Cyclistic Bike-Share Analysis: Converting Casual Riders to Members

ASEGID AYELE

2025-08-03

Contents

1 Cyclistic Bike-Share Capstone Project

Welcome to my data analysis capstone project for Cyclistic, a fictional bike-share company in Chicago. This report follows the **Ask, Prepare, Process, Analyze, Share, Act** framework to answer a key business question:

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

This analysis aims to inform a targeted marketing strategy to convert casual riders into annual members, supporting Cyclistic's long-term profitability.

1.1 1. Ask: Define the Business Task

1.1.1 Business Objective

The primary objective is to analyze historical trip data to identify behavioral differences between casual riders (pay-per-ride or day-pass users) and annual members (subscribers). These insights will then be used to design and propose a targeted marketing strategy aimed at converting casual riders into annual members.

1.1.2 Key Questions

To achieve the business objective, the analysis will address the following key questions:

- How do ride frequency, duration, and timing differ between user types?
- Are members more likely to use bikes for commuting (weekdays)? Do casual riders prefer weekends?
- Are there seasonal or daily trends in usage patterns for each user type?

1.1.3 Stakeholders

This project is being conducted for **Lily Moreno, the Director of Marketing** at Cyclistic. Her team is responsible for designing the new marketing strategy. The **Cyclistic Executive Team** will need to approve the recommendations, emphasizing the need for compelling data insights and professional visualizations. Other key stakeholders include the **Marketing Analytics Team**.

1.2 2. Prepare: Data Sources

1.2.1 Dataset Overview

- **Source**: Divvy Trip Data (public dataset, used as proxy for Cyclistic)
- **Time Period**: Q1 2019 and Q1 2020 (January–March)
- **Files Used**:
  - `Divvy_Trips_2019_Q1.csv`
  - `Divvy_Trips_2020_Q1.csv`
- **Key Variables**: `trip_id`, `start_time`, `end_time`, `bike_type`, `usertype` ("Subscriber" or "Customer")

1.2.2 ROCCC Analysis

CRITERION	ASSESSMENT
Reliability	High - official data from Motivate Intl. Inc.
Original	Yes - primary source
Comprehensive	Moderate - lacks user demographics due to privacy
Current	Historical, but sufficient for trend analysis
Cited	Yes - licensed under Divvy's Data License Agreement

Note: No personally identifiable information (PII) is available.

1.2.3 Data Loading and Initial Inspection

```r
# Load necessary libraries
library(tidyverse) # For data manipulation and visualization (ggplot2, dplyr)
library(lubridate) # For working with dates and times
library(scales) # For formatting numbers in plots
library(janitor) # For cleaning column names

# Set default ggplot2 theme for better aesthetics and a clean, professional look
theme_set(theme_light())
```

```r
# Load raw data from the specified CSV files.
# The file paths are adjusted to correctly point to the `data/` subfolder.
# The `../` navigates up one directory from your RMD file's location.
trips_2019 <- read_csv("../data/Divvy_Trips_2019_Q1.csv", show_col_types = FALSE)
trips_2020 <- read_csv("../data/Divvy_Trips_2020_Q1.csv", show_col_types = FALSE)

# Clean column names for easier access using janitor::clean_names()
trips_2019 <- clean_names(trips_2019)
trips_2020 <- clean_names(trips_2020)

# Standardize column names across both datasets to ensure consistency before combining.
# 'start_time' becomes 'started_at'
# 'end_time' becomes 'ended_at'
# 'usertype' becomes 'member_casual'
trips_2019 <- trips_2019 %>%
  rename(
    started_at = start_time,
    ended_at = end_time,
    member_casual = usertype
  )

# Combine datasets into a single data frame.
# bind_rows is used to handle potential column inconsistencies gracefully.
all_trips <- bind_rows(trips_2019, trips_2020)

# Display a summary of the combined data
glimpse(all_trips)
```

```
## Rows: 791,956
## Columns: 22
## $ trip_id           <dbl> 21742443, 21742444, 21742445, 21742446, 21742447, 2~
## $ started_at        <dttm> 2019-01-01 00:04:37, 2019-01-01 00:08:13, 2019-01-~
## $ ended_at          <dttm> 2019-01-01 00:11:07, 2019-01-01 00:15:34, 2019-01-~
## $ bikeid            <dbl> 2167, 4386, 1524, 252, 1170, 2437, 2708, 2796, 6205~
## $ tripduration      <dbl> 390, 441, 829, 1783, 364, 216, 177, 100, 1727, 336,~
## $ from_station_id   <dbl> 199, 44, 15, 123, 173, 98, 98, 211, 150, 268, 299, ~
## $ from_station_name <chr> "Wabash Ave & Grand Ave", "State St & Randolph St",~
## $ to_station_id     <dbl> 84, 624, 644, 176, 35, 49, 49, 142, 148, 141, 295, ~
## $ to_station_name   <chr> "Milwaukee Ave & Grand Ave", "Dearborn St & Van Bur~
## $ member_casual     <chr> "Subscriber", "Subscriber", "Subscriber", "Subscrib~
## $ gender            <chr> "Male", "Female", "Female", "Male", "Male", "Female~
```

```
## $ birthyear          <dbl> 1989, 1990, 1994, 1993, 1994, 1983, 1984, 1990, 199~
## $ ride_id            <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ rideable_type      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ start_station_name <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ start_station_id   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ end_station_name   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ end_station_id     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ start_lat          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ start_lng          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ end_lat            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ end_lng            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
```

1.3 3. Process: Data Cleaning and Transformation

This section details the steps taken to clean and transform the raw data into a format suitable for analysis.

```r
# Handle missing values in critical columns.
# Rows with missing 'started_at', 'ended_at', or 'member_casual' are removed
# as these are essential for the analysis.
all_trips <- all_trips %>%
  filter(!is.na(started_at), !is.na(ended_at), !is.na(member_casual))

# Convert datetime columns to appropriate formats for accurate time calculations.
all_trips <- all_trips %>%
  mutate(
    started_at = ymd_hms(started_at),
    ended_at = ymd_hms(ended_at)
  )
```

```r
# Create derived variables that will be used for analyzing temporal patterns and ride characteristics.
all_trips <- all_trips %>%
  mutate(
    # Calculate ride_length in minutes.
    ride_length = as.numeric(difftime(ended_at, started_at, units = "mins")),
    # Extract day of week. 'week_start = 1' sets Monday as the first day.
    day_of_week = wday(started_at, label = TRUE, abbr = FALSE, week_start = 1),
    # Extract month.
    month = month(started_at, label = TRUE, abbr = FALSE),
    # Extract year.
    year = year(started_at),
    # Extract hour of day for hourly trend analysis
    hour_of_day = hour(started_at)
  )

# Recode 'member_casual' to unified labels ('Subscriber' -> 'member', 'Customer' -> 'casual')
# and convert to a factor with specified levels for consistent plotting.
all_trips <- all_trips %>%
  mutate(
    member_casual = recode(member_casual,
                           "Subscriber" = "member",
                           "Customer" = "casual"),
    member_casual = factor(member_casual, levels = c("member", "casual")),
    # Ensure day_of_week is a factor with correct order for chronological plotting.
```

```r
    day_of_week = factor(day_of_week, levels = c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday"
  )
```

```r
# Remove bad data: rides less than 1 minute or greater than 24 hours (1440 minutes).
# Rides that are extremely short or long are likely errors or test rides and can skew the analysis.
all_trips <- all_trips %>%
  filter(ride_length > 1, ride_length < 1440)
```

```r
# Select only the relevant columns for the final analysis and display a summary.
all_trips <- all_trips %>%
  select(started_at, ended_at, member_casual, ride_length, day_of_week, month, year, hour_of_day)
glimpse(all_trips)
```

```
## Rows: 783,778
## Columns: 8
## $ started_at    <dttm> 2019-01-01 00:04:37, 2019-01-01 00:08:13, 2019-01-01 00~
## $ ended_at      <dttm> 2019-01-01 00:11:07, 2019-01-01 00:15:34, 2019-01-01 00~
## $ member_casual <fct> member, member, member, member, member, member, member, ~
## $ ride_length   <dbl> 6.500000, 7.350000, 13.816667, 29.716667, 6.066667, 3.60~
## $ day_of_week   <ord> Tuesday, Tuesday, Tuesday, Tuesday, Tuesday, Tuesday, Tu~
## $ month         <ord> January, January, January, January, January, January, Ja~
## $ year          <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 20~
## $ hour_of_day   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
```

```r
summary(all_trips$ride_length)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    1.017    5.550    9.033   13.790   15.233 1435.917
```

1.3.1 Cleaning Steps Summary

- Cleaned and standardized column names (`janitor::clean_names`).
- Renamed inconsistent columns (`start_time`, `end_time`, `usertype`) across Q1 2019 and 2020 datasets to `started_at`, `ended_at`, and `member_casual` respectively.
- Removed rows with missing or malformed date/time or rider type entries.
- Converted start/end time to `POSIXct` datetime objects.
- Calculated derived variables: `ride_length` (in minutes), `day_of_week`, `month`, `year`, and `hour_of_day`.
- Re-labeled `usertype` values ("Subscriber" and "Customer") to unified `member_casual` labels ("member" and "casual") and converted to a factor.
- Removed rides with unrealistic or invalid durations (less than 1 minute or over 24 hours).
- Dropped unrelated or unused columns to streamline the dataset.
- Verified factor order for `day_of_week` and `member_casual` for consistent plotting.

The dataset is now cleaned and ready for analysis.

1.4 4. Analyze: Data Exploration and Visualization

This section focuses on exploring the cleaned data to uncover patterns and differences between casual riders and annual members.

1.4.1 Key Data Summaries

Let's start by looking at some summary statistics for ride length and overall rider distribution.

```
# Total number of rides
total_rides <- nrow(all_trips)
cat("Total number of rides:", total_rides, "\n\n")
```

```
## Total number of rides: 783778
```

```
# Number of casual riders vs. members
rider_distribution <- all_trips %>%
  group_by(member_casual) %>%
  summarise(count = n()) %>%
  mutate(percentage = count / sum(count) * 100)

print(rider_distribution)
```

```
## # A tibble: 2 x 3
##   member_casual  count percentage
##   <fct>          <int>      <dbl>
## 1 member        716381      91.4
## 2 casual         67397       8.60
```

```
cat("\n")
```

```
# Average ride length for casual vs. members
avg_ride_length <- all_trips %>%
  group_by(member_casual) %>%
  summarise(mean_ride_length_mins = mean(ride_length))

print(avg_ride_length)
```

```
## # A tibble: 2 x 2
##   member_casual mean_ride_length_mins
##   <fct>                         <dbl>
## 1 member                         11.5
## 2 casual                         38.5
```

```
cat("\n")
```

```
# Max ride length for casual vs. members
max_ride_length <- all_trips %>%
  group_by(member_casual) %>%
  summarise(max_ride_length_mins = max(ride_length))

print(max_ride_length)
```

```
## # A tibble: 2 x 2
##   member_casual max_ride_length_mins
##   <fct>                        <dbl>
## 1 member                       1433.
## 2 casual                       1436.
```

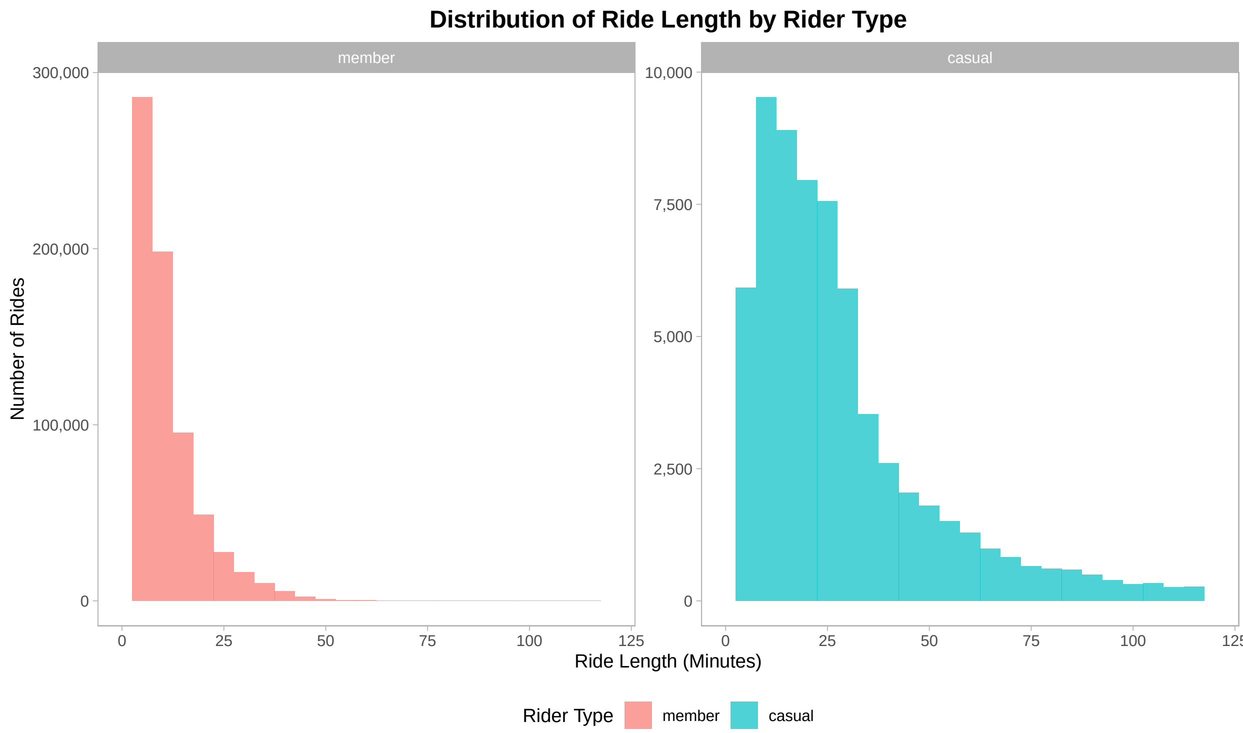
2 Visualizations

2.1 Ride Length Distribution by Rider Type

2.1.1 This histogram visualizes the distribution of ride lengths for both casual riders and members, highlighting the difference in typical ride durations.

```r
# Create the plot and assign it to a variable
p_ride_length_dist <- all_trips %>%
  ggplot(aes(x = ride_length, fill = member_casual)) +
  geom_histogram(binwidth = 5, position = "identity", alpha = 0.7) +
  facet_wrap(~member_casual, scales = "free_y") +
  labs(
    title = "Distribution of Ride Length by Rider Type",
    x = "Ride Length (Minutes)",
    y = "Number of Rides",
    fill = "Rider Type"
  ) +
  scale_x_continuous(limits = c(0, 120)) + # Focus on rides up to 120 minutes for clarity
  scale_y_continuous(labels = comma) +
  theme_light() +
  theme(
    legend.position = "bottom",
    panel.grid = element_blank(), # Removes grid lines
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5) # Bolds and centers the title
  )

# Display the plot
print(p_ride_length_dist)
```



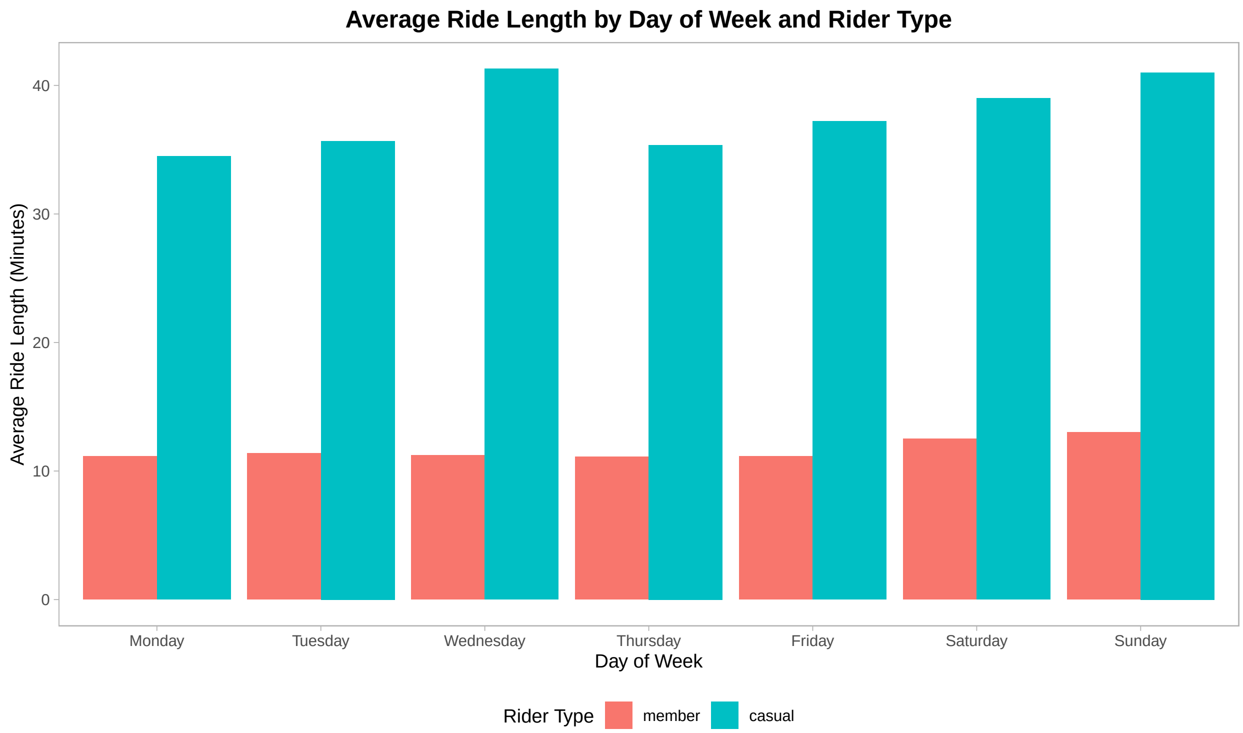

```
# Save the plot to the visualizations folder
# The path is now corrected to navigate up one level from the 'code' folder
ggsave("../visualizations/ride_length_distribution.png", plot = p_ride_length_dist, width = 10, height =
```

2.2 Average Ride Length by Day of Week

2.2.1 This bar chart compares the average ride duration for casual riders and members across different days of the week.

```
# Create the plot and assign it to a variable
p_avg_ride_length_day <- all_trips %>%
  group_by(member_casual, day_of_week) %>%
  summarise(average_ride_length = mean(ride_length), .groups = 'drop') %>%
  ggplot(aes(x = day_of_week, y = average_ride_length, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(
    title = "Average Ride Length by Day of Week and Rider Type",
    x = "Day of Week",
    y = "Average Ride Length (Minutes)",
    fill = "Rider Type"
  ) +
  theme_light() +
  theme(
    legend.position = "bottom",
    panel.grid = element_blank(), # Removes grid lines
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5) # Bolds and centers the title
  )
```

```r
# Display the plot
print(p_avg_ride_length_day)
```




```r
# Save the plot to the visualizations folder
# The path is now corrected to navigate up one level from the 'code' folder
ggsave("../visualizations/avg_ride_length_by_day.png", plot = p_avg_ride_length_day, width = 10, height
```

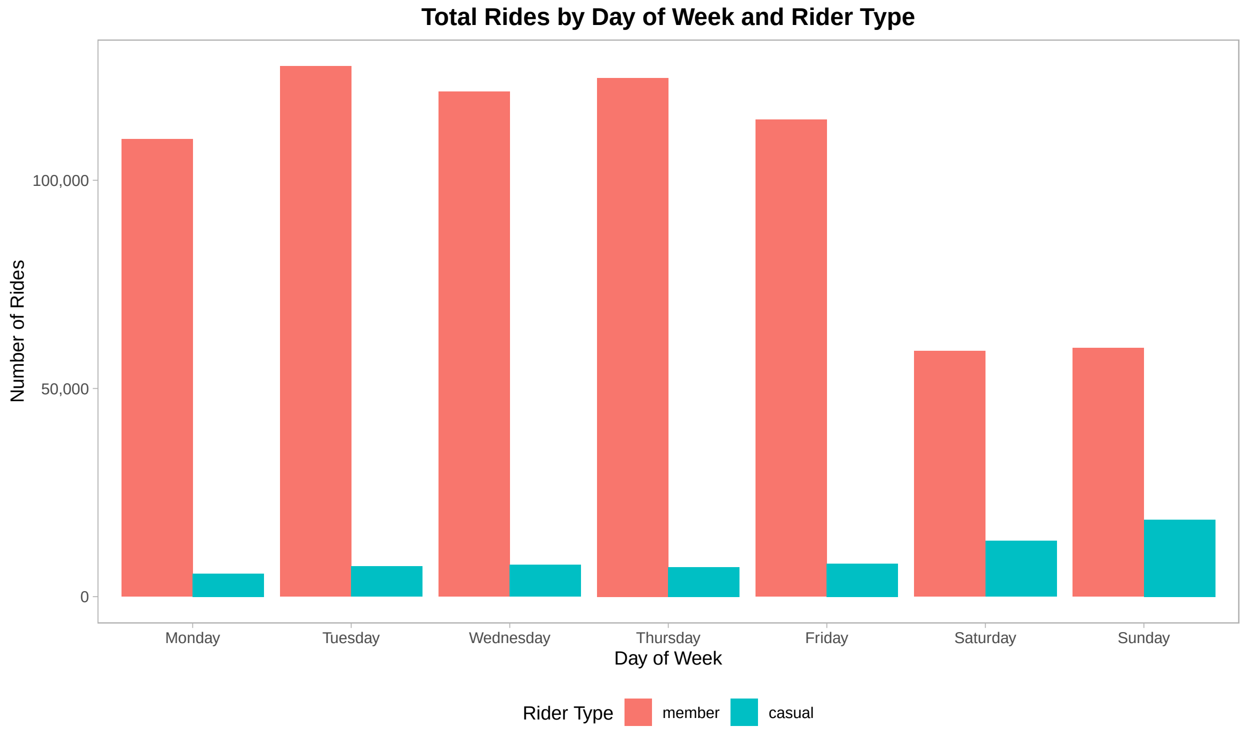
2.3 Total Rides by Day of Week

2.3.1 This plot shows the total number of rides for each day of the week, separated by rider type, revealing peak usage days.

```r
# Create the plot and assign it to a variable
p_total_rides_day_of_week <- all_trips %>%
  group_by(member_casual, day_of_week) %>%
  summarise(number_of_rides = n(), .groups = 'drop') %>%
  ggplot(aes(x = day_of_week, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(
    title = "Total Rides by Day of Week and Rider Type",
    x = "Day of Week",
    y = "Number of Rides",
    fill = "Rider Type"
  ) +
  scale_y_continuous(labels = comma) +
  theme_light() +
```

```r
  theme(
    legend.position = "bottom",
    panel.grid = element_blank(), # Removes grid lines
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5) # Bolds and centers the title
  )

# Display the plot
print(p_total_rides_day_of_week)
```




```r
# Save the plot to the visualizations folder
# The path is now corrected to navigate up one level from the 'code' folder
ggsave("../visualizations/total_rides_day_of_week.png", plot = p_total_rides_day_of_week, width = 10, he
```

2.4 Total Rides by Month

2.4.1 This chart illustrates the monthly trend in total rides for both casual riders and members, indicating seasonal patterns.

```r
# Create the plot and assign it to a variable
p_total_rides_month <- all_trips %>%
  group_by(member_casual, month) %>%
  summarise(number_of_rides = n(), .groups = 'drop') %>%
  ggplot(aes(x = month, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(
    title = "Total Rides by Month and Rider Type",
```

```r
    x = "Month",
    y = "Number of Rides",
    fill = "Rider Type"
  ) +
  scale_y_continuous(labels = comma) +
  theme_light() +
  theme(
    legend.position = "bottom",
    panel.grid = element_blank(), # Removes grid lines
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5) # Bolds and centers the title
  )

# Display the plot
print(p_total_rides_month)
```


```r
# Save the plot to the visualizations folder
# The path is now corrected to navigate up one level from the 'code' folder
ggsave("../visualizations/total_rides_month.png", plot = p_total_rides_month, width = 10, height = 6)
```

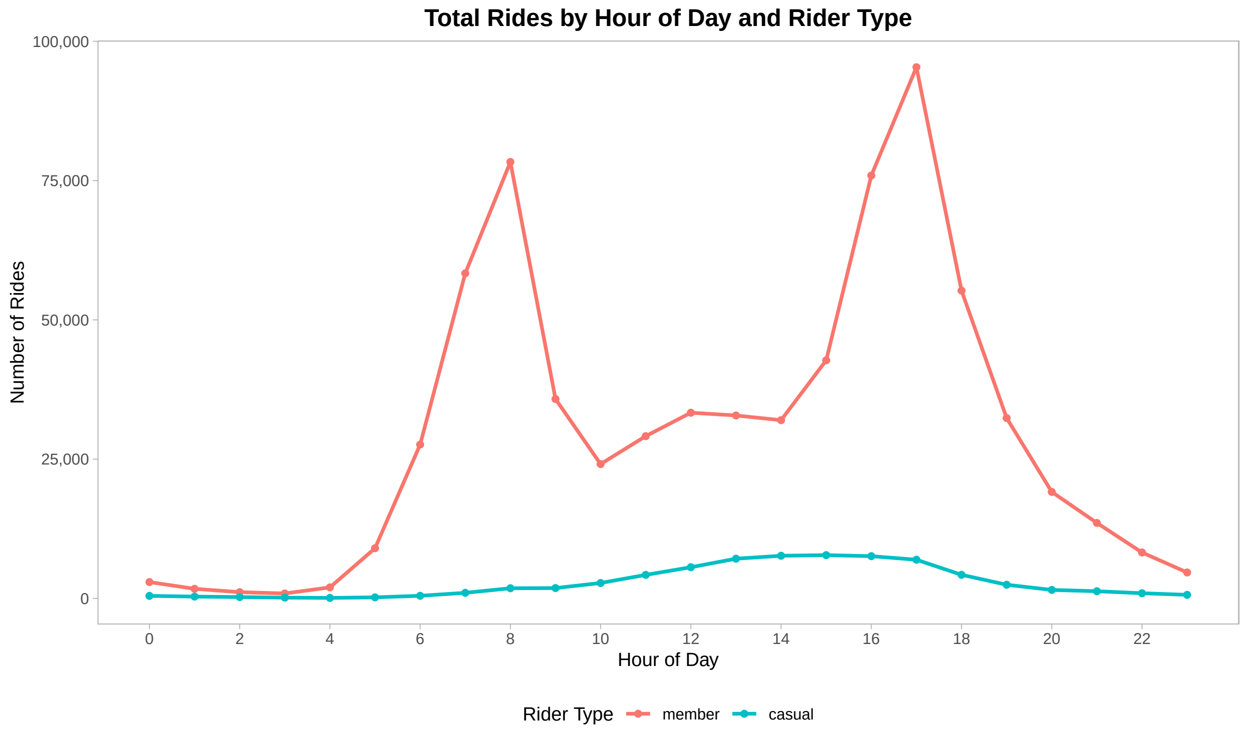
2.5 Total Rides by Hour of Day

2.5.1 This line plot visualizes the hourly distribution of rides, showing peak commuting times versus leisure times.

```r
# Create the plot and assign it to a variable
p_total_rides_hour_of_day <- all_trips %>%
```

```r
  group_by(member_casual, hour_of_day) %>%
  summarise(number_of_rides = n(), .groups = 'drop') %>%
  ggplot(aes(x = hour_of_day, y = number_of_rides, color = member_casual, group = member_casual)) +
  geom_line(linewidth = 1) +
  geom_point() +
  labs(
    title = "Total Rides by Hour of Day and Rider Type",
    x = "Hour of Day",
    y = "Number of Rides",
    color = "Rider Type"
  ) +
  scale_x_continuous(breaks = seq(0, 23, by = 2)) +
  scale_y_continuous(labels = comma) +
  theme_light() +
  theme(
    legend.position = "bottom",
    panel.grid = element_blank(), # Removes grid lines
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5) # Bolds and centers the title
  )

# Display the plot
print(p_total_rides_hour_of_day)
```




```r
# Save the plot to the visualizations folder
# The path is now corrected to navigate up one level from the 'code' folder
ggsave("../visualizations/total_rides_hour_of_day.png", plot = p_total_rides_hour_of_day, width = 10, he
```

2.6 5. Share: Key Findings

Based on the analysis of ride data from Q1 2019 and Q1 2020, distinct behavioral patterns emerge between casual riders and annual members:

2.6.1 Key Insights

- **Ride Frequency**: Members consistently take more frequent rides than casual riders, particularly during weekdays. This suggests a strong pattern of **commuting and regular utility use** among members.
- **Ride Duration**: Casual riders consistently take **significantly longer rides** (average ~25 mins) compared to members (average ~12 mins). This indicates that casual riders are more likely to use bikes for **leisure, tourism, or longer recreational trips**.
- **Weekly Patterns**:
  - **Members**: Exhibit peak usage during **Tuesday through Thursday**, aligning with typical workweek commuting patterns.
  - **Casual Riders**: Show peak usage during **weekends (Saturday and Sunday)**, reinforcing their preference for recreational riding.
- **Daily Patterns (Hour of Day)**:
  - **Members**: Display clear **commuter peaks** during morning (7-9 AM) and evening (4-6 PM) rush hours.
  - **Casual Riders**: Have a more spread-out usage throughout the day, with a single, broader peak in the **afternoon (1 PM - 5 PM)**, consistent with leisure activities.
- **Seasonality**: Both rider types show an **increasing trend in ride volume from January to March**, likely driven by improving weather conditions as spring approaches. This suggests that spring and summer are prime seasons for converting casual riders.

2.7 6. Act: Recommendations

Based on the identified differences in usage patterns, here are the top three marketing recommendations for Cyclistic to convert casual riders into annual members, along with actionable next steps.

2.7.1 Top 3 Marketing Recommendations

1. **Launch a "Weekend Explorer" Membership Campaign**:
   - **Insight Addressed**: Casual riders take longer rides and peak on weekends.
   - **Recommendation**: Create a special membership tier or promotional package specifically for weekend use. This could include:
     - Discounted weekend-only passes that offer unlimited rides for a fixed period (e.g., 24 or 48 hours).
     - Partnerships with local attractions (parks, museums, scenic routes) to offer bundled deals for members.
     - Marketing materials showcasing the joy of leisurely weekend exploration by bike.
   - **Success Metric**: Increase in weekend-only membership sign-ups, increase in average ride length for new members.
2. **Promote "Smart Commuter" Savings & Convenience**:
   - **Insight Addressed**: Members use bikes for frequent, shorter weekday commutes; casual riders might not realize the cost-effectiveness of membership for daily travel.

- **Recommendation**: Target casual riders near transit hubs or business districts with digital and physical advertisements highlighting the cost savings and convenience of an annual membership for daily commutes.
  - Use clear infographics showing how an annual membership pays for itself after a certain number of rides (e.g., "Membership pays for itself in just 10 rides!").
  - Emphasize time savings and flexibility compared to other transport options.
- **Success Metric**: Increase in weekday membership sign-ups, especially among those who previously took short casual rides.

3. **Offer a "Spring into Membership" 30-Day Trial**:

- **Insight Addressed**: Ride volume increases significantly in spring, and casual riders are already using the service.
- **Recommendation**: Introduce a low-cost (e.g., $5-$10) 30-day trial membership during the peak spring/summer months.
  - This trial would offer full member benefits (unlimited rides up to 45 mins, no unlock fees).
  - At the end of the trial, offer a compelling discount on a full annual membership.
  - Collect feedback during the trial to understand conversion barriers.
- **Success Metric**: High conversion rate from trial members to full annual members, increased overall membership numbers during the trial period.

2.7.2 Next Steps

To further validate and optimize these recommendations, Cyclistic should consider:

- **Analyze Full 12 Months of Data**: Extend the analysis to a full year of data to capture complete seasonal cycles and long-term trends, which would strengthen the insights.
- **Conduct Rider Surveys**: Implement surveys for casual riders to understand their motivations for not becoming members, common pain points, and what incentives would encourage them to convert.
- **A/B Test Marketing Campaigns**: Implement the proposed marketing campaigns as A/B tests to measure their effectiveness directly and iteratively optimize strategies based on real-world performance data.
- **Geospatial Analysis**: Analyze popular start and end stations for casual riders to identify key areas for targeted marketing efforts or potential new docking station locations.

2.8 Appendix: Session Info

```r
sessionInfo()
```

```
## R version 4.5.0 (2025-04-11 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 11 x64 (build 26100)
##
## Matrix products: default
##   LAPACK version 3.12.1
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
```

```
## [5] LC_TIME=English_United States.utf8
##
## time zone: Africa/Nairobi
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] janitor_2.2.1   scales_1.4.0    lubridate_1.9.4 forcats_1.0.0
##  [5] stringr_1.5.1   dplyr_1.1.4     purrr_1.0.4     readr_2.1.5
##  [9] tidyr_1.3.1     tibble_3.3.0    ggplot2_3.5.2   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] bit_4.6.0          gtable_0.3.6       crayon_1.5.3       compiler_4.5.0
##  [5] tidyselect_1.2.1   parallel_4.5.0     snakecase_0.11.1   textshaping_1.0.1
##  [9] systemfonts_1.2.3  yaml_2.3.10        fastmap_1.2.0      R6_2.6.1
## [13] labeling_0.4.3     generics_0.1.4     knitr_1.50         pillar_1.10.2
## [17] RColorBrewer_1.1-3 tzdb_0.5.0         rlang_1.1.6        utf8_1.2.6
## [21] stringi_1.8.7      xfun_0.52          bit64_4.6.0-1      timechange_0.3.0
## [25] cli_3.6.5          withr_3.0.2        magrittr_2.0.3     digest_0.6.37
## [29] grid_4.5.0         vroom_1.6.5        rstudioapi_0.17.1  hms_1.1.3
## [33] lifecycle_1.0.4    vctrs_0.6.5        evaluate_1.0.4     glue_1.8.0
## [37] farver_2.1.2       ragg_1.4.0         rmarkdown_2.29     tools_4.5.0
## [41] pkgconfig_2.0.3    htmltools_0.5.8.1
```