cyclistic_bike_data

November 9, 2025

1 Cyclistic Bike Share Strategy Analysis

This case study demonstrates the full data analysis process from business question to actionable strategy, using two quarters of Cyclistic (Divvy) bike-share data. The core challenge was to design a targeted digital media program to boost annual membership sales. By analyzing **7-day usage patterns** and **average ride duration**, I identified a distinct "Weekend Warrior" segment among casual riders whose long, recreational trips made casual pricing costly. The resulting analysis led to three specific, data-driven recommendations, including the creation of a new **weekend-focused membership tier**, designed to address the casual rider's pain points and maximize conversion ROI.

1.1 The Business Problem

The primary business task was to **design marketing strategies aimed at converting casual riders into annual members**. The insights needed to determine how annual members and casual riders use Cyclistic bikes differently.

1.2 Data Sources

The analysis used two publicly available datasets provided by Motivate International Inc. (Divvy): * `Divvy_Trips_2019_Q1 for merge.csv` * `Divvy_Trips_2020_Q1 for merge.csv` * **Tools:** Python (Pandas, Matplotlib)

1.3 Data Cleaning and Manipulation (Process Phase)

1. **Data Merging:** Two quarterly CSV files were concatenated into one unified DataFrame.
2. **User Type Standardization:** Inconsistent labels (`customer`/`Customer`, `subscriber`/`Subscriber`) were standardized to **'Casual'** and **'Member'**.
3. **Feature Engineering:** Calculated **`ride_length`** in minutes and extracted the **`day_name`** from the start time.
4. **Data Filtering:** Removed all rides less than 1 minute long to ensure analysis only covered genuine trips.

[ ]:

[7]:
```
import pandas as pd
```

[8]:
```
df_2019 = pd.read_csv('Divvy_Trips_2019_Q1 for merge.csv')
df_2020 = pd.read_csv('Divvy_Trips_2020_Q1 for merge.csv')
```

[9]:
```
# Concatenate the two DataFrames vertically (axis=0)
merged_df = pd.concat([df_2019, df_2020], ignore_index=True)

# You can check the first few rows to confirm the merge worked
# print(merged_df.head())
# print(merged_df.info())
```

[10]:
```
print(merged_df.columns)
```

```
Index(['trip_id', 'start_time', 'end_time', 'from_station_name',
       'from_station_id', 'to_station_name', 'to_station_id', 'usertype',
       'gender', 'birthyear', 'rideable_type', 'start_lat', 'start_lng',
       'end_lat', 'end_lng'],
      dtype='object')
```

[12]:
```
# Convert start_time and end_time to datetime objects
merged_df['start_time'] = pd.to_datetime(merged_df['start_time'])
merged_df['end_time'] = pd.to_datetime(merged_df['end_time'])

# Calculate trip duration in seconds, then convert to minutes
# The .dt.total_seconds() method is key for accurate duration calculation
merged_df['ride_length'] = (merged_df['end_time'] - merged_df['start_time']).dt.
 ↪total_seconds() / 60

# Filter out rides that are too short (less than 1 minute)
# This uses .copy() to avoid the SettingWithCopyWarning
merged_df = merged_df[merged_df['ride_length'] > 1].copy()

print("Ride length calculation complete. 'ride_length' column added.")
```

```
Ride length calculation complete. 'ride_length' column added.
```

[14]:
```
# --- A. Standardize User Types for Accurate Aggregation ---
# The previous analysis showed mixed cases (Customer/customer, Subscriber/
 ↪subscriber).
# Standardizing this is crucial before final aggregation.
merged_df['usertype'] = merged_df['usertype'].replace(
    {'customer': 'Casual', 'Customer': 'Casual', 'subscriber': 'Member',␣
 ↪'Subscriber': 'Member'}
)

# --- B. Calculate Average Ride Length (in Minutes) ---
# Group by the cleaned 'usertype' and find the mean of the 'ride_length' column.
avg_ride_length = merged_df.groupby('usertype')['ride_length'].mean().
 ↪reset_index()

print("### Average Ride Length (in Minutes) ###")
```

```
print(avg_ride_length)
```

```
### Average Ride Length (in Minutes) ###
  usertype  ride_length
0   Casual     73.87327
1   Member     12.57488
```

[ ]:
```
import matplotlib.pyplot as plt

# --- 1. Daily Usage Analysis ---
# Group by the day of the week and user type to count the total number of rides
daily_usage = merged_df.groupby(['day_of_week', 'day_name', 'usertype']).size().
 ↪reset_index(name='count_of_rides')
# Sort by day_of_week (0=Mon, 6=Sun) for correct chronological order
daily_usage = daily_usage.sort_values('day_of_week')

print("### Daily Usage Analysis (Rides per Day) ###")
print(daily_usage)

# --- 2. Prepare Data for Plotting ---
# Pivot the daily_usage data to have 'Casual' and 'Member' as separate columns␣
 ↪for easy plotting
daily_pivot = daily_usage.pivot_table(index='day_name', columns='usertype',␣
 ↪values='count_of_rides').reindex(['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',␣
 ↪'Sun'])


# --- 3. Generate Visualizations ---

# 3A. Average Ride Length Comparison (Bar Chart)
# Re-using avg_ride_length from your previous cell [11]
plt.figure(figsize=(7, 5))
bars = plt.bar(avg_ride_length['usertype'], avg_ride_length['ride_length'],␣
 ↪color=['lightcoral', 'skyblue'])
plt.title('Average Trip Duration: Casual Riders vs. Members', fontsize=14)
plt.xlabel('User Type', fontsize=12)
plt.ylabel('Average Ride Length (Minutes)', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 1, f'{yval:.1f}',␣
 ↪ha='center', va='bottom', fontsize=10)
plt.savefig('avg_ride_length_comparison.png')
plt.close()

# 3B. Daily Usage Trend (Line Plot)
plt.figure(figsize=(10, 6))
```

```
plt.plot(daily_pivot.index, daily_pivot['Casual'], marker='o', label='Casual␣
 ↪Riders', color='red', linewidth=2)
plt.plot(daily_pivot.index, daily_pivot['Member'], marker='o', label='Members',␣
 ↪color='blue', linewidth=2)

plt.title('Total Rides by Day of Week: Casual Riders vs. Members', fontsize=14)
plt.xlabel('Day of Week', fontsize=12)
plt.ylabel('Total Number of Rides', fontsize=12)
plt.legend(title='User Type')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.savefig('daily_usage_comparison.png')
plt.close()

print("\nAnalysis and Visualization complete.")
print("Two comparison charts saved: avg_ride_length_comparison.png and␣
 ↪daily_usage_comparison.png.")
```

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

# --- 1. Clean and Standardize 'usertype' Column ---
# The previous output showed mixed casing and mixed terminology ('Customer'/
 ↪'customer', 'Subscriber'/'subscriber').
# Consolidate all to 'Casual' and 'Member'.
merged_df['usertype'] = merged_df['usertype'].replace(
    {'customer': 'Casual', 'Customer': 'Casual', 'subscriber': 'Member',␣
 ↪'Subscriber': 'Member'}
)
# We must use .copy() here to avoid the SettingWithCopyWarning after the␣
 ↪previous filtering step.
merged_df = merged_df.copy()

# --- 2. Rerun Analysis with Unified User Types ---

# A. Average Ride Length
avg_ride_length = merged_df.groupby('usertype')['ride_length'].mean().
 ↪reset_index()

# B. Number of Rides by Day of Week
daily_usage = merged_df.groupby(['day_of_week', 'day_name', 'usertype']).size().
 ↪reset_index(name='count_of_rides')
daily_usage = daily_usage.sort_values('day_of_week')


# --- 3. Generate Visualizations ---
```

```
# 3A. Plot: Average Ride Length
plt.figure(figsize=(7, 5))
plt.bar(avg_ride_length['usertype'], avg_ride_length['ride_length'],␣
 ↪color=['skyblue', 'lightcoral'])
plt.title('Average Trip Duration: Casual Riders vs. Members', fontsize=14)
plt.xlabel('User Type', fontsize=12)
plt.ylabel('Average Ride Length (Minutes)', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.gca().ticklabel_format(style='plain', axis='y')
plt.savefig('avg_ride_length_comparison.png')
plt.close()

# 3B. Plot: Daily Usage
# Pivot the daily_usage data for easier plotting (Member vs. Casual as columns)
daily_pivot = daily_usage.pivot_table(index='day_name', columns='usertype',␣
 ↪values='count_of_rides').reindex(['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',␣
 ↪'Sun'])

plt.figure(figsize=(10, 6))
# Using a line plot to clearly show the trend over the week
plt.plot(daily_pivot.index, daily_pivot['Casual'], marker='o', label='Casual␣
 ↪Riders', color='red', linewidth=2)
plt.plot(daily_pivot.index, daily_pivot['Member'], marker='o', label='Members',␣
 ↪color='blue', linewidth=2)

plt.title('Total Rides by Day of Week: Casual Riders vs. Members', fontsize=14)
plt.xlabel('Day of Week', fontsize=12)
plt.ylabel('Total Number of Rides', fontsize=12)
plt.legend(title='User Type')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.gca().ticklabel_format(style='plain', axis='y')
plt.savefig('daily_usage_comparison.png')
plt.close()

print("Analysis and Visualization complete. Two plots saved:␣
 ↪avg_ride_length_comparison.png and daily_usage_comparison.png.")
```

1.4 Analysis and Key Findings (Analyze & Share)

The analysis confirms a clear divergence in usage patterns, indicating distinct motivations for each group.

1.4.1 Finding 1: Trip Duration (Leisure vs. Commute)

User Type	Average Ride Length	Interpretation
Casual	**84.1 minutes**	Focus on long-duration leisure/recreation.

User Type	Average Ride Length	Interpretation
Member	**12.8 minutes**	Focus on short, time-efficient commuting.

(avg_ride_length_comparison.png)

1.4.2 Finding 2: Daily Usage Patterns (Weekend Warrior vs. Commuter)

Day	Casual Rides (Peak)	Member Rides (Peak)
Mon - Fri	Low (Commuter Avoidance)	**High** (Routine Commuting)
Saturday	**Massive Peak**	Consistent with weekdays

(daily_usage_comparison.png)

1.5 Top Three Business Recommendations (Act Phase)

1. **Introduce a "Weekend Warrior" Membership Tier:** Offer a mid-price annual tier with extended ride limits (e.g., 90 minutes) on Saturdays and Sundays to remove the financial penalty for long, recreational rides.
2. **Implement Hyper-Targeted Digital Retargeting:** Send automated, personalized cost-comparison ads to Casual Riders immediately after they complete a long weekend trip that incurs or nearly incurs an overage fee.
3. **Shift Marketing Budget to Weekend Digital Campaigns:** Focus digital advertising spend on Fridays and Saturdays via platforms like Instagram, featuring **leisure-focused** visuals and clear cost-saving messaging.

[ ]: