10.42	1	1
2021-01-27	4.07	1	1
2021-01-21	1.33	1	1
2021-01-07	11.70	1	1
2021-01-23	0.72	1	1

started_at	riding_duration	month	season_code
2021-01-04	5.58	1	1

```
cor(select(cor9_df,  season_code, riding_duration))
```

	season_code	riding_duration
season_code	1.000000000	0.004026174
riding_duration	0.004026174	1.000000000

10. Most Visited Stations

```
ms_df <- cleaned_bike_df[,c('client_type','start_station_name',
        'end_station_name')]
```

```
head(ms_df)
```

client_type	start_station_name	end_station_name
member	California Ave & Cortez St	not_provided
member	California Ave & Cortez St	not_provided
member	California Ave & Cortez St	not_provided
member	California Ave & Cortez St	not_provided
casual	California Ave & Cortez St	not_provided
member	California Ave & Cortez St	not_provided

By Casual Riders

- 228,056 observations have no named start stations for casual riders
- 256,860 datapoints have no named end stations for casual riders
- Totally, 484,916 rows don't have named start or end stations for casual riders

```
cas_df <- ms_df[ms_df$client_type == "casual",]
```

```
not_provided_sc <- cas_df[cas_df$start_station_name ==
        "not_provided",]
dim(not_provided_sc)
```

1. 228056
2. 3

```
not_provided_ec <- cas_df[cas_df$end_station_name == "not_provided",]
dim(not_provided_ec)
```

1. 256860
2. 3

```
ssc_df <- cas_df [,c('client_type','start_station_name')]
esc_df <- cas_df [,c('client_type','end_station_name')]

vec10s <- c(dim(ssc_df))
vec10e <- c(dim(esc_df))
number10s <- list(rep(1, vec10s[1]))
number10e <- list(rep(1, vec10e[1]))

ssc_df <- data.frame(start_station_name = ssc_df$start_station_name,
                         cnt = number10s)
esc_df <- data.frame(end_station_name = esc_df$end_station_name,
                         cnt = number10e)
names(ssc_df)[2] <- 'number'
names(esc_df)[2] <- 'number'

ssc_df <- ssc_df%>% group_by(start_station_name)%>%
   summarize(sum_start_stations = sum(number))%>%
   arrange(-sum_start_stations)
ssc_df <- ssc_df[ssc_df$start_station_name  != "not_provided",]

esc_df <- esc_df%>% group_by(end_station_name)%>%
   summarize(sum_end_stations = sum(number))%>%
   arrange(-sum_end_stations)

esc_df <- esc_df[esc_df$end_station_name  != "not_provided",]
```

- Top Six Start Stations most visited by Casual Riders, excluding unknown stations

```
head(ssc_df)
```

start_station_name	sum_start_stations
Streeter Dr & Grand Ave	45648
Millennium Park	21291
Michigan Ave & Oak St	21155

start_station_name	sum_start_stations
Theater on the Lake	17430
Wells St & Concord Ln	17329
Shedd Aquarium	17221

- Top Six End Stations most visited by Casual Riders, excluding unknown stations

```
head(esc_df)
```

end_station_name	sum_end_stations
Streeter Dr & Grand Ave	45367
Millennium Park	22994
Michigan Ave & Oak St	21832
Theater on the Lake	17845
Wells St & Concord Ln	17135
Lake Shore Dr & North Blvd	15854

By Annual Members

- 254, 359 rows have no named start stations for member riders
- 254, 885 datapoints have no named end stations for member riders
- In total, 508, 718 observations contain no named start or end stations for member riders

```
mem_df <- ms_df[ms_df$client_type == "member",]
```

```
not_pro_ss <- mem_df[mem_df$start_station_name == "not_provided",]
dim(not_pro_ss)
```

1. 254359
2. 3

```
not_pro_es <- mem_df[mem_df$end_station_name == "not_provided",]
dim(not_pro_es)
```

1. 254885
2. 3

```
ssm_df <- mem_df [,c('client_type','start_station_name')]
esm_df <- mem_df [,c('client_type','end_station_name')]

vec10sm <- c(dim(ssm_df))
vec10em <- c(dim(esm_df))
number10sm <- list(rep(1, vec10sm[1]))
number10em <- list(rep(1, vec10em[1]))

ssm_df <- data.frame(start_station_name = ssm_df$start_station_name,
                         cnt = number10sm)
esm_df <- data.frame(end_station_name = esm_df$end_station_name,
                         cnt = number10em)
names(ssm_df)[2] <- 'number'
names(esm_df)[2] <- 'number'

ssm_df <- ssm_df%>% group_by(start_station_name)%>%
   summarize(sum_start_stations = sum(number))%>%
   arrange(-sum_start_stations)
ssm_df <- ssm_df[ssm_df$start_station_name  != "not_provided",]

esm_df <- esm_df%>% group_by(end_station_name)%>%
   summarize(sum_end_stations = sum(number))%>%
   arrange(-sum_end_stations)

esm_df <- esm_df[esm_df$end_station_name  != "not_provided",]
```

- Top Six Start Stations most visited by Member Riders, excluding unknown stations

```
head(ssm_df)
```

start_station_name	sum_start_stations
Clark St & Elm St	23285
Wells St & Concord Ln	21551
Kingsbury St & Kinzie St	20582
Wells St & Elm St	19368
Dearborn St & Erie St	17896
Wells St & Huron St	17623

- Top Six End Stations most visited by Member Riders, excluding unknown stations

```
head(esm_df)
```

end_station_name	sum_end_stations
Clark St & Elm St	23686
Wells St & Concord Ln	22193
Kingsbury St & Kinzie St	20938
Wells St & Elm St	19854
Dearborn St & Erie St	18425
St. Clair St & Erie St	17845

Step-5: SHARE

Now that I have performed my analysis and gained some insights into my data, it is time to create visualizations to share my findings. Moreno has reminded me that they should be sophisticated and polished in order to effectively communicate to the executive team. I employed the following Case Study Roadmap as a guide.

Case Study Roadmap - Share

Guiding questions

- Were I able to answer the question of how annual members and casual riders use Cyclistic bikes differently?
- What story does my data tell?
- How do my findings relate to my original question?
- Who is my audience? What is the best way to communicate with them?
- Can data visualization help me share my findings?
- Is my presentation accessible to my audience?

Key tasks

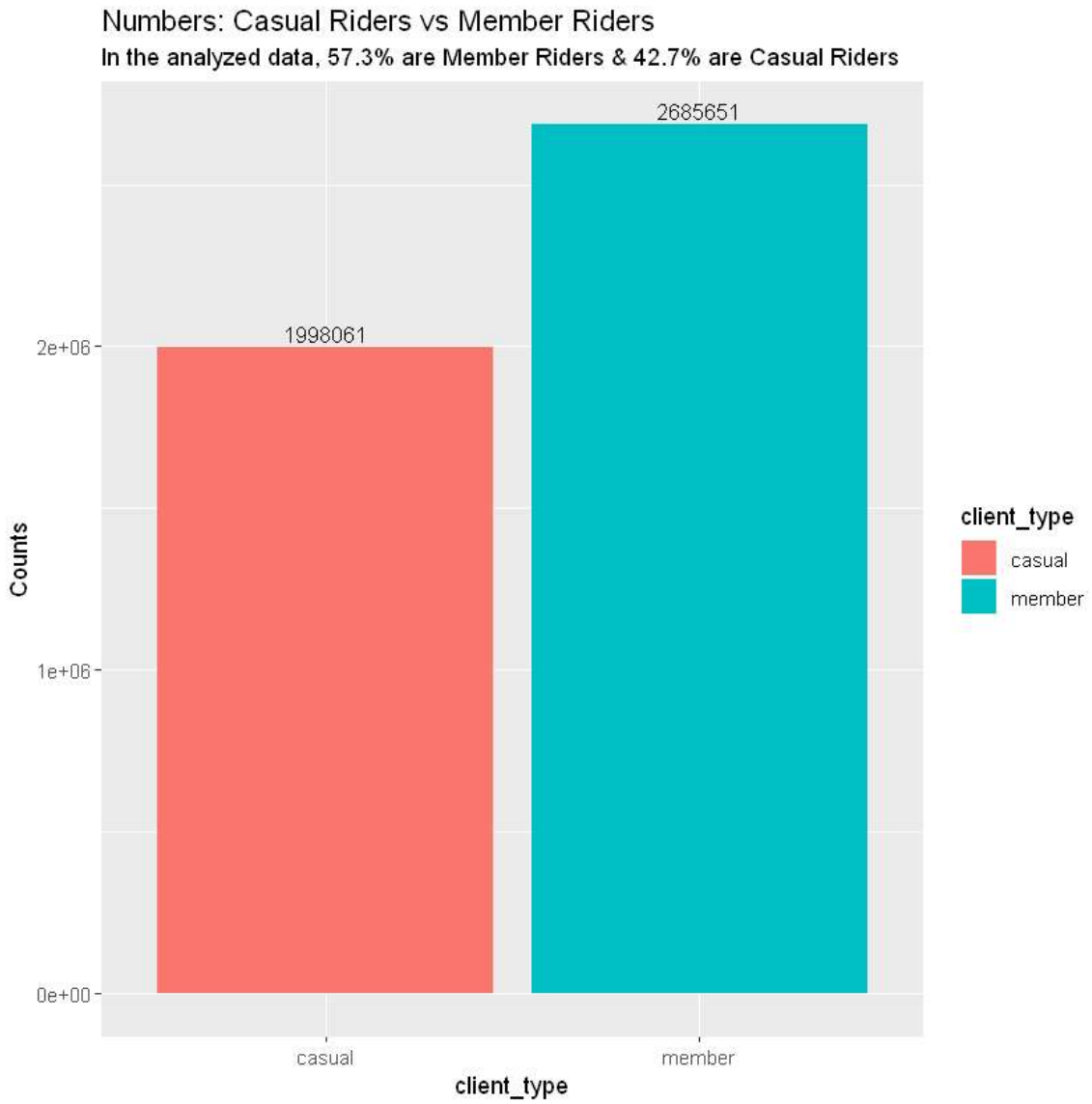
1. Determine the best way to share my findings.
2. Create effective data visualizations.
3. Present my findings.
4. Ensure my work is accessible.

Deliverable

- Supporting visualizations and key findings

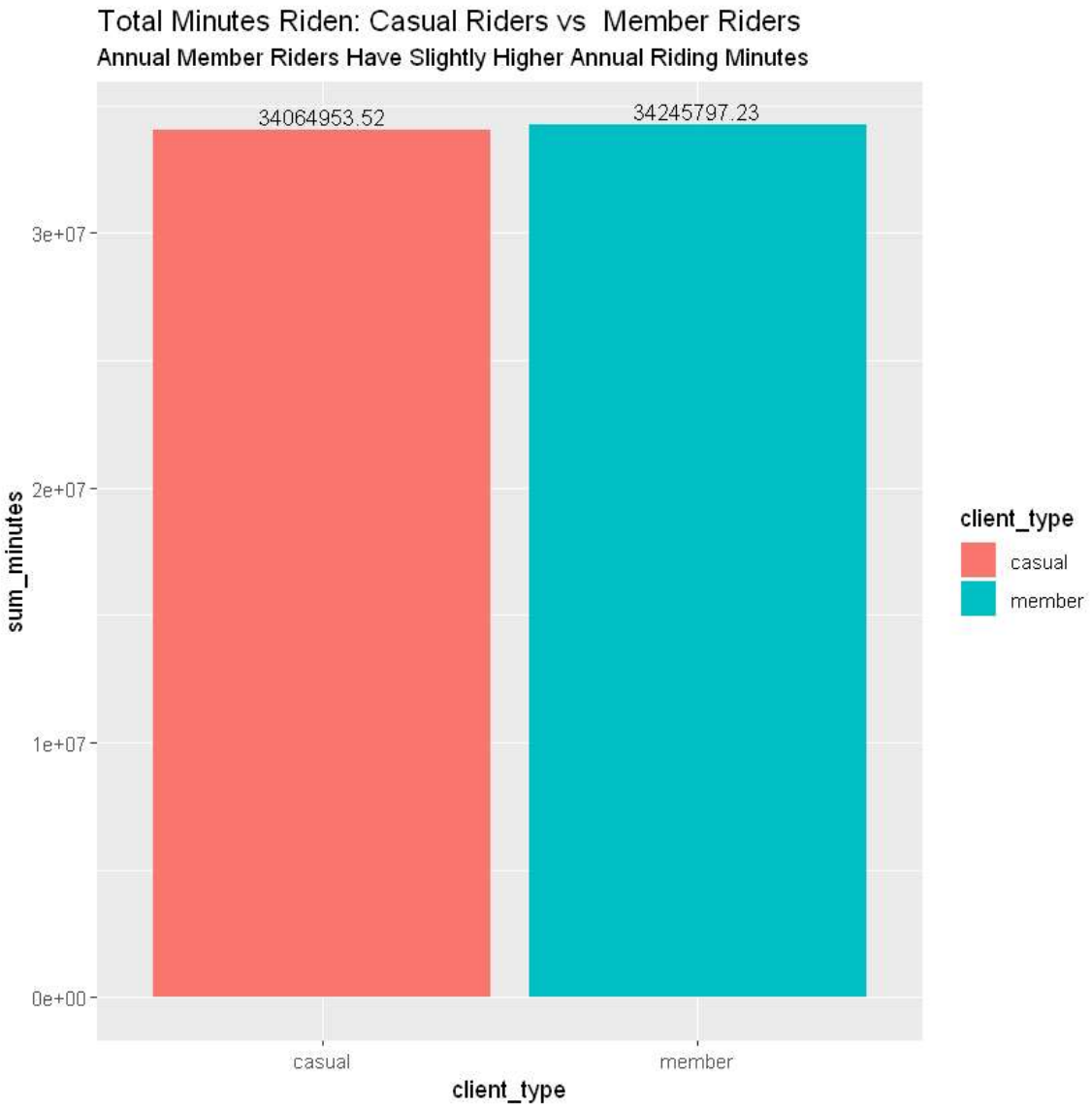
1. Number of Casual Riders vs Annual Member Riders

```
p<-ggplot(data=aggregates, aes(x=client_type, y=Counts, fill
        =client_type)) +
  labs(title = "Numbers: Casual Riders vs Member Riders",
       subtitle = "In the analyzed data, 57.3% are Member Riders &
         42.7% are Casual Riders")+
 geom_text(aes(label=Counts), vjust=-0.3, size=3.5)+
  geom_bar(stat="identity")
p
```



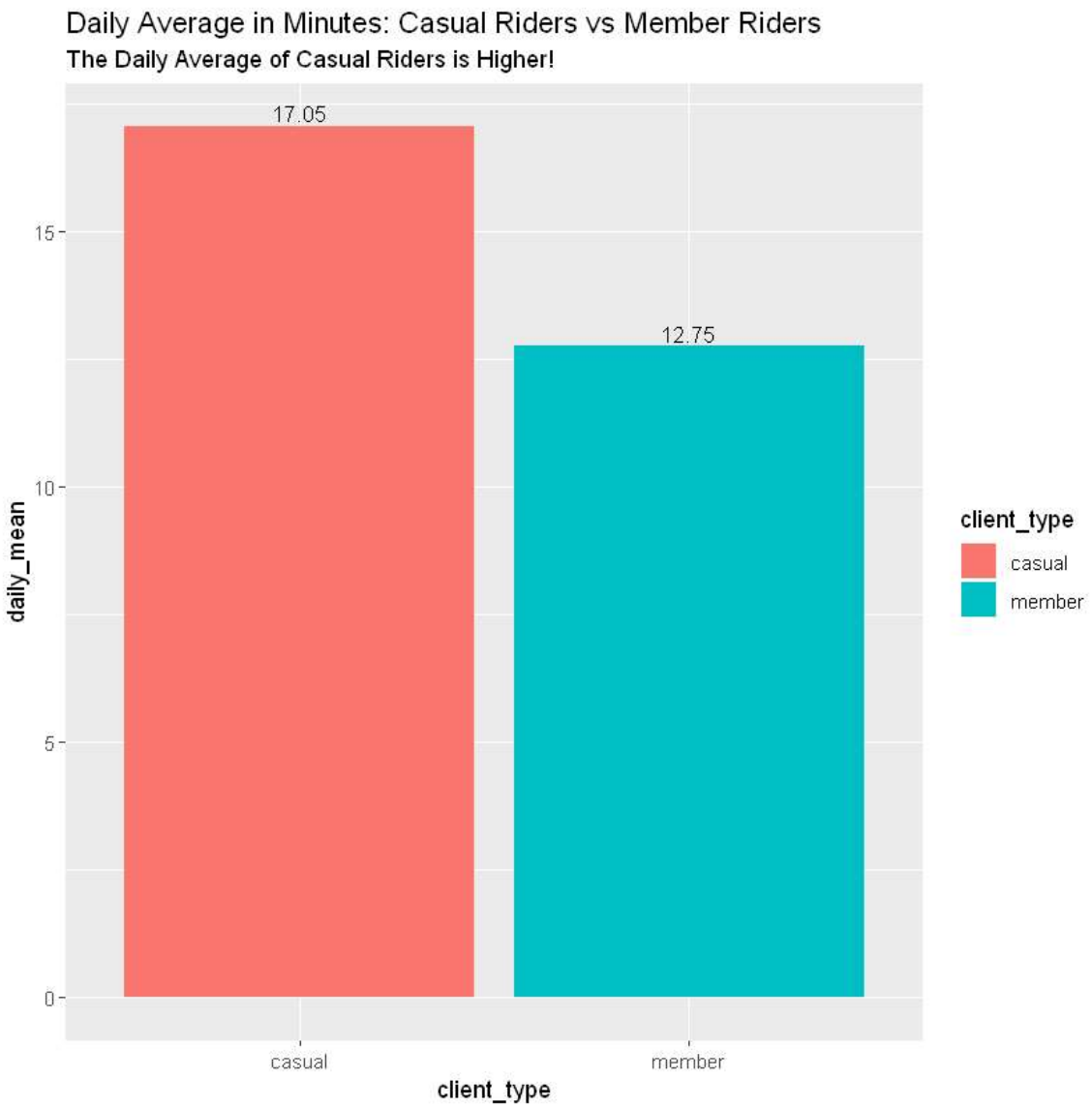
2. Total Minutes Riden over the past 12 months by both Tyes of Riders

```
p<-ggplot(data=sum_mins, aes(x=client_type, y=sum_minutes, fill
        =client_type)) +
  labs(title = "Total Minutes Riden: Casual Riders vs  Member Riders",
       subtitle = "Annual Member Riders Have Slightly Higher Annual
        Riding Minutes")+
 geom_text(aes(label=sum_minutes), vjust=-0.3, size=3.5)+
  geom_bar(stat="identity")
p
```



3. Daily Riding Average of Casual Riders vs Annual Member Riders

```
# Round to two decimal places
daily_gp1 <- daily_gp1 %>% mutate_if(is.numeric, ~round(.,2))


# Bar Plot
p<-ggplot(data=daily_gp1, aes(x=client_type, y=daily_mean, fill
        =client_type)) +
  labs(title = "Daily Average in Minutes: Casual Riders vs Member
        Riders",
       subtitle = "The Daily Average of Casual Riders is Higher!")+
 geom_text(aes(label=daily_mean), vjust=-0.3, size=3.5)+
  geom_bar(stat="identity")
p
```



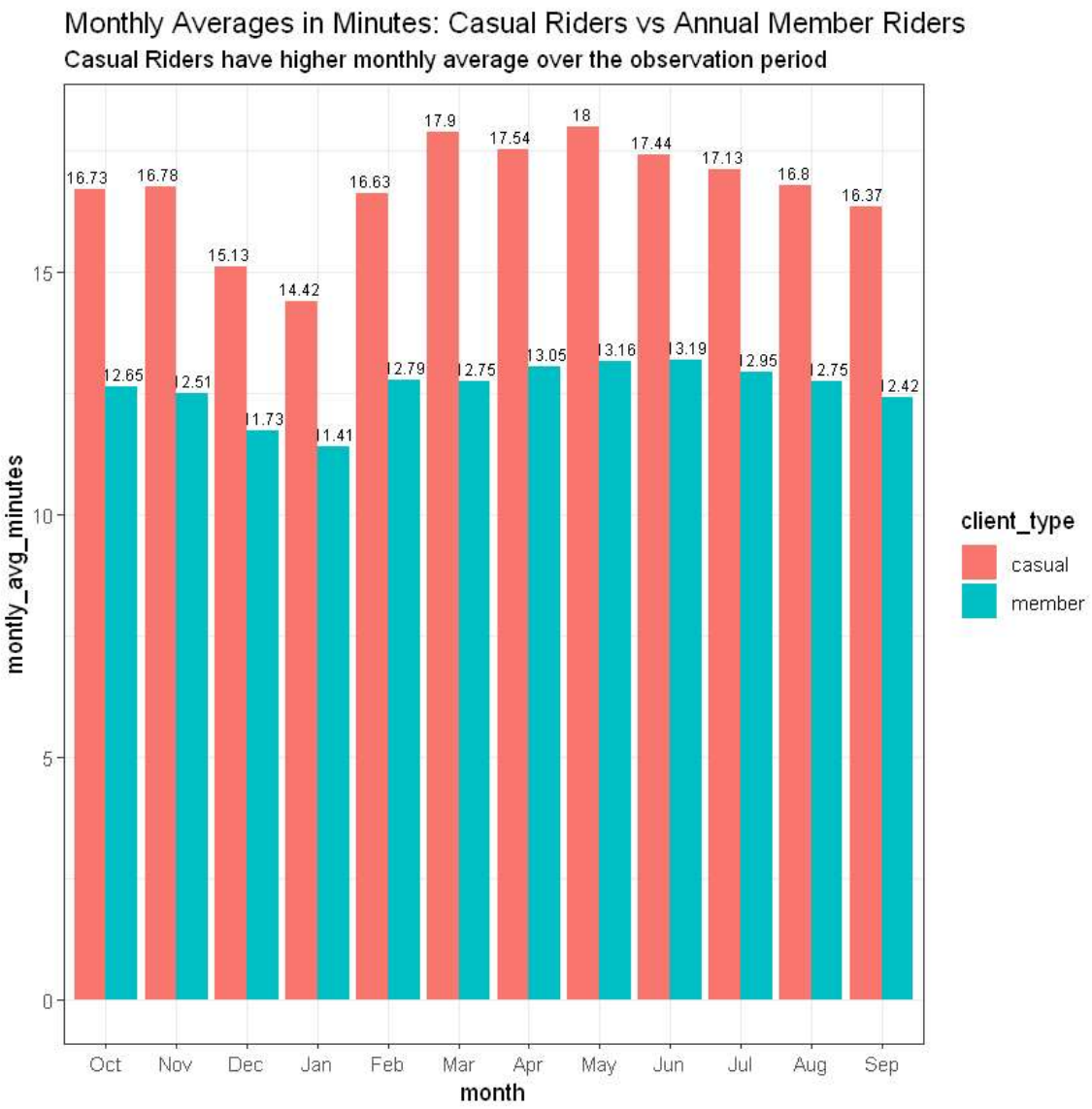
4. Monthly Average Rides of Casual Riders vs Member Riders in Minutes

- As illustrated in the barplot below, the monthly average rides in minute of the casual riders is higher than that of annual member riders throughout the period of observation.

```
mnth_gp1 <- cbind(mnth_gp)
mnth_gp1$month <- substr(mnth_gp1$month, 6,7)
mnth_gp1$month <- as.integer(mnth_gp1$month)
#months vector assuming 1st month is Jan.
mymonths <- c("Jan","Feb","Mar",
              "Apr","May","Jun",
              "Jul","Aug","Sep",
              "Oct","Nov","Dec")
```

```
#add abbreviated month name
mnth_gp1$month <- mymonths[mnth_gp1$month]
#head(mnth_gp1)

# Round to two decimal places
mnth_gp1 <- mnth_gp1 %>% mutate_if(is.numeric, ~round(.,2))

# Order the x-axis vals
xorder0 <- c("Oct","Nov","Dec",
             "Jan","Feb","Mar",
               "Apr","May","Jun",
               "Jul","Aug","Sep" )
mnth_gp1$month <- factor(mnth_gp1$month, levels=xorder0)



# Grouped Bars
ggplot(mnth_gp1, aes(x=month, y=montly_avg_minutes,
    fill=client_type, group = client_type)) +
labs(title = "Monthly Averages in Minutes: Casual Riders vs Annual
        Member Riders",
     subtitle = "Casual Riders have higher monthly average over the
        observation period")+
geom_text(aes(x=month, y=montly_avg_minutes,
              label=montly_avg_minutes, group = client_type),
          position = position_dodge(width = 1),
          vjust=-0.5, size=2.5)+
    geom_bar(position="dodge", stat="identity")+
theme_bw()
```

```
`mutate_if()` ignored the following grouping variables:
Column `client_type`
```

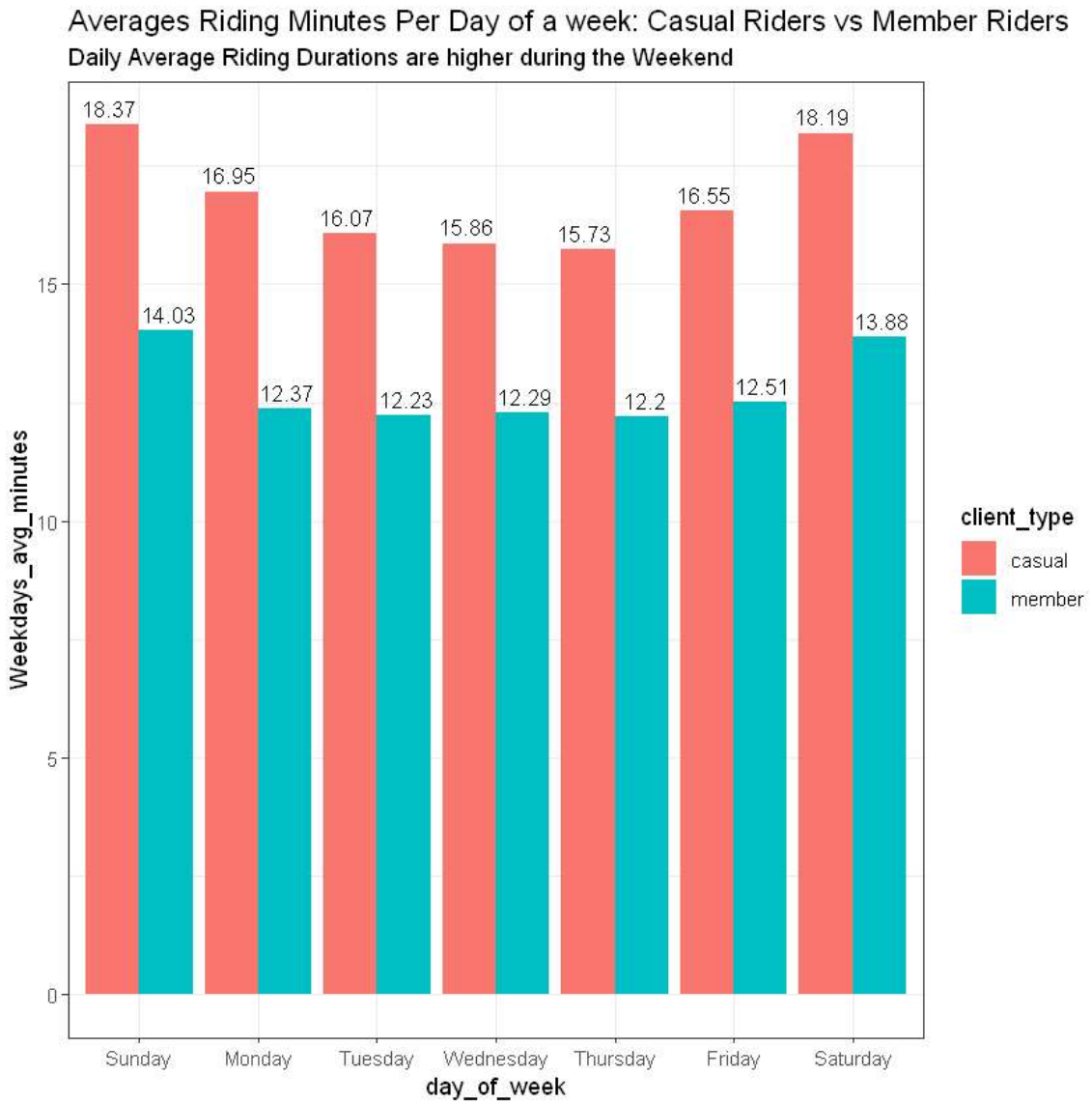


5. Average Minutes of riding per day of the week: Member vs Casual Riders

- As clearly depicted in the line graphs and barplots below, the daily average rides in minutes of the casual riders are consistently higher than that of annual member riders over the whole period of observation (last 12-months).
- Besides, the average daily riding durations are much higher during the weekend.

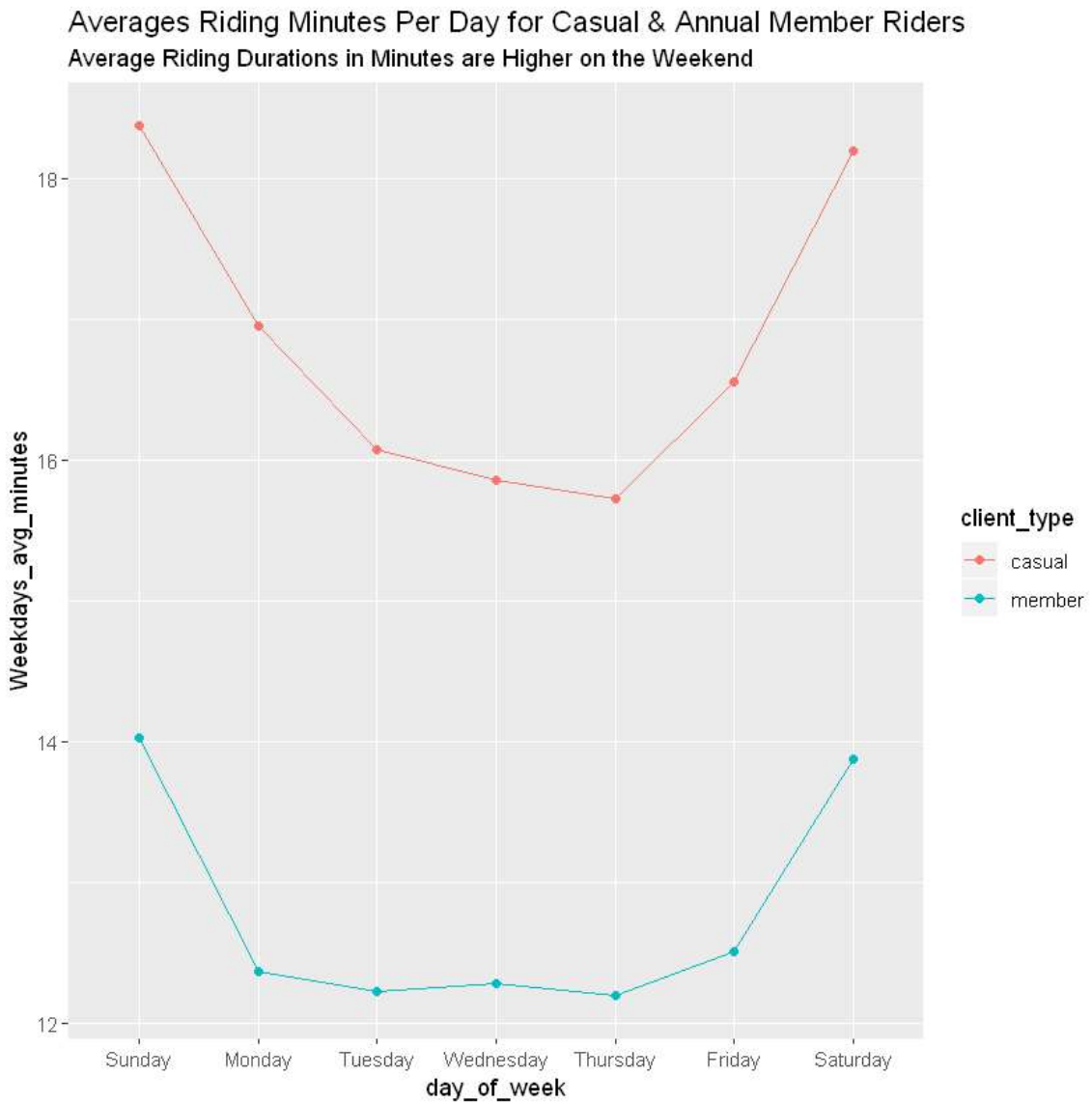
```
xorder2 <- c("Sunday","Monday","Tuesday",
             "Wednesday","Thursday","Friday",
               "Saturday")
day_minuntes$day_of_week <- factor(day_minuntes$day_of_week,
        levels=xorder2)
```

```
# Grouped Bars
ggplot(day_minuntes, aes(x=day_of_week, y=Weekdays_avg_minutes,
    fill=client_type, group = client_type)) +
labs(title = "Averages Riding Minutes Per Day of a week: Casual Riders
        vs Member Riders",
     subtitle = "Daily Average Riding Durations are higher during the
        Weekend")+
geom_text(aes(x=day_of_week, y=Weekdays_avg_minutes,
              label=Weekdays_avg_minutes, group = client_type),
          position = position_dodge(width = 1),
          vjust=-0.5, size=3.5)+
    geom_bar(position="dodge", stat="identity")+
theme_bw()
```




```
# Order x labels
xorder2 <- c("Sunday","Monday","Tuesday",
             "Wednesday","Thursday","Friday",
```

```
                "Saturday")
day_minuntes$day_of_week <- factor(day_minuntes$day_of_week,
        levels=xorder2)



# Grouped line plots
ggplot(day_minuntes, aes(x=day_of_week, y=Weekdays_avg_minutes,
                         group = client_type, color = client_type)) +
labs(title = "Averages Riding Minutes Per Day for Casual & Annual
        Member Riders",
     subtitle = "Average Riding Durations in Minutes are Higher on the
        Weekend")+


    geom_line()+ geom_point()
```

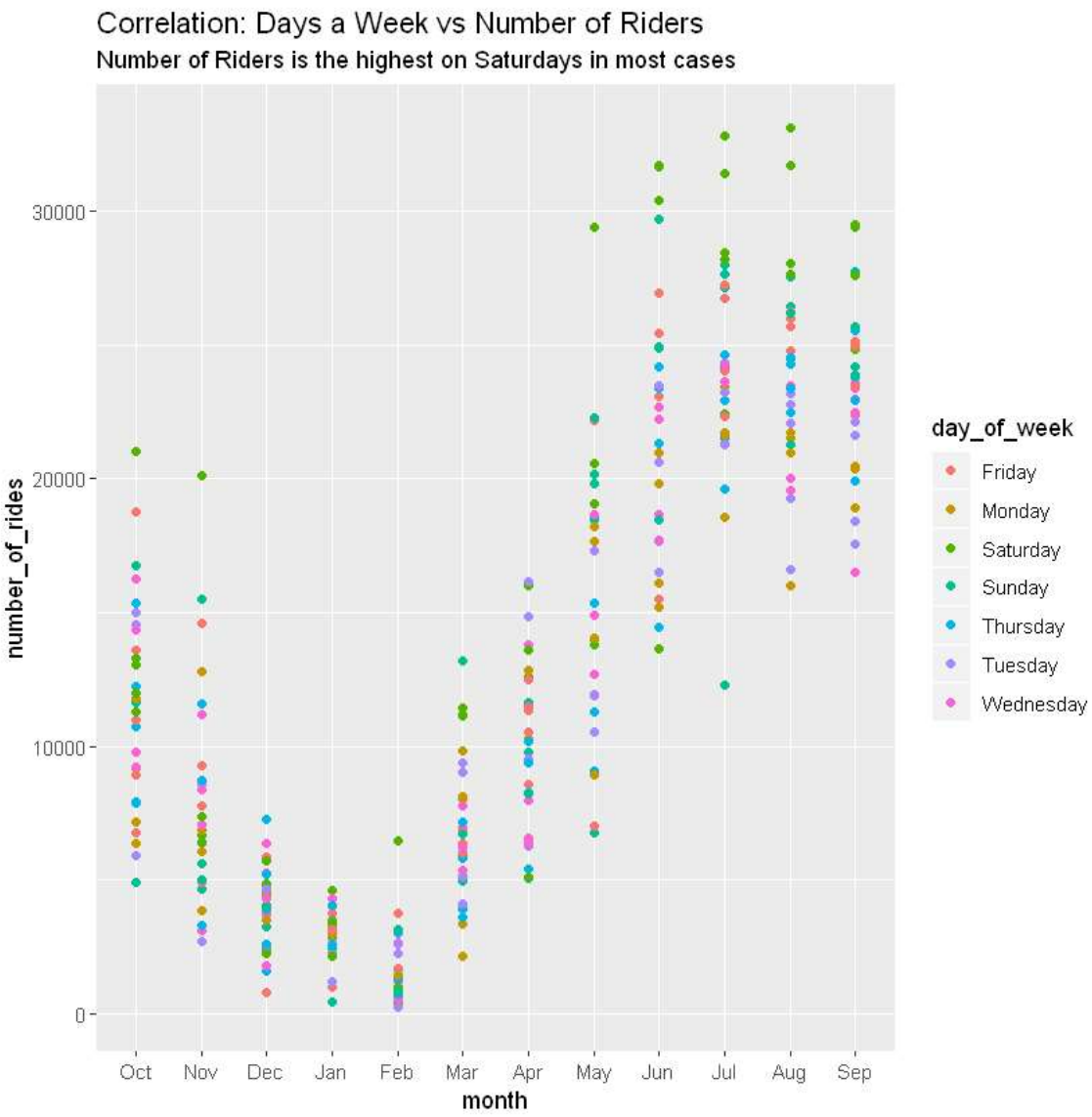


6. Correlation between Weekdays and Number of Rides

- In most cases, Saturday has the highest number of riders; especially during Summer the number is way higher. Sunday has the seocnd highest numbers of riders followed by Friday.
- However, there is no meaningful correlation between the days of the week and number of rides. The conclusion here is that number of rides are higher during the weekends in most cases (rain or shine)

```
vec6v <- c(dim(cleaned_bike_df))
number6v <- list(rep(1, vec6v[1]))
cor6v_df <- data.frame(day_date = cleaned_bike_df$started_at, cnt =
        number6v)
names(cor6v_df)[2] <- 'number_of_rides'
cor6v_df$day_date <- as.Date(cor6v_df$day_date, format = "%Y-%m-%d")

cor6v_df <- cor6v_df%>% group_by(day_date) %>%
        summarize(number_of_rides = sum(number_of_rides))
cor6v_df$month <- format(cor6v_df$day_date,"%Y-%m")
cor6v_df <- arrange(cor6v_df, month)
cor6v_df$month <- format(cor6v_df$day_date,"%m")

cor6v_df$month <- as.integer(cor6v_df$month)

#months vector assuming 1st month is Jan.
mymonths <- c("Jan","Feb","Mar",
              "Apr","May","Jun",
              "Jul","Aug","Sep",
              "Oct","Nov","Dec")
#add abbreviated month name
cor6v_df$month <- mymonths[cor6v_df$month]



cor6v_df$day_of_week <- weekdays(as.Date(cor6v_df$day_date))

xorder <- c("Oct","Nov","Dec",
            "Jan","Feb","Mar",
              "Apr","May","Jun",
              "Jul","Aug","Sep" )
cor6v_df$month <- factor(cor6v_df$month, levels=xorder)

# Grouped Bars
ggplot(cor6v_df, aes(y=number_of_rides, x=month, color = day_of_week))
        +
```

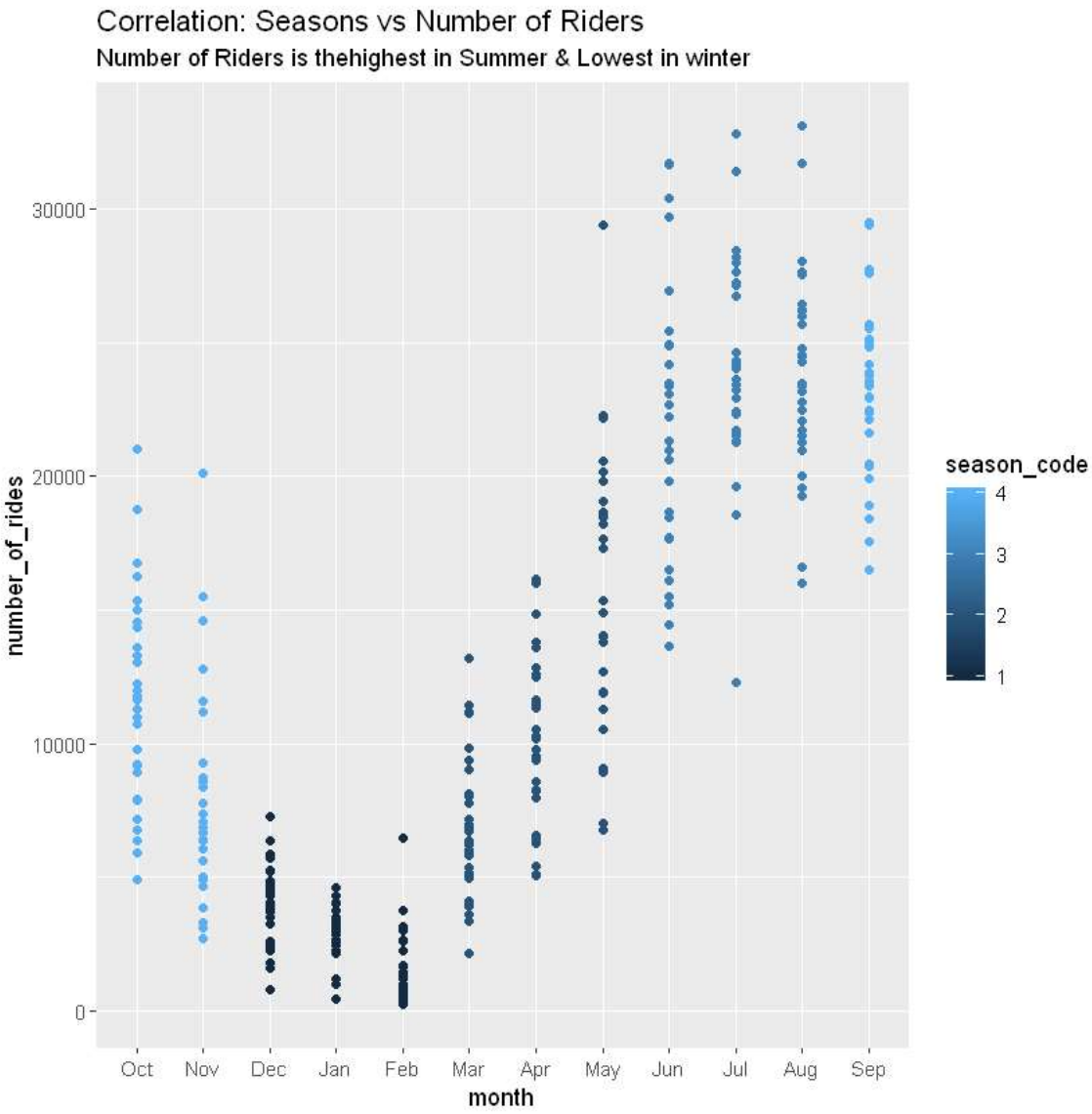
```
labs(title = "Correlation: Days a Week vs Number of Riders",
    subtitle = "Number of Riders is the highest on Saturdays in most
        cases")+
    geom_point(stat="identity")
```



7. Correlation between Seasons and Number of Rides

- The scatter portrayed in the figure below shows an upward trend starting from the end of Winter to Summer.
- The peak number of rides are registered in summer, particularly in the months of July and August.
- The lowest number of rides are recored in Winter, in the months of January and February.
- There is a strong correlation between the number of rides and seasons. As the season gets warmer, the number of rides increases, and vice versa.

```
#add abbreviated month name
mymonths <- c("Jan","Feb","Mar",
              "Apr","May","Jun",
              "Jul","Aug","Sep",
              "Oct","Nov","Dec")
cor8_df$month <- mymonths[cor8_df$month2]

# Order months
xorder <- c("Oct","Nov","Dec",
            "Jan","Feb","Mar",
              "Apr","May","Jun",
              "Jul","Aug","Sep" )
cor8_df$month <- factor(cor8_df$month, levels=xorder)



# Grouped Bars
ggplot(cor8_df, aes(y=number_of_rides, x=month, color = season_code))
        +
labs(title = "Correlation: Seasons vs Number of Riders",
    subtitle = "Number of Riders is thehighest in Summer & Lowest in
        winter")+
    geom_point(stat="identity")
```



8. Correlation Between Riding Durations and Seasons

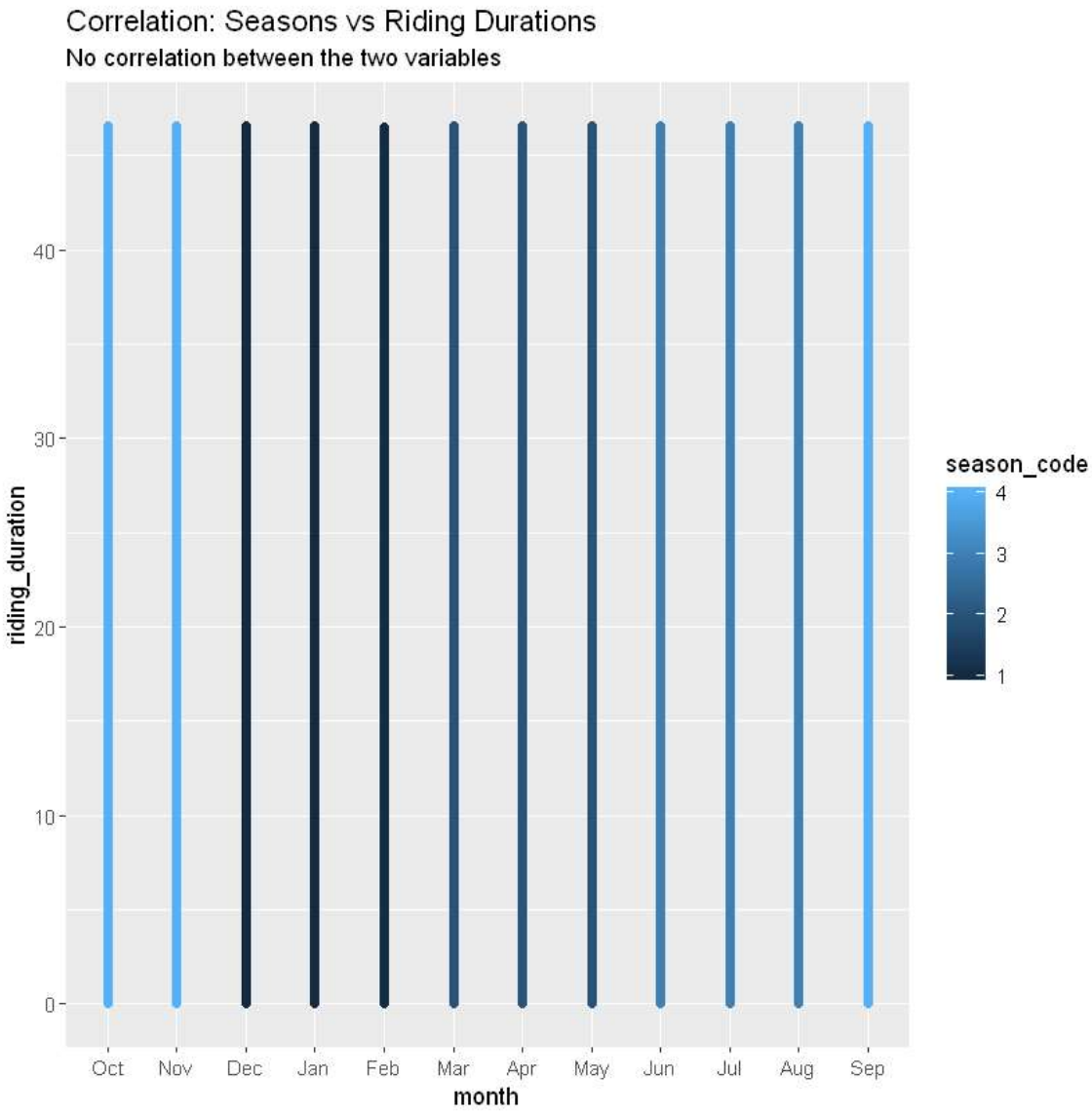
- The seasons do not have a meaningful correlation with daily average of riding durations.

```
#add abbreviated month name
cor9_df$month <- mymonths[cor9_df$month]
# Order months
cor9_df$month <- factor(cor9_df$month, levels=xorder)

head(cor9_df)
```

started_at	riding_duration	month	season_code
2021-01-23	10.42	Jan	1
2021-01-27	4.07	Jan	1

started_at	riding_duration	month	season_code
2021-01-21	1.33	Jan	1
2021-01-07	11.70	Jan	1
2021-01-23	0.72	Jan	1
2021-01-04	5.58	Jan	1

```
# Grouped Bars
ggplot(cor9_df, aes(x=month, y=riding_duration, color = season_code))
        +
labs(title = "Correlation: Seasons vs Riding Durations",
    subtitle = "No correlation between the two variables")+
    geom_point(stat="identity")
```

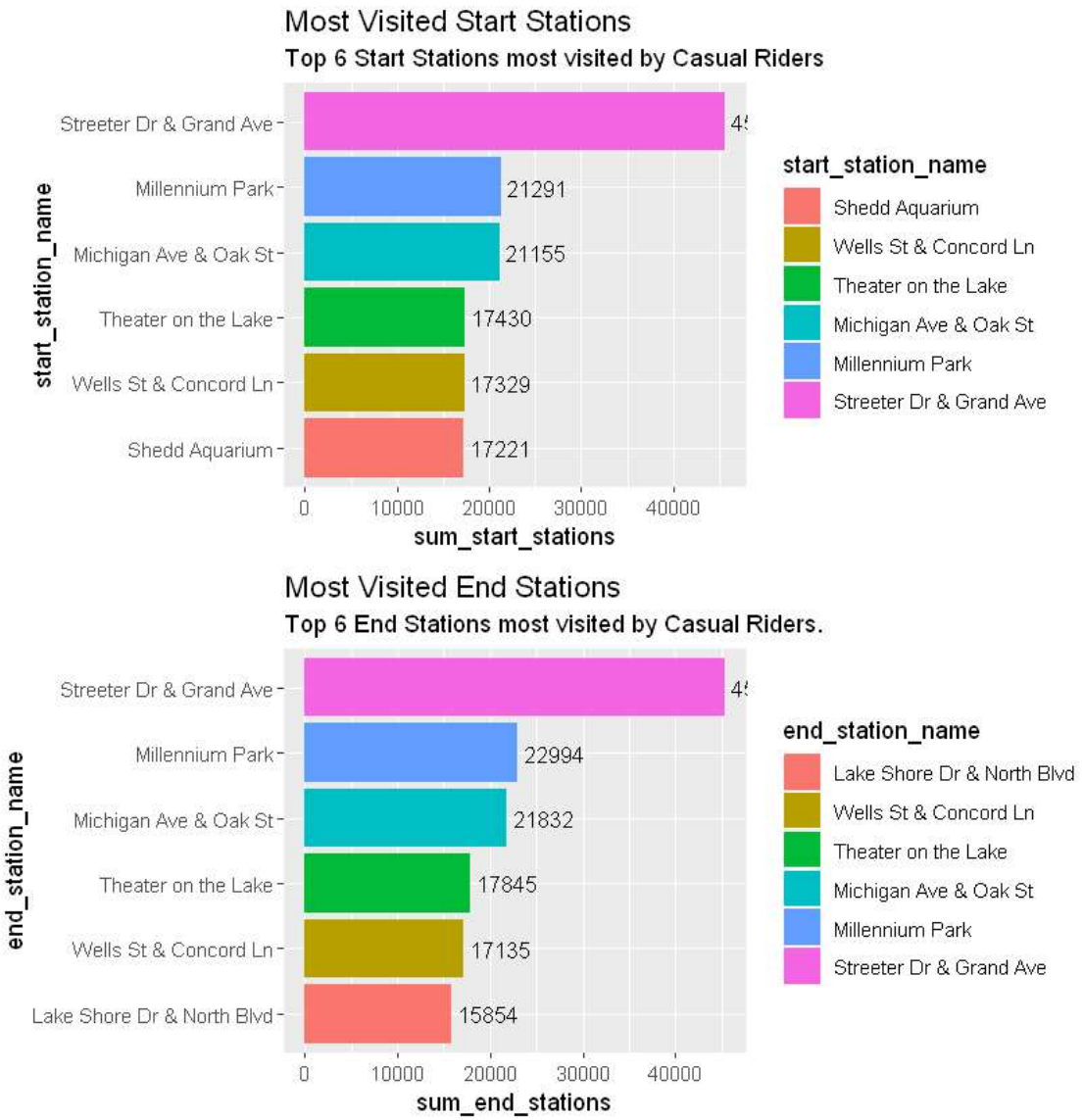


9. Top Six Stations Most Visited by Casual Riders

- Unnamed start or end stations were counted out in this analysis

```
# Start Stations
ss_df9 <- ssc_df[ssc_df$start_station_name!='not_provided',]
ss_df9 <- head(ss_df9,6)


# Order Start Stations
ss_order <- c("Streeter Dr & Grand Ave","Millennium Park","Dec",
              "Michigan Ave & Oak St","Theater on the Lake","Wells St &
        Concord Ln",
                "Shedd Aquarium")
ss_df9$start_station_name <- factor(ss_df9$start_station_name,
        levels=rev(ss_order))


p<-ggplot(data=ss_df9, aes(x=start_station_name,
                  y=sum_start_stations, group=start_station_name, fill
        =start_station_name)) +
  labs(title = "Most Visited Start Stations",
       subtitle = "Top 6 Start Stations most visited by Casual
        Riders")+
    geom_text(aes(x=start_station_name, y=sum_start_stations,
                label=sum_start_stations, group = start_station_name),
          position = position_dodge(width = 1),
          hjust=-0.1, size=3.5)+
      geom_bar(position="dodge", stat="identity")



# End Stations
es_df9 <- esc_df[esc_df$end_station_name!='not_provided',]
es_df9 <- head(es_df9,6)


# Order Start Stations
es_order <- c("Streeter Dr & Grand Ave","Millennium Park",
              "Michigan Ave & Oak St","Theater on the Lake","Wells St
        & Concord Ln",
                "Lake Shore Dr & North Blvd")
es_df9$end_station_name <- factor(es_df9$end_station_name,
        levels=rev(es_order))


# Bar plots
p2<-ggplot(data=es_df9, aes(x=end_station_name,
                  y=sum_end_stations, group=end_station_name, fill
        =end_station_name)) +
  labs(title = "Most Visited End Stations",
```

```
        subtitle = "Top 6 End Stations most visited by Casual Riders.")+
    geom_text(aes(x=end_station_name, y=sum_end_stations,
                  label=sum_end_stations, group = end_station_name),
              position = position_dodge(width = 1),
              hjust=-0.1, size=3.5)+
  geom_bar(position="dodge", stat="identity")

ggarrange(p + coord_flip(), p2 + coord_flip(), ncol=1)
```



Step-6: ACT

Now that I have finished creating the pertinent visualizations, it is time to act on my findings. I have to prepare the deliverables that Morena asked me to create, including the three top recommendations based on my analysis. I used the following Case Study Roadmap as a guide.

Case Study Roadmap - Act

Guiding questions

- What are my final conclusions based on my analysis?
- How could my team and business apply my insights?
- What next steps would I or my stakeholders take based on my findings?
- Is there additional data I could use to expand on my findings?

Key tasks

1. Creating my portfolio.
2. Adding my case study.
3. Practicing presenting my case study to a friend or family member.

Deliverable

- My top three recommendations based on my analysis

A. conclusions

Based on the insights uncovered/mined from the data by the extensive analysis carried out, the following conclusions were drawn:

1. In the data analyzed after apropos cleaning, 57.3% of the riders are annual member of the Bike-share company in Chicago; whereas the remaining 42.7% are casual riders. Nonetheless, the member riders have only slightly greater total annual riding duration (34, 245, 797.23 minutes) than that of casual riders (34, 064, 953.52 minutes).
2. The casual riders have longer daily riding average (16.97 minutes) than the member riders (12.72 minutes) do have. Likewise, the casual riders have better monthly riding averages than the member riders through the whole observation period (12 months: October, 2020 to the end of September, 2021).

3. Casual riders have better riding lengths on all days of the week compared to the member riders. In most cases, the number of riders of both types is much higher during the weekend (the lion-share of it belongs to Saturday) than the weekdays.
4. A strong correlation was uncovered between the seasons (Winter, Spring, Summer, & Fall vis-a-vis North America) and the number of riders. It was found out that the number of riders of both types start going up by the end of Winter and reaches its peak by late Summer, and conversely. Putting it another way, as the temperature increases, the number of riders increase sharply. Then, it starts going down at the start of Fall and plummets during the coldest season, Winter. Despite the season-wise variation of rider's number, the number of casual riders is consistently higher than that of member riders under all scenarios.
5. After thoroughly analyzing the locations of the top six start or end stations most visited by casual riders in Chicago, it was discovered that almost all of them are closer to reactional centers like theatre, amphitheater, gymnasium, et cetera.

B. Recommendations

Based on the insights derived from the data, the following three marketing strategies or approaches are recommended in order to convert casual riders to annual member riders.

1. Intensify promotional/advertising works in reactional centers like theatre, amphitheater, gymnasium, cafeterias, restaurants, ... situated nearby most visited start or end stations.
2. Introduce seasonal membership discounts (good discount during the summer would be so encouraging) and weekend-based (especially on Saturdays) riding price discount for members.
3. Create partnership with service centers nearby most visited stations so as to reward newly joining members with free gymnasium, restaurant, or theatre tickets.

"Wih Patience We Can Dissect An Ant and Reach Its Heart"

~END~