Cyclistic

Jayron Andrade

2023-10-10

Divvy Case Study

This analysis is based on the Divvy case study "'Sophisticated, Clear, and Polished': Divvy and Data Visualization" written by Kevin Hartman (found here: https://artscience.blog/home/divvy-dataviz-case-study (https://artscience.blog/home/divvy-dataviz-case-study)). The purpose of this script is to consolidate downloaded Divvy data into a single dataframe and then conduct simple analysis to help answer the key question: **"In what ways do members and casual riders use Divvy bikes differently?"**.

Install Required Packages:

- tidyverse for data import and wrangling
- lubridate for date functions
- ggplot for visualization

Importing The Packages:

```
library(tidyverse)  #helps wrangle data
library(lubridate)  #helps wrangle date attributes
library(ggplot2)  #helps visualize data
setwd("C:/Users/Andra/Documents/divvy-tripdata") #sets your working directory to simplify calls to data
```

Collect Data

Upload Divvy Datasets (csv files)

```
df_2022_01 <- read_csv("divvy-tripdata/202201-divvy-tripdata.csv")
df_2022_02 <- read_csv("divvy-tripdata/202202-divvy-tripdata.csv")
df_2022_03 <- read_csv("divvy-tripdata/202203-divvy-tripdata.csv")
df_2022_04 <- read_csv("divvy-tripdata/202204-divvy-tripdata.csv")
df_2022_05 <- read_csv("divvy-tripdata/202205-divvy-tripdata.csv")
df_2022_06 <- read_csv("divvy-tripdata/202206-divvy-tripdata.csv")
df_2022_07 <- read_csv("divvy-tripdata/202207-divvy-tripdata.csv")
df_2022_08 <- read_csv("divvy-tripdata/202208-divvy-tripdata.csv")
df_2022_09 <- read_csv("divvy-tripdata/202209-divvy-publictripdata.csv")
df_2022_10 <- read_csv("divvy-tripdata/202210-divvy-tripdata.csv")
df_2022_11 <- read_csv("divvy-tripdata/202211-divvy-tripdata.csv")
df_2022_12 <- read_csv("divvy-tripdata/202201-divvy-tripdata.csv")
```

Wrangle Data And Combine Into a Single File

Compare column names each of the files While the names don't have to be in the same order, they DO need to match perfectly before we can use a command to join

them into one file

```
colnames(df_2022_01)
colnames(df_2022_02)
colnames(df_2022_03)...
colnames(df_2022_12)
```

Inspect the dataframes and look for incongruencies

```
str(df_2022_01)
str(df_2022_02)
str(df_2022_03)...
str(df_2022_12)
```

Stack individual quarter's data frames into one big data frame

```
all_trips <- bind_rows(df_2022_01, df_2022_02, df_2022_03, df_2022_04,
                       df_2022_05, df_2022_06, df_2022_07, df_2022_08,
                       df_2022_09, df_2022_10, df_2022_11, df_2022_12)
```

Remove the latitude and longitude fields as we do not use this data for this analysis

```
all_trips <- all_trips %>%
  select(-c(start_lat, start_lng, end_lat, end_lng))
```

Clean Up And Add Data To Prepare For Analysis

Inspect The New Table That Has Been Created

List of column names

```
colnames(all_trips) #List of column names
```

```
## [1] "ride_id"            "rideable_type"      "started_at"
## [4] "ended_at"           "start_station_name" "start_station_id"
## [7] "end_station_name"   "end_station_id"     "member_casual"
```

How many rows are in data frame?

```
nrow(all_trips)
```

```
## [1] 5589681
```

Dimensions of the data frame?

```
dim(all_trips)
```

```
## [1] 5589681       9
```

See the first 3 rows of data frame

```
head(all_trips, 3)
```

```
## # A tibble: 3 × 9
##   ride_id          rideable_type started_at          ended_at
##   <chr>            <chr>         <dttm>              <dttm>
## 1 C2F7DD78E82EC875 electric_bike 2022-01-13 11:59:47 2022-01-13 12:02:44
## 2 A6CF8980A652D272 electric_bike 2022-01-10 08:41:56 2022-01-10 08:46:17
## 3 BD0F91DFF741C66D classic_bike  2022-01-25 04:53:40 2022-01-25 04:58:01
## # ℹ 5 more variables: start_station_name <chr>, start_station_id <chr>,
## #   end_station_name <chr>, end_station_id <chr>, member_casual <chr>
```

See list of columns and data types (numeric, character, etc)

```
str(all_trips)
```

```
## tibble [5,589,681 × 9] (S3: tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:5589681] "C2F7DD78E82EC875" "A6CF8980A652D272" "BD0F91DFF741C66D" "CBB80ED419105406" ...
##  $ rideable_type     : chr [1:5589681] "electric_bike" "electric_bike" "classic_bike" "classic_bike" ...
##  $ started_at        : POSIXct[1:5589681], format: "2022-01-13 11:59:47" "2022-01-10 08:41:56" ...
##  $ ended_at          : POSIXct[1:5589681], format: "2022-01-13 12:02:44" "2022-01-10 08:46:17" ...
##  $ start_station_name: chr [1:5589681] "Glenwood Ave & Touhy Ave" "Glenwood Ave & Touhy Ave" "Sheffield Ave & Fullerton Ave" "Clark St & Bryn Mawr Ave" ...
##  $ start_station_id  : chr [1:5589681] "525" "525" "TA1306000016" "KA1504000151" ...
##  $ end_station_name  : chr [1:5589681] "Clark St & Touhy Ave" "Clark St & Touhy Ave" "Greenview Ave & Fullerton Ave" "Paulina St & Montrose Ave" ...
##  $ end_station_id    : chr [1:5589681] "RP-007" "RP-007" "TA1307000001" "TA1309000021" ...
##  $ member_casual     : chr [1:5589681] "casual" "casual" "member" "casual" ...
```

Number Of Members (casual or member)

```
table(all_trips$member_casual)
```

```
##
##  casual  member
## 2295658 3294023
```

Statistical summary of data. Mainly for numerics

```
summary(all_trips)
```

```
##    ride_id          rideable_type        started_at
##  Length:5589681     Length:5589681     Min.   :2022-01-01 00:00:05.00
##  Class :character   Class :character   1st Qu.:2022-05-22 18:08:27.00
##  Mode  :character   Mode  :character   Median :2022-07-16 21:55:37.00
##                                        Mean   :2022-07-12 02:57:24.82
##                                        3rd Qu.:2022-09-09 05:50:28.00
##                                        Max.   :2022-11-30 23:56:11.00
##     ended_at                      start_station_name start_station_id
##  Min.   :2022-01-01 00:01:48.00   Length:5589681     Length:5589681
##  1st Qu.:2022-05-22 18:31:50.00   Class :character   Class :character
##  Median :2022-07-16 22:16:59.00   Mode  :character   Mode  :character
##  Mean   :2022-07-12 03:16:58.36
##  3rd Qu.:2022-09-09 06:09:47.00
##  Max.   :2022-12-01 11:45:53.00
##  end_station_name   end_station_id     member_casual
##  Length:5589681     Length:5589681     Length:5589681
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
```

There are a few problems we will need to fix:

- 1. The data can only be aggregated at the ride-level, which is too granular. We will want to add some additional columns of data – such as day, month, year – that provide additional opportunities to aggregate the data.
- 2. We will want to add a calculated field for length of ride since the 2020Q1 data did not have the “tripduration” column. We will add “ride_length” to the entire dataframe for consistency.
- 3. There are some rides where tripduration shows up as negative, including several hundred rides where Divvy took bikes out of circulation for Quality Control reasons. We will want to delete these rides.

Add columns that list the date, month, day, and year of each ride

This will allow us to aggregate ride data for each month, day, or year … before completing these operations we could only aggregate at the ride level

more on date formats in R found at that link (https://www.statmethods.net/input/dates.html)

```
all_trips$date <- as.Date(all_trips$started_at) #The default format is yyyy-mm-dd
all_trips$month <- format(as.Date(all_trips$date), "%m")
all_trips$day <- format(as.Date(all_trips$date), "%d")
all_trips$year <- format(as.Date(all_trips$date), "%Y")
all_trips$day_of_week <- format(as.Date(all_trips$date), "%A")
```

Add a “ride_length” calculation to all_trips (in seconds) more information link (https://stat.ethz.ch/R-manual/R-devel/library/base/html/difftime.html)

```
all_trips$ride_length <- difftime(all_trips$ended_at,all_trips$started_at)
```

Inspect the structure of the columns

```
str(all_trips)
```

```
## tibble [5,589,681 × 15] (S3: tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:5589681] "C2F7DD78E82EC875" "A6CF8980A652D272" "BD0F91DFF741C6
6D" "CBB80ED419105406" ...
##  $ rideable_type     : chr [1:5589681] "electric_bike" "electric_bike" "classic_bike" "class
ic_bike" ...
##  $ started_at        : POSIXct[1:5589681], format: "2022-01-13 11:59:47" "2022-01-10 08:41:5
6" ...
##  $ ended_at          : POSIXct[1:5589681], format: "2022-01-13 12:02:44" "2022-01-10 08:46:1
7" ...
##  $ start_station_name: chr [1:5589681] "Glenwood Ave & Touhy Ave" "Glenwood Ave & Touhy Ave"
"Sheffield Ave & Fullerton Ave" "Clark St & Bryn Mawr Ave" ...
##  $ start_station_id  : chr [1:5589681] "525" "525" "TA1306000016" "KA1504000151" ...
##  $ end_station_name  : chr [1:5589681] "Clark St & Touhy Ave" "Clark St & Touhy Ave" "Greenv
iew Ave & Fullerton Ave" "Paulina St & Montrose Ave" ...
##  $ end_station_id    : chr [1:5589681] "RP-007" "RP-007" "TA1307000001" "TA1309000021" ...
##  $ member_casual     : chr [1:5589681] "casual" "casual" "member" "casual" ...
##  $ date              : Date[1:5589681], format: "2022-01-13" "2022-01-10" ...
##  $ month             : chr [1:5589681] "01" "01" "01" "01" ...
##  $ day               : chr [1:5589681] "13" "10" "25" "04" ...
##  $ year              : chr [1:5589681] "2022" "2022" "2022" "2022" ...
##  $ day_of_week       : chr [1:5589681] "quinta-feira" "segunda-feira" "terça-feira" "terça-f
eira" ...
##  $ ride_length       : 'difftime' num [1:5589681] 177 261 261 896 ...
##   ..- attr(*, "units")= chr "secs"
```

Convert “ride_length” from Factor to numeric so we can run calculations on the data

```
is.factor(all_trips$ride_length) #FALSE
all_trips$ride_length <- as.numeric(as.character(all_trips$ride_length))
is.numeric(all_trips$ride_length) # TRUE
```

Remove “bad” data

###The dataframe includes a few hundred entries when bikes were taken out of docks and checked for quality by Divvy or ride_length was negative ###We will create a new version of the dataframe (v2) since data is being removed ####https://www.datasciencemadesimple.com/delete-or-drop-rows-in-r-with-conditions-2/

(https://www.datasciencemadesimple.com/delete-or-drop-rows-in-r-with-conditions-2/)

```
all_trips_v2 <- all_trips[!(all_trips$start_station_name == "HQ QR" | all_trips$ride_length<
0),]
```

CONDUCT DESCRIPTIVE ANALYSIS

Descriptive analysis on ride_length (all figures in seconds)

Min, Median, Meand and Max

```
summary(all_trips_v2$ride_length) #straight average (total ride length / rides)
```

```
##   Length    Class     Mode
##  5589581 difftime  numeric
```

Compare members and casual users

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = mean) # comparison of th
e mean between members
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual           1909.7523 secs
## 2                     member            774.7864 secs
```

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = median) # comparison of
the median between members
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                 820 secs
## 2                     member                 538 secs
```

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = max) # comparison of the
max between members
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual             2483235 secs
## 2                     member               93594 secs
```

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = min) # comparison of the
min between members
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                   0 secs
## 2                     member                   0 secs
```

See the average ride time by each day for members vs casual users

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual + all_trips_v2$day_of_week, FUN
= mean)
```

```
##    all_trips_v2$member_casual all_trips_v2$day_of_week all_trips_v2$ride_length
## 1                      casual                  domingo           2219.6913 secs
## 2                      member                  domingo            860.0324 secs
## 3                      casual             quarta-feira           1620.8342 secs
## 4                      member             quarta-feira            735.0936 secs
## 5                      casual             quinta-feira           1680.7756 secs
## 6                      member             quinta-feira            749.3536 secs
## 7                      casual                   sábado           2129.3055 secs
## 8                      member                   sábado            866.5549 secs
## 9                      casual            segunda-feira           1915.5322 secs
## 10                     member            segunda-feira            746.3288 secs
## 11                     casual              sexta-feira           1832.6507 secs
## 12                     member              sexta-feira            762.2044 secs
## 13                     casual              terça-feira           1696.2522 secs
## 14                     member              terça-feira            738.1760 secs
```

Notice that the days of the week are out of order. Let's fix that.

```
all_trips_v2$day_of_week <- ordered(all_trips_v2$day_of_week, levels=c("segunda-feira", "terça-
feira", "quarta-feira", "quinta-feira", "sexta-feira", "sábado", "domingo"))
```

Now, let's run the average ride time by each day for members vs casual users

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual + all_trips_v2$day_of_week, FUN
= mean)
```

```
##    all_trips_v2$member_casual all_trips_v2$day_of_week all_trips_v2$ride_length
## 1                      casual            segunda-feira           1915.5322 secs
## 2                      member            segunda-feira            746.3288 secs
## 3                      casual              terça-feira           1696.2522 secs
## 4                      member              terça-feira            738.1760 secs
## 5                      casual             quarta-feira           1620.8342 secs
## 6                      member             quarta-feira            735.0936 secs
## 7                      casual             quinta-feira           1680.7756 secs
## 8                      member             quinta-feira            749.3536 secs
## 9                      casual              sexta-feira           1832.6507 secs
## 10                     member              sexta-feira            762.2044 secs
## 11                     casual                   sábado           2129.3055 secs
## 12                     member                   sábado            866.5549 secs
## 13                     casual                  domingo           2219.6913 secs
## 14                     member                  domingo            860.0324 secs
```

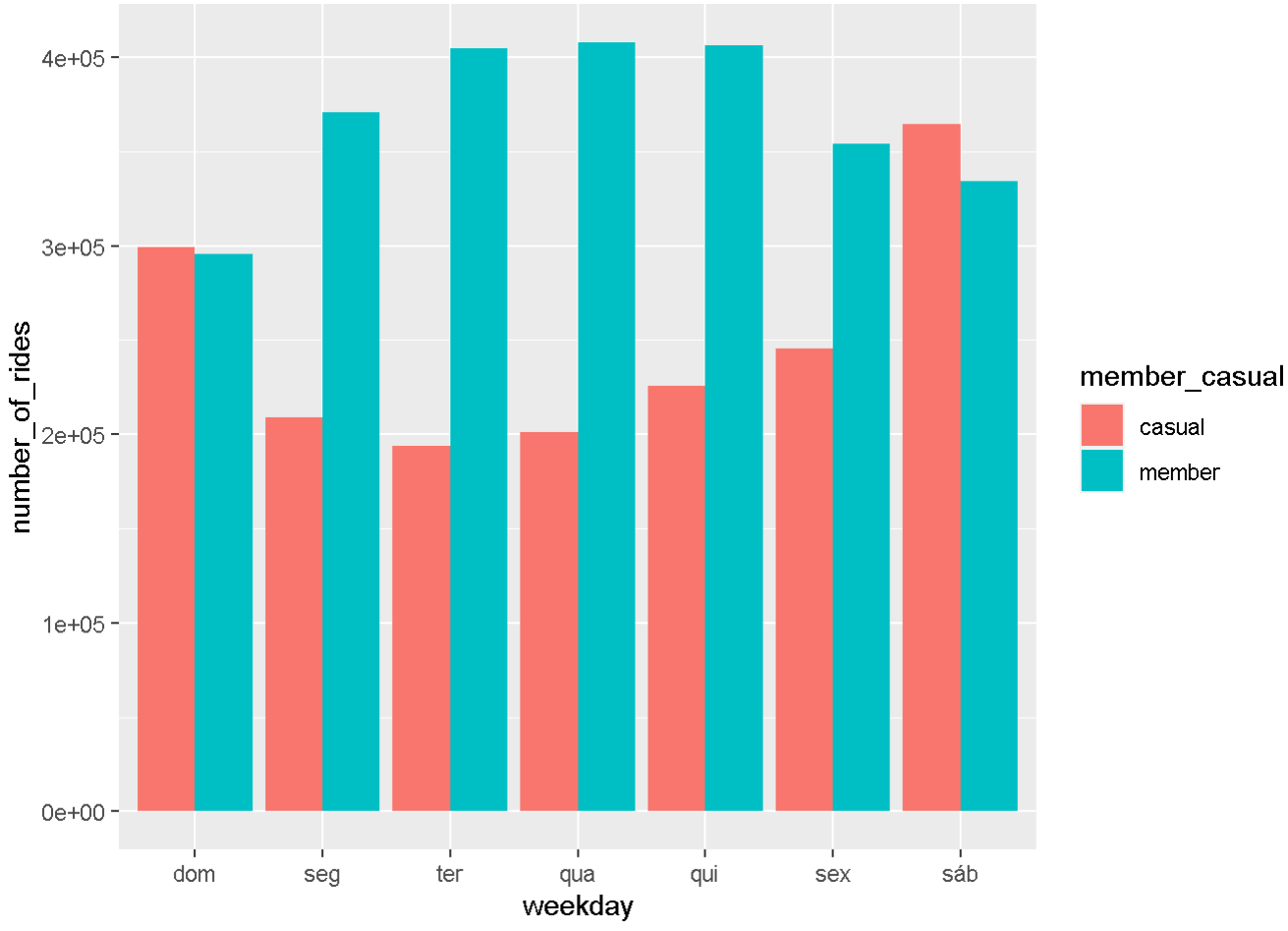
analyze ridership data by type and weekday

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

```
## # A tibble: 15 × 4
## # Groups:   member_casual [3]
##    member_casual weekday number_of_rides average_duration
##    <chr>         <ord>             <int> <drtn>
##  1 casual        dom              331999 2219.6913 secs
##  2 casual        seg              234351 1915.5322 secs
##  3 casual        ter              219825 1696.2522 secs
##  4 casual        qua              228442 1620.8342 secs
##  5 casual        qui              255950 1680.7756 secs
##  6 casual        sex              277991 1832.6507 secs
##  7 casual        sáb              404093 2129.3055 secs
##  8 member        dom              324346  860.0324 secs
##  9 member        seg              403036  746.3288 secs
## 10 member        ter              440662  738.1760 secs
## 11 member        qua              445306  735.0936 secs
## 12 member        qui              445005  749.3536 secs
## 13 member        sex              389700  762.2044 secs
## 14 member        sáb              368857  866.5549 secs
## 15 <NA>          <NA>             820018        NA secs
```

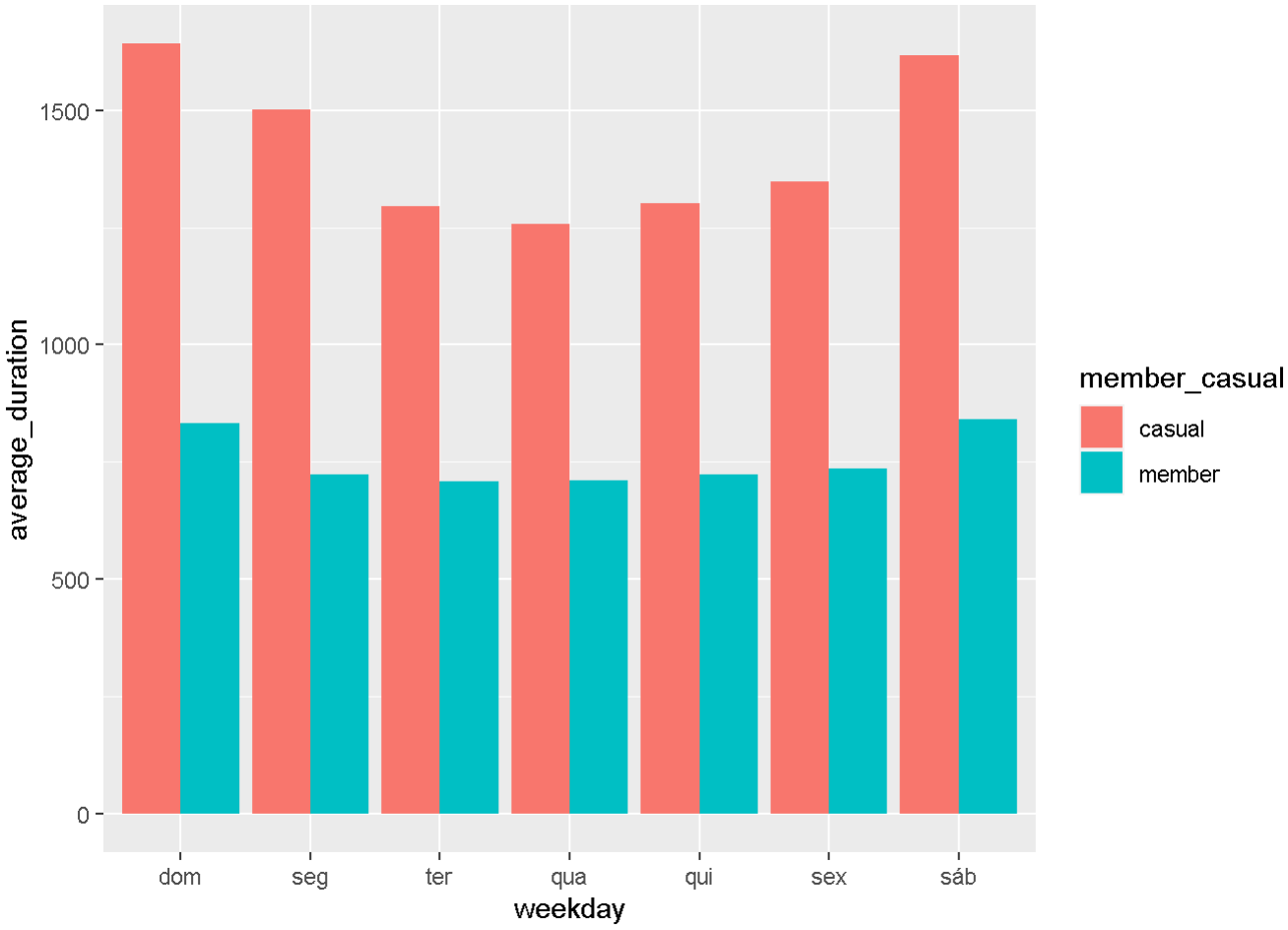
Let's visualize the number of rides by rider type

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



Let's create a visualization for average duration

```
## `summarise()` has grouped output by 'member_casual'. You can override using the
## `.groups` argument.
## Don't know how to automatically pick scale for object of type <difftime>.
## Defaulting to continuous.
```



EXPORT SUMMARY FILE FOR FURTHER ANALYSIS

```
#counts <- aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual + all_trips_v2$day_o
f_week, FUN = mean)
#write.csv(counts, file = 'C:/Users/Andra/Desktop/avg_ride_length.csv')
```

Exporting the “all_trips_v2”

```
#write.csv(all_trips_v2, file = 'C:/Users/Andra/Desktop/all_trips_cleaned.csv')
```