CAPSTONE DESIGN CASE STUDY: CYCLISTIC BIKE SHARE

Investigating bike share usage data for trends and business strategy and marketing opportunities.

DATA ANALYSIS GOOGLE CERTIFICATE

PRESENTER: VO THI THIEN MY

TABLE OF CONTENT:

CASE SUMMARY:

Scenario

I am a junior 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 my 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 ordering 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 your 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. You joined this team six months ago and have been busy learning about Cyclistic's mission and business goals — as well as how you, as a junior 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 ordering. Since then, the program has grown to a fleet of 5,824 bicycles that are geo-tracked 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.

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

DETAIL ANALYSIS PROCESS:

PHASE 1: ASK

1.1. IDENTIFY THE BUSINESS TASK & KEY STAKEHOLDERS

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?

Moreno has assigned the first question to answer: **How do annual members and casual riders use Cyclistic bikes differently?**

Case study Roadmap - Ask	
- what is the problem I am trying to solve? - How can my insights guide business decisions?	- How do annual members and casual riders use Cyclistic bikes differently - My insights can help direct the marketing strategies of Cyclistic in a more suitable way
Key tasks: - Identify business task - Consider key stakeholder	- Business task: Increase Cyclistic company's profit by using marketing strategy to convert casual riders to annual members - Key stakeholder: Lily Moreno (*The director of marketing and my manager),* Cyclistic executive team
Deliverables: Provide a clear statement of a business task	Increase Cyclistic company's profit by using marketing strategy to convert casual riders to annual members with the first problem is to understand how annual members and casual riders use Cyclistic bikes differently

PHASE 2: PREPARE

2.1. GENERAL DATA SET INFORMATION

The data has been made available by Motivate International Inc. (Divvy), (which for the purposes of this exercise has been renamed Cyclistic) under this license. It is available here. The data is in .csv files and arranged by month.

The data does not appear to have issues with bias or credibility, however data-privacy issues prohibit use of riders' personally identifiable information, so it will not be possible to connect pass purchases to credit card numbers to determine if casual riders live in the Cyclistic service area or if they have purchased multiple single passes - information that would be valuable within the scope of this assignment.

I will be using the 12-month data, from July 2022 to June 2023. The data is in 12 spreadsheets, by month, and in total is over 1GB in size. Because of this large size I will be using PostgreSQL 16 to do all the work with the data and Tableau for visualizations.

First, I begin to create 13 tables that can store data from 12-month Cyclistic csv files (Figure 2.1) whereas 12 tables for 12 months and a main table named "bike_share_table" to merge all of these figures together (Figure 2.2):

```sql
CREATE TABLE tripdata_2022_7
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

```sql
CREATE TABLE tripdata_2022_8
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```sql
CREATE TABLE tripdata_2022_9
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

Figure 2.1. Create tables to insert data from csv files

```
--insert data into tables for each month, provided in the above code
--create an aggregated table:
CREATE TABLE bike_share_table
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),
start_lat DOUBLE PRECISION, start_lng DOUBLE PRECISION,
end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

```
--INSERT CSV FILES INTO THE NEW TABLE
INSERT INTO bike_share_table (ride_id,rideable_type,started_at ,ended_at,
start_station_name ,start_station_id ,end_station_name ,end_station_id ,
start_lat ,start_lng ,end_lat,end_lng, member_casual)
(
SELECT ride_id,rideable_type,started_at ,ended_at,start_station_name ,
start_station_id ,end_station_name ,end_station_id ,start_lat ,start_lng ,
end_lat,end_lng, member_casual
FROM tripdata_2022_7
UNION ALL
```

```
SELECT ride_id,rideable_type,started_at ,ended_at,start_station_name ,
start_station_id ,end_station_name ,end_station_id ,start_lat ,start_lng ,
end_lat,end_lng, member_casual
FROM tripdata_2022_8
UNION ALL
```

Figure 2.2. Insert csv files into new tables

PHASE 3: PROCESS

3.1. DATA CLEANING PROCESS AND DOCUMENTATION

After merging these datasets, I set out to inspect and clean the dataframe using SQL.

To start, a *day_of_week* columns is added to *bike_share_table* that is derived from *started_at column.*

```
--PART 2: PROCESS:
SELECT COUNT(*) FROM bike_share_table;
Add new column: day_of_week and ride time
ALTER TABLE bike_share_table
ADD COLUMN day_of_week VARCHAR;
UPDATE bike_share_table
SET day_of_week = TO_CHAR(started_at, 'Day');
```

Figure 3.1. Create *day_of_week* columns derived from *started_at* timestamps

After creating "day_of_week", I started to calculate riding time in minutes using the deduction between "ended_at" and "started_at" timestamps as presented in Figure 3.2. Then, the values will then be stored in a separate column named "ride_time_minutes".

```
ALTER TABLE bike_share_table
ADD COLUMN ride_time_minutes INT;
UPDATE bike_share_table
SET ride_time_minutes = EXTRACT(EPOCH FROM (ended_at - started_at))/60;
```

Figure 3.2. Calculate ride time in minutes for each ride and store in a new column *ride_time_minutes*

Next, I will inpsect whether the "ride_id", which is a primary key in "bike_share_table" has any duplicate, as presented in Figure 3.3:

```sql
--Next, we need to count duplicates:
--my table has duplicates -> create new table with no duplicates
SELECT COUNT(DISTINCT (ride_id)) AS distinct_id_count,
COUNT(ride_id) AS total_ride_count
FROM bike_share_table;
```

```sql
CREATE TABLE new_bike_share AS
SELECT DISTINCT *
FROM bike_share_table;
```

Figure 3.3. Evaluate the new_bike_share dataframe to see if there are any duplicates

It can be seen from figure 3.3 that the "bike_share_table" has duplicates in ride_id. Thus, these records will be removed, and to do this, I will create new table that contains only unique values with the name "new_bike_share" and the aggregations will be performed in this new one.

```sql
--Step 3a: we need to check for spelling errors of rideable_type
--and member_casual columns:
SELECT rideable_type, member_casual
FROM new_bike_share
GROUP BY rideable_type, member_casual; -- Result: no spelling error!
```

Figure 3.4. Check for spelling errors of rideable_type

In the Figure 3.5, records of "started_at" and "ended_at" columns are checked to see whether the end time of the service is eariler that the start time in "new_bike_share". As the result, it turns out that there are 137 instances of such. From my perspective, there are two ways to solve this invalid results, either delete this invalid timestamp or swapping the "ended_at" with "started_at" values and recalculate the riding time in "ride_time_minutes". For this case study, since the "ended_at" and "started_at" columns are necessary to the riding time calculations and answering

the business question, I will demonstrate the 2nd approach. However, in reality, any types of data modifications will need to be informed to the team or my supervisor/partner to recieve timely feedback and assistance.

```
--Step 3b: we need to check for instances where end time is before start time:
SELECT *
FROM new_bike_share
WHERE ended_at < started_at; -- Result: there are 137 instances
-- There are two ways to do this, either delete this invalid timestamp or
--swapping ended_at with started_at
-- value and re-calculate the ride_time_minutes.
--Hence, i will demonstrate the 2nd approach.
UPDATE new_bike_share
SET (started_at, ended_at) = (ended_at, started_at)
WHERE ended_at < started_at;
```

```
--I will recalculate ride_time_minutes after correction:
UPDATE new_bike_share
SET ride_time_minutes = EXTRACT(EPOCH FROM (ended_at - started_at))/60
WHERE ended_at > started_at;
```

```
SELECT *
FROM new_bike_share
WHERE ended_at < started_at; -- Double-check: no more ended_at < started_at!
```

Figure 3.5. Check for any instances where end time is before start time

The next step is to look further into null values of station names and coordinations of these stations in Chicago sector, provided in figure 3.6 below.

```sql
--Step 4: I will now look further into null values of station names
--and coordinates of these stations:
--4a: Check null values in start and end station names:
WITH null_station_name AS (SELECT COUNT(*) AS null_val
FROM new_bike_share
WHERE start_station_name IS NULL
  OR end_station_name IS NULL)
SELECT ROUND((SELECT null_val FROM null_station_name)100/COUNT(),2)
  AS null_entries
FROM new_bike_share;
```

Figure 3.6. Check for null values in station names

```sql
--4b: check null values in lat and lng of stations:
SELECT count(*)
FROM new_bike_share
WHERE start_lat IS NULL
  OR start_lng IS NULL
  OR end_lat IS NULL
  OR end_lng IS NULL;
--Result: 7049 entries where either latitude or longitude of
--start and end stations is null
```

Figure 3.7. Check for null valus in latitude and longitude of stations

```sql
--4c: Next, I check for distinct start station and end station names:
SELECT DISTINCT start_station_name, end_station_name
FROM new_bike_share;

SELECT DISTINCT start_station_name, end_station_name
FROM new_bike_share
WHERE NOT (start_station_name LIKE '%&%' OR end_station_name LIKE '%&%');
There are several unsual entries where start_station_name = end_station_name:
SELECT DISTINCT start_station_name, end_station_name
FROM new_bike_share
WHERE start_station_name = end_station_name;
```

Figure 3.8. Check for any distinction in names of start and end station names

The results from figure 3.8 suggests that there are several cases where the stations are named after intersections. For example: “LaSalle St & Adams St”, so I write a query to exclusively filter results that include the ‘&’ symbol to reduce the number of stations I must look through to see if there are any unusual names e.g. ‘service’ or ‘test’ that are not for bike_share service but rather for charging and maintenance => Results: none of the stations are “service” or “test”

And since there are 1384 rows with start and end station names are the same, it is possible that these regions required bike as a mean of transportation to quickly travel from one place to another within the area. Therefore, I will keep these records.

PHASE 4: ANALYSE

4.1. ANALYSE PROCESS AND DOCUMENTATION

4.1.1. Total ride by member type

```
SELECT member_casual, COUNT(ride_id) as total_rides
FROM new_bike_share
GROUP BY member_casual
ORDER BY member_casual;
```

Figure 4.1. pivot table 1: total ride by member type

4.1.2. Average ride by membership type

```
SELECT member_casual, ROUND(AVG(ride_time_minutes),3) AS average_ride_time
FROM new_bike_share
GROUP BY member_casual
ORDER BY member_casual;
```

Figure 4.2. pivot table 2: average ride by casual riders and members

4.1.3. Count of rideable type and average time by membership type and bike type

```
--COUNT OF RIDEABLE TYPE AND AVERAGE TIME BY MEMBER AND BIKE TYPE
SELECT member_casual, rideable_type,
ROUND(AVG(ride_time_minutes),2) AS average_ride_time,
COUNT(rideable_type) AS bike_type_totals
FROM new_bike_share
GROUP BY member_casual, rideable_type
ORDER BY member_casual;
```

Figure 4.3. pivot table 3: average ride time for each rideable type in members and casuals

4.1.4. Ride count and average time by day of week.

```sql
--RIDE COUNT AND AVERAGE TIME BY MEMBER AND DAY OF WEEK
SELECT member_casual, rideable_type,
COUNT(ride_id) AS ride_count,
ROUND(AVG(ride_time_minutes),2) AS average_time_in_days,
day_of_week
FROM new_bike_share
GROUP BY rideable_type, member_casual, day_of_week
ORDER BY member_casual;
```

Figure 4.4. pivot table 4: number of rides and riding time by day of week

4.1.5. Ride count by month

```sql
--COUNT OF RIDEABLE TYPE BY MONTH
SELECT member_casual,rideable_type,
COUNT(rideable_type) AS count_bike_type,
EXTRACT (MONTH FROM started_at) AS month
FROM new_bike_share
GROUP BY member_casual, rideable_type, month
ORDER BY member_casual;
```

Figure 4.5. pivot table 5: number of rides by month

4.1.6. Average ride time and monthly total by membership type

```sql
--AVERAGE RIDE TIME AND MONTHLY TOTAL BY MEMBERSHIP TYPE
SELECT member_casual, EXTRACT (MONTH FROM started_at) AS month,
ROUND(AVG(ride_time_minutes),2) AS average_ride_time,
COUNT(ride_id) AS monthly_rides
FROM new_bike_share
GROUP BY member_casual, month
ORDER BY member_casual, month;
```

Figure 4.6. pivot table 6: average ride time by month and membership type

4.1.7. Count average ride time and the number of rides

```sql
--COUNT AVERAGE RIDE TIME AND NUMBER OF RIDES
--BY DAY OF WEEK AND AND MONTH AND MEMBERSHIP AND RIDE TYPE
SELECT member_casual, rideable_type,
EXTRACT(MONTH FROM started_at) AS month,
ROUND(AVG(ride_time_minutes),2) AS average_ride_time,
COUNT (rideable_type) AS bike_totals, day_of_week
FROM new_bike_share
GROUP BY rideable_type, member_casual, day_of_week, month
ORDER BY month;
```

Figure 4.7. pivot table 7: average ride time by week and month

4.1.8. Top 20 starting and ending stations for each membership type.

4.1.8.1. Casual riders

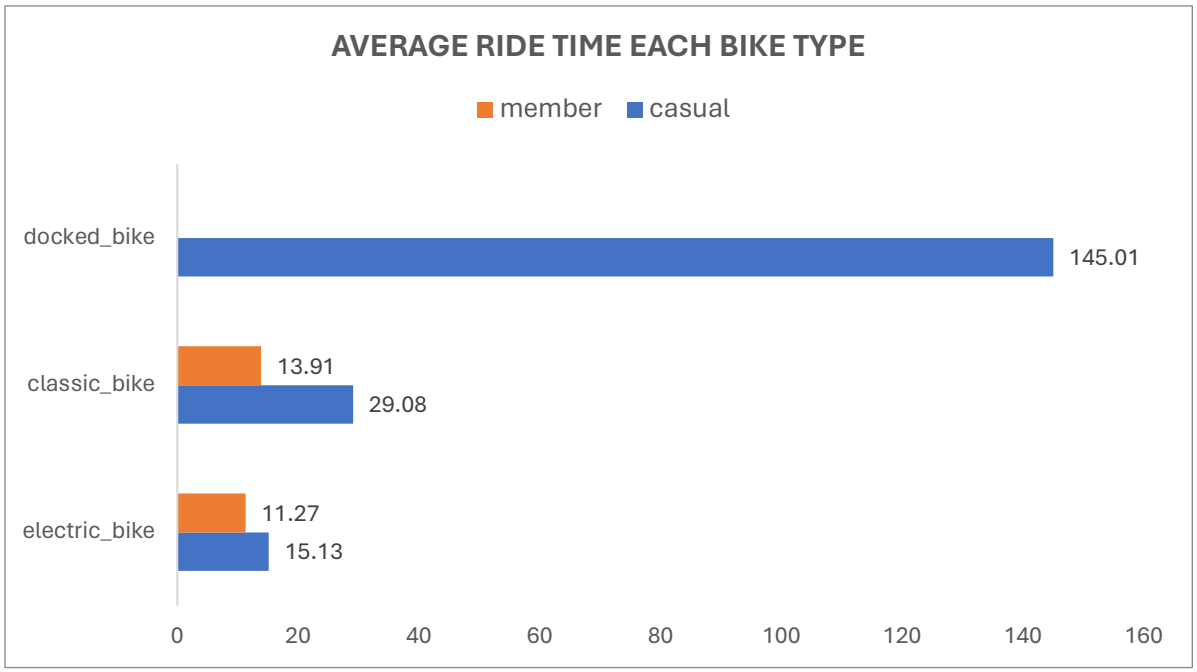
```sql
--COUNT TOP 20 STARTING AND ENDING STATIONS FOR CASUAL RIDERS
SELECT member_casual, start_station_name,
COUNT(start_station_name) AS count_start_station
FROM new_bike_share
WHERE member_casual = 'casual'
GROUP BY member_casual, start_station_name
ORDER BY count_start_station DESC
LIMIT 20;
SELECT member_casual, end_station_name,
COUNT(end_station_name) AS count_end_station
FROM new_bike_share
WHERE member_casual = 'casual'
GROUP BY member_casual, end_station_name
ORDER BY count_end_station DESC
LIMIT 20;
```

4.1.8.2. Cyclistic members

```sql
--COUNT TOP 20 STARTING AND ENDING STATIONS FOR MEMBERS
SELECT member_casual, start_station_name,
COUNT(start_station_name) AS count_start_station
FROM new_bike_share
WHERE member_casual = 'member'
GROUP BY member_casual, start_station_name
ORDER BY count_start_station DESC
LIMIT 20;
SELECT member_casual, end_station_name,
COUNT(end_station_name) AS count_end_station
FROM new_bike_share
WHERE member_casual = 'member'
GROUP BY member_casual, end_station_name
ORDER BY count_end_station DESC
LIMIT 20;
```

PHASE 5: SHARE

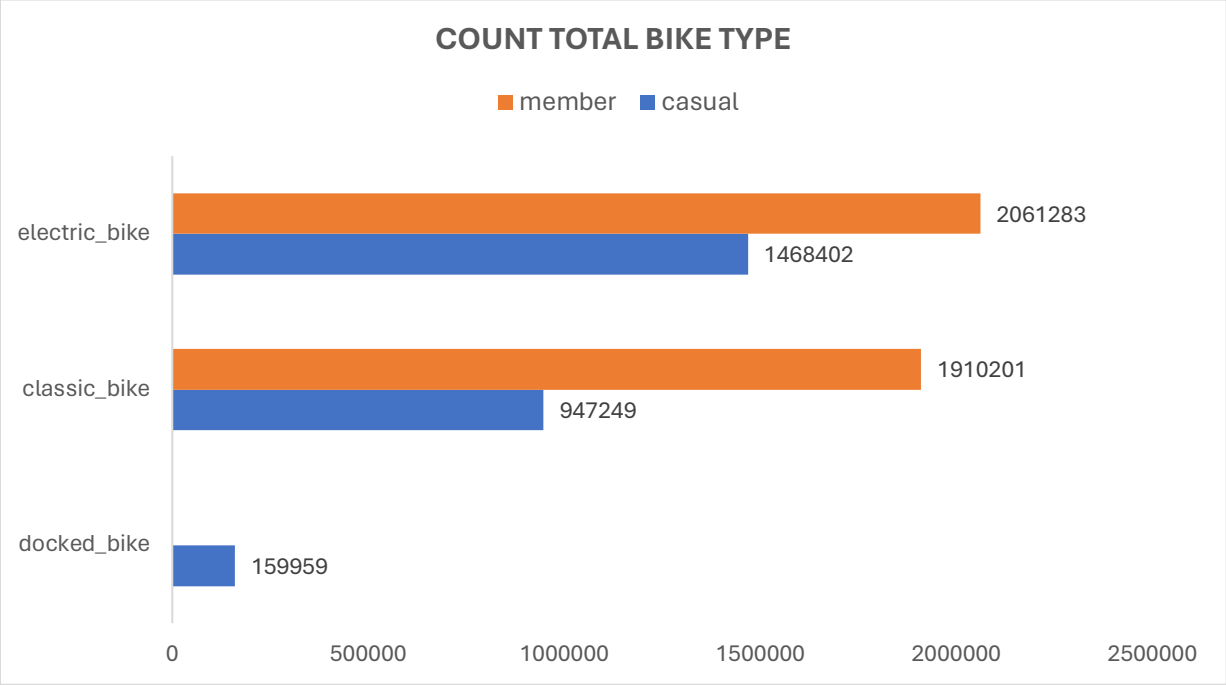
5.1. CREATE VISUALIZATION



Visualization 1: Average ride time of each rideable type

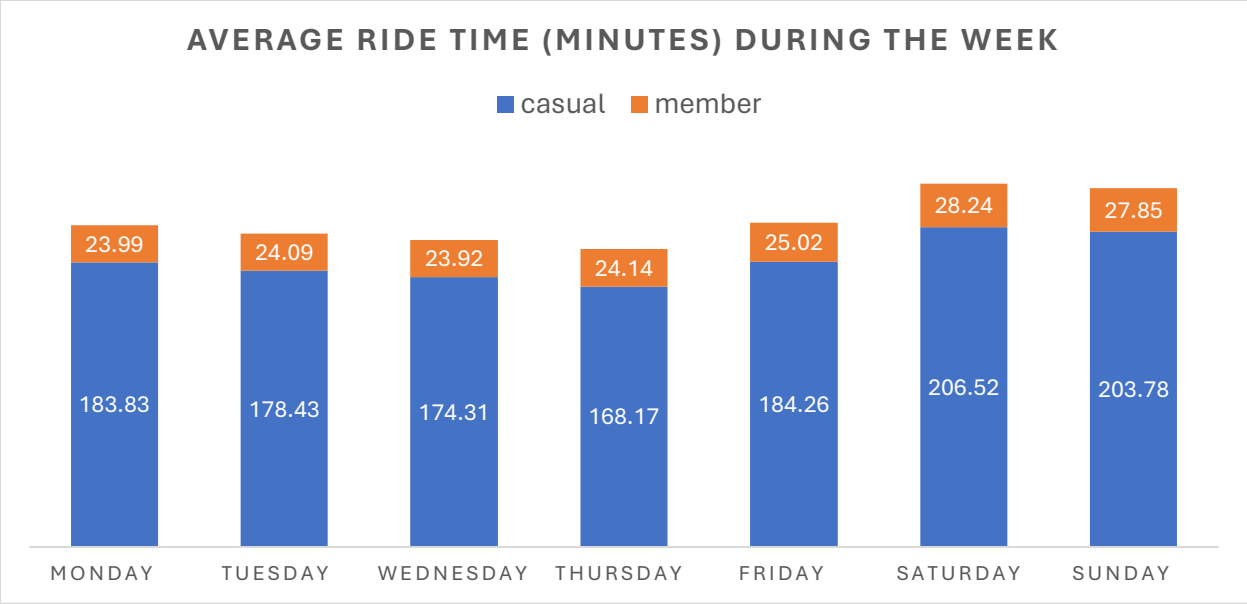
It can be seen that members of Cyclistic service only use classic and electronic bikes to travel, while docked bikes are used among casual riders to travel long distances with average ride time was more than 2 hours for each ride. What stands out in this visualization is that the time spent on each ride for the other two rideable types of casual riders was somehow higher than that of members.

This suggests that casual riders use the service to travel greater distances than members already in the program, hence, promotions for members on long distance travels should be encouraged.



Visualization 2: The number of rides for each rideable type in 2022-2023 recorded

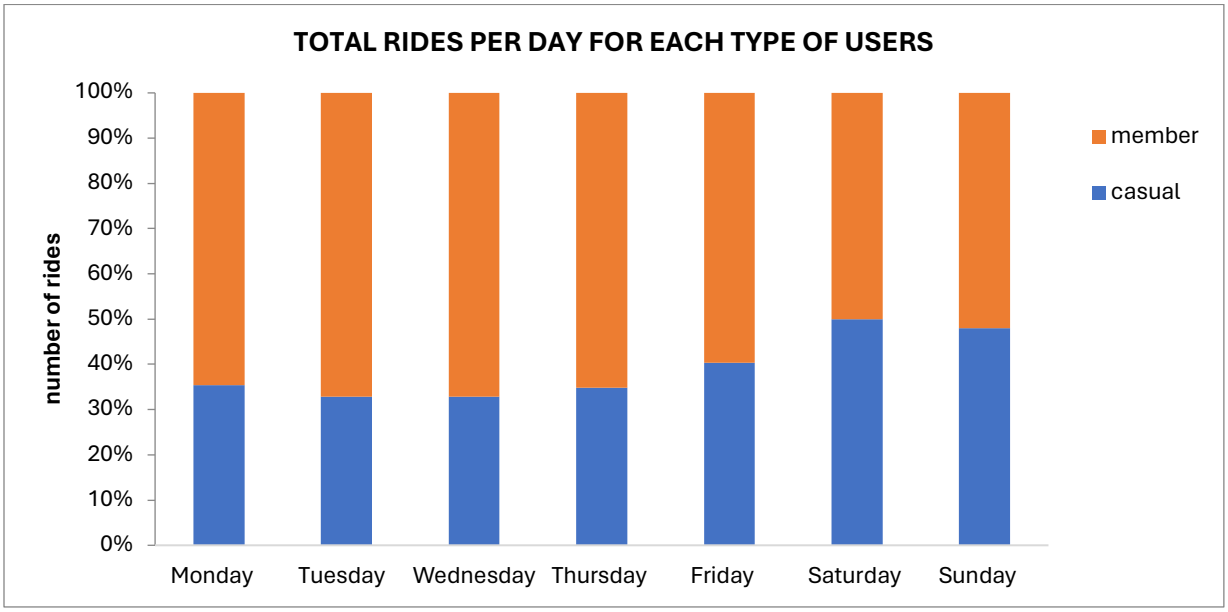
While docked bikes had the highest riding time among membership types and rideable bike types, the frequency of it was the lowest while members are the ones to use electric bike and classic bike more often throughout the year. The opposite trend with visualization 1 above.



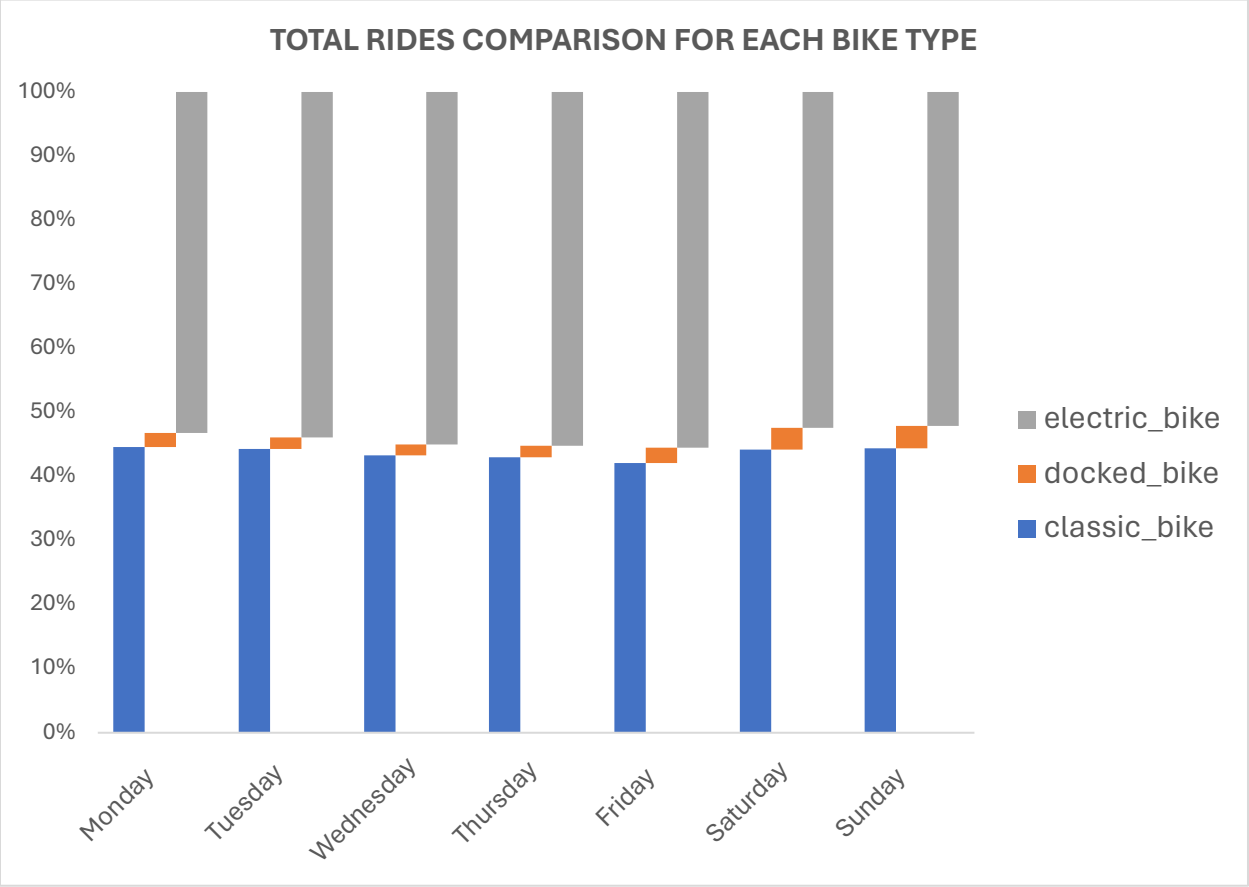
Visualization 3: Average ride time in minutes recorded throughout the week of 2022 and 2023.

Though the frequency of use among casual riders was lower than that of members, casual riders of the service accounted for the higher riding time during any day of the week recorded in the past year.

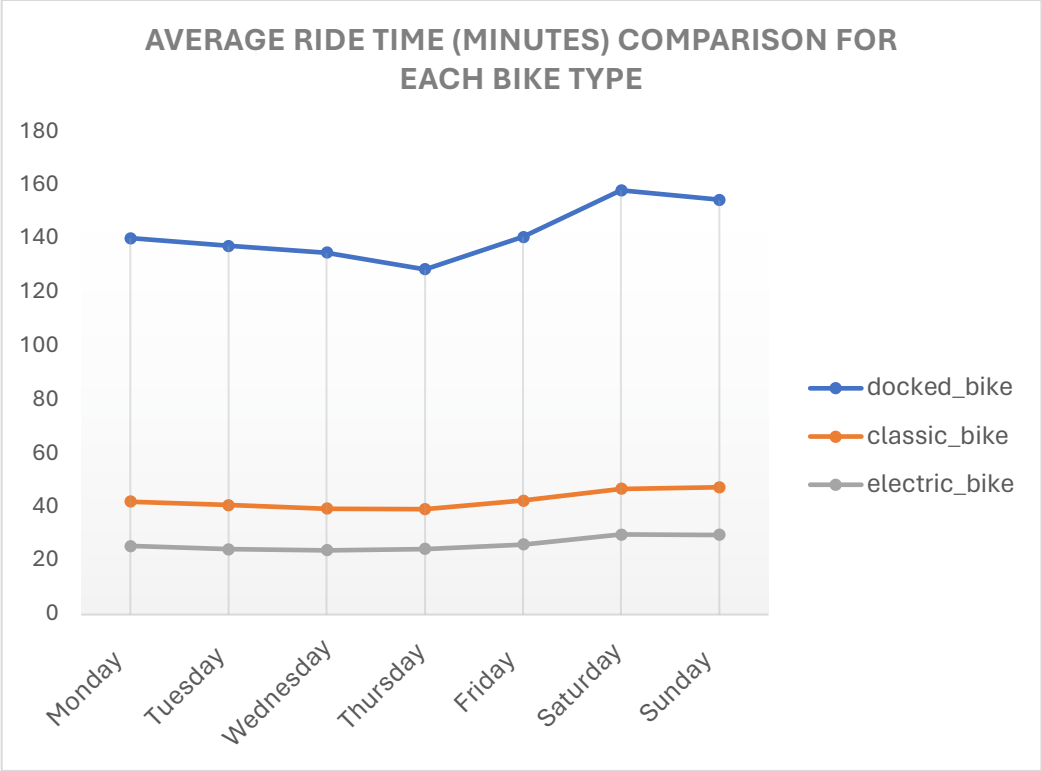
This suggests that promotions to attract casual riders become member should emphasize on the riding time rather than the frequency of use.



Visualization 4: The number of rides per day recorded among members and casual riders

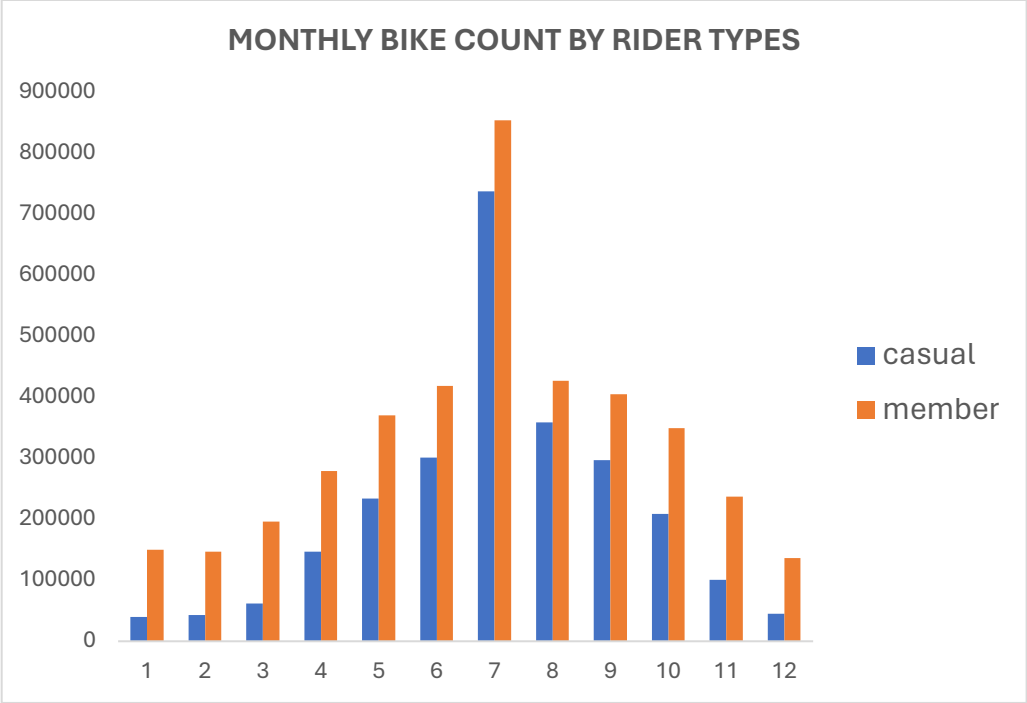


Visualization 5: the comparison of each rideable type recorded throughout the week

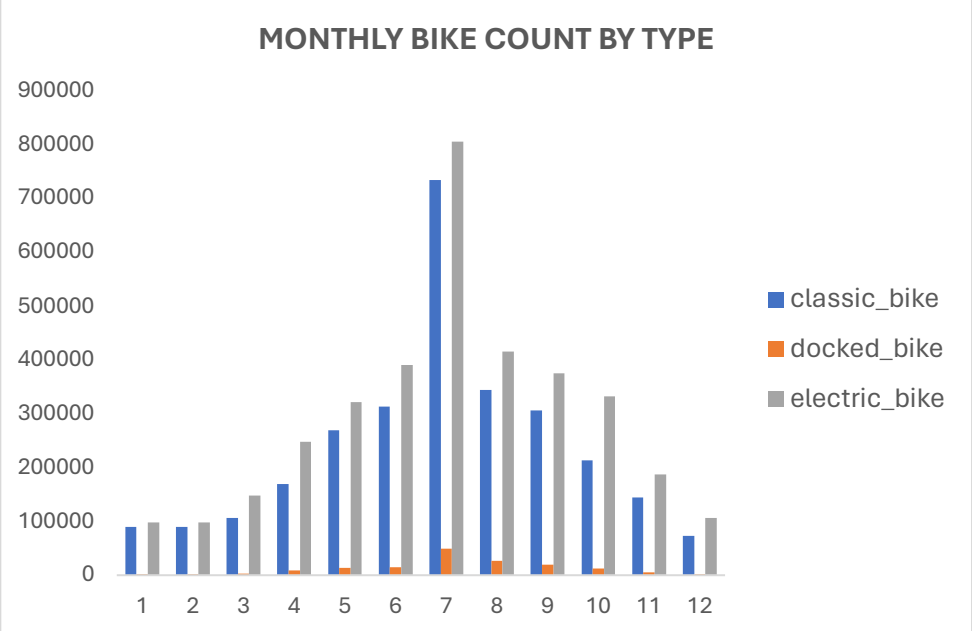


Visualization 6: Weekly average ride time in minutes for each rideable bike type

According to weekly report of the number of rides and riding time, Members tend to use the service more often than Casuals during Weekdays, while the number of casual riders increase during the weekend, which increase by more than 10% during Saturday and Sunday. This can suggest that casual riders are ones that use the service for tourism purposes. This idea can be further investigated with the to 20 starting and ending station between members and casual riders.

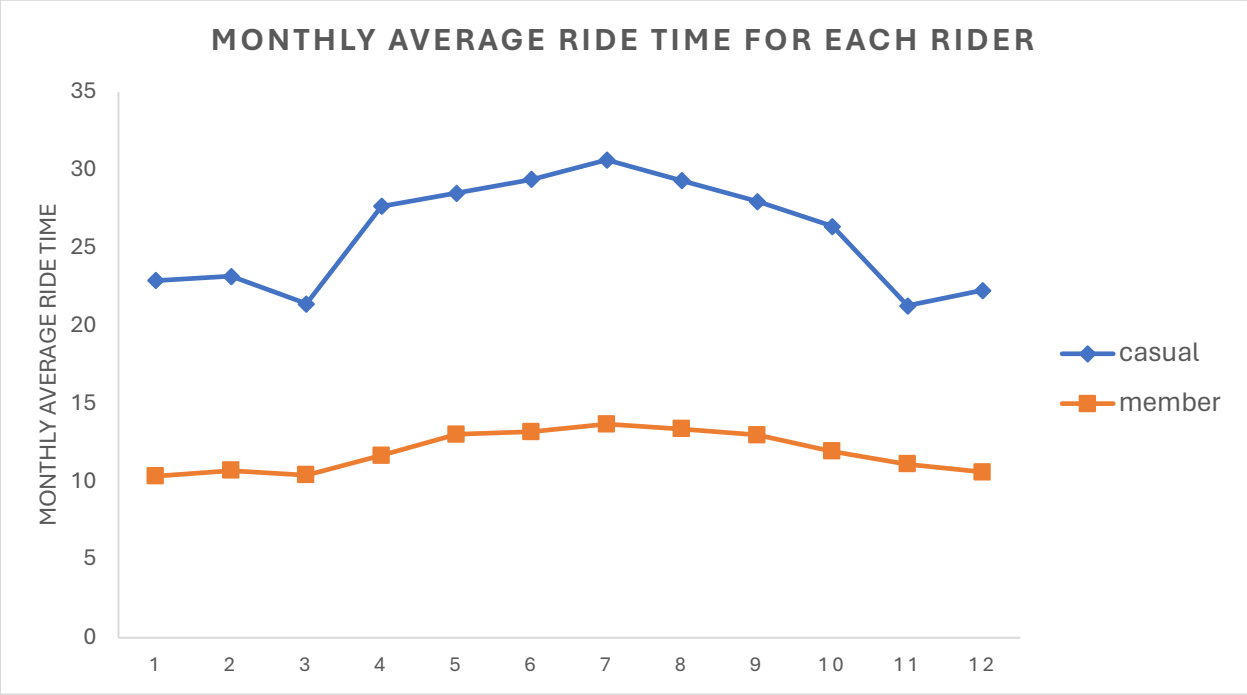


Visualization 7: Monthly bike count among members and casual riders

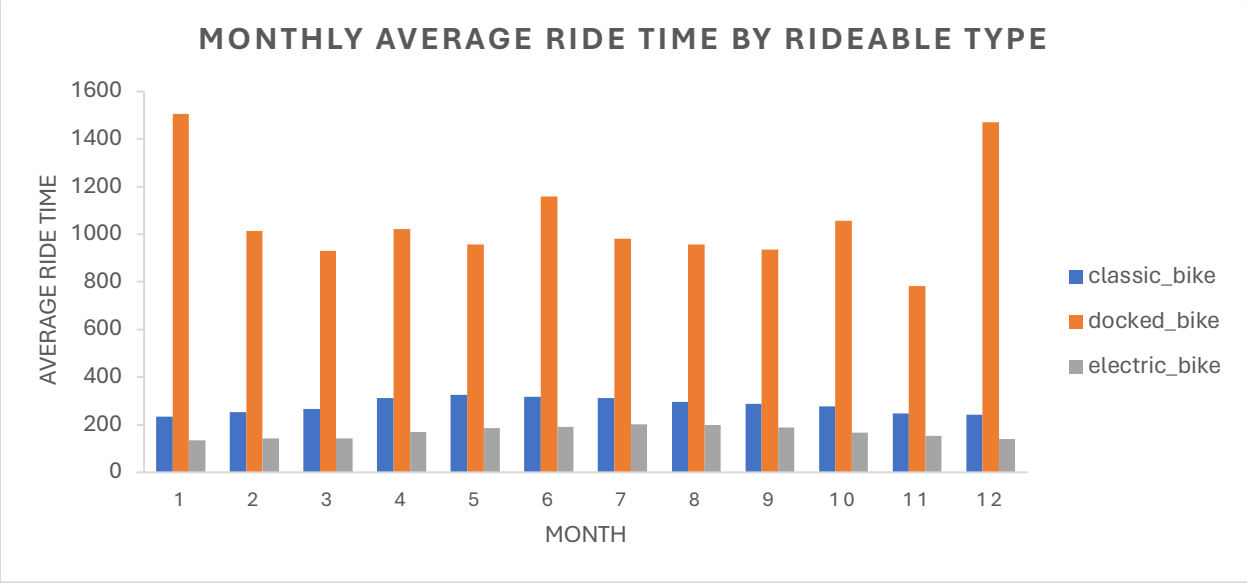


Visualization 8: Monthly bike count among rideable bike type

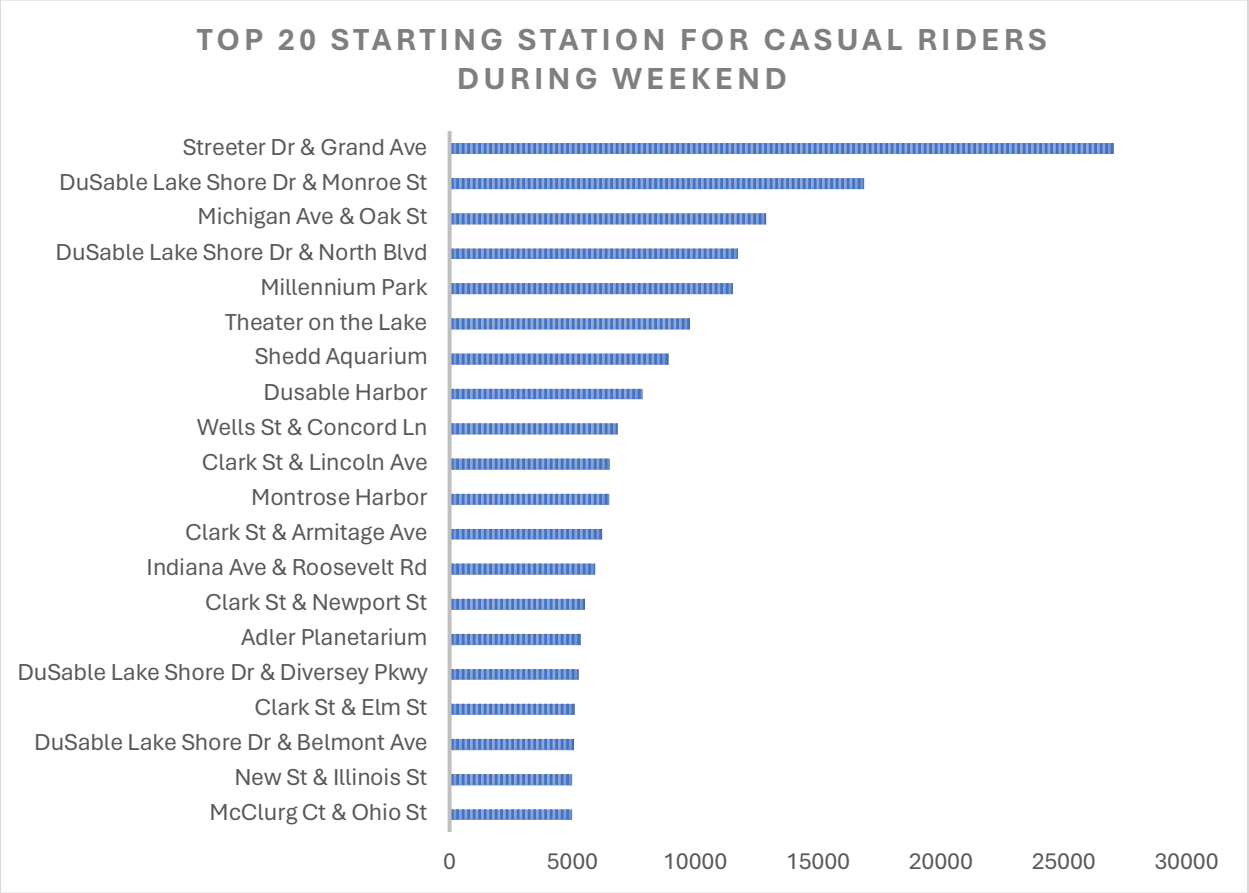
For 12-month report on the frequency usage of the service, most casual riders prefer to use Cyclistic services during Summer (June to August) since these months reported the highest number of rides in classic and electric bikes.

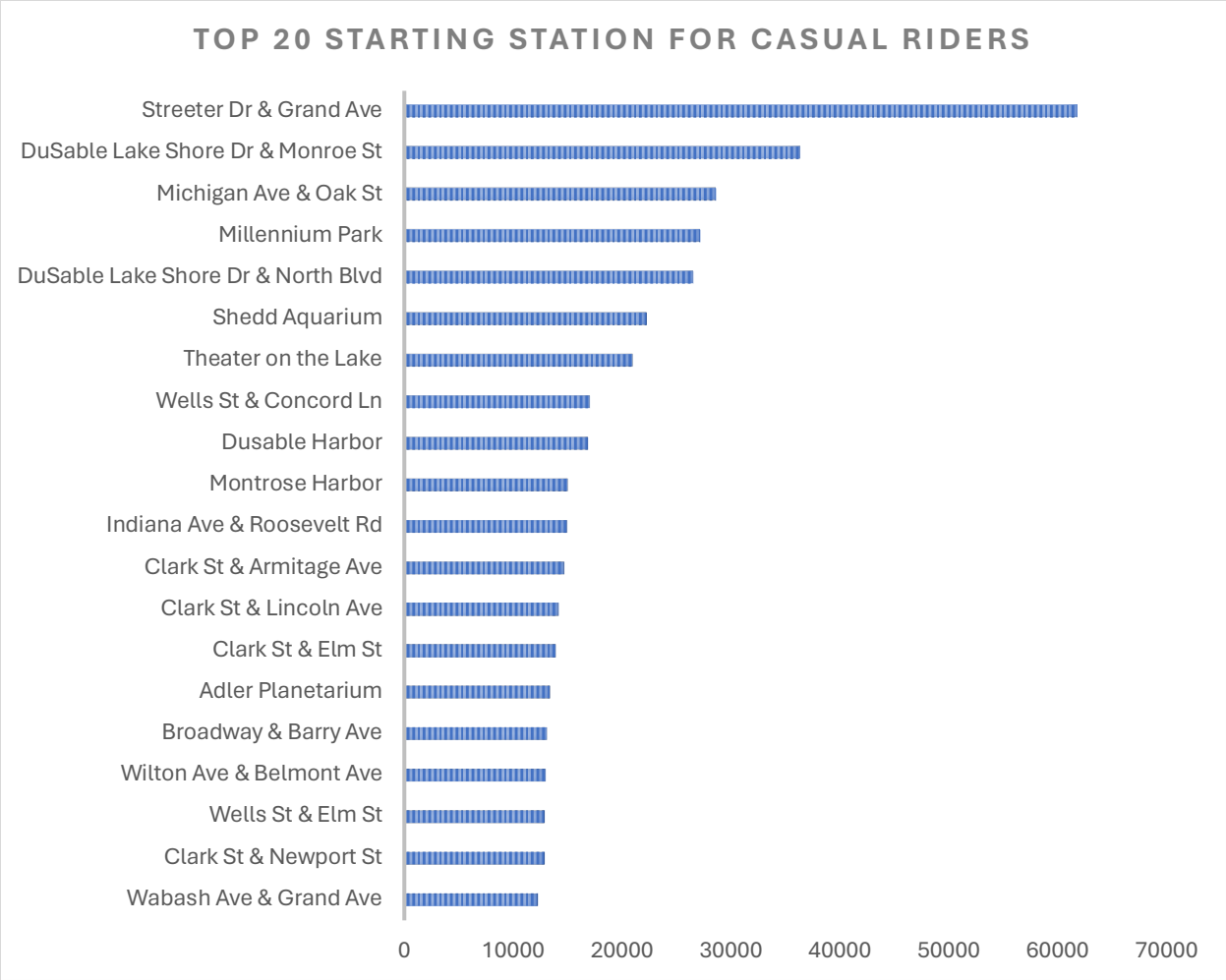


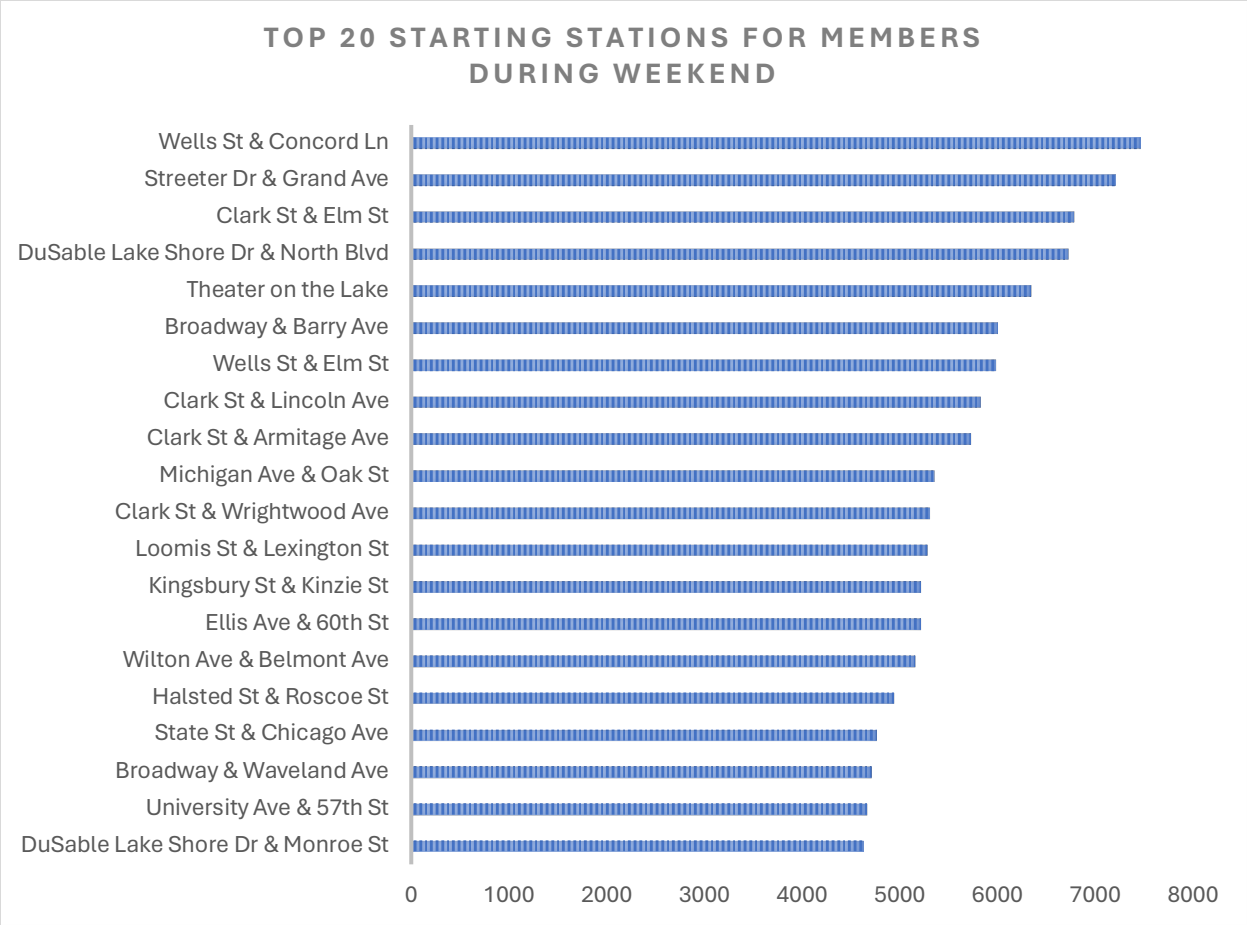
Visualization 9: monthly average ride time among membership type

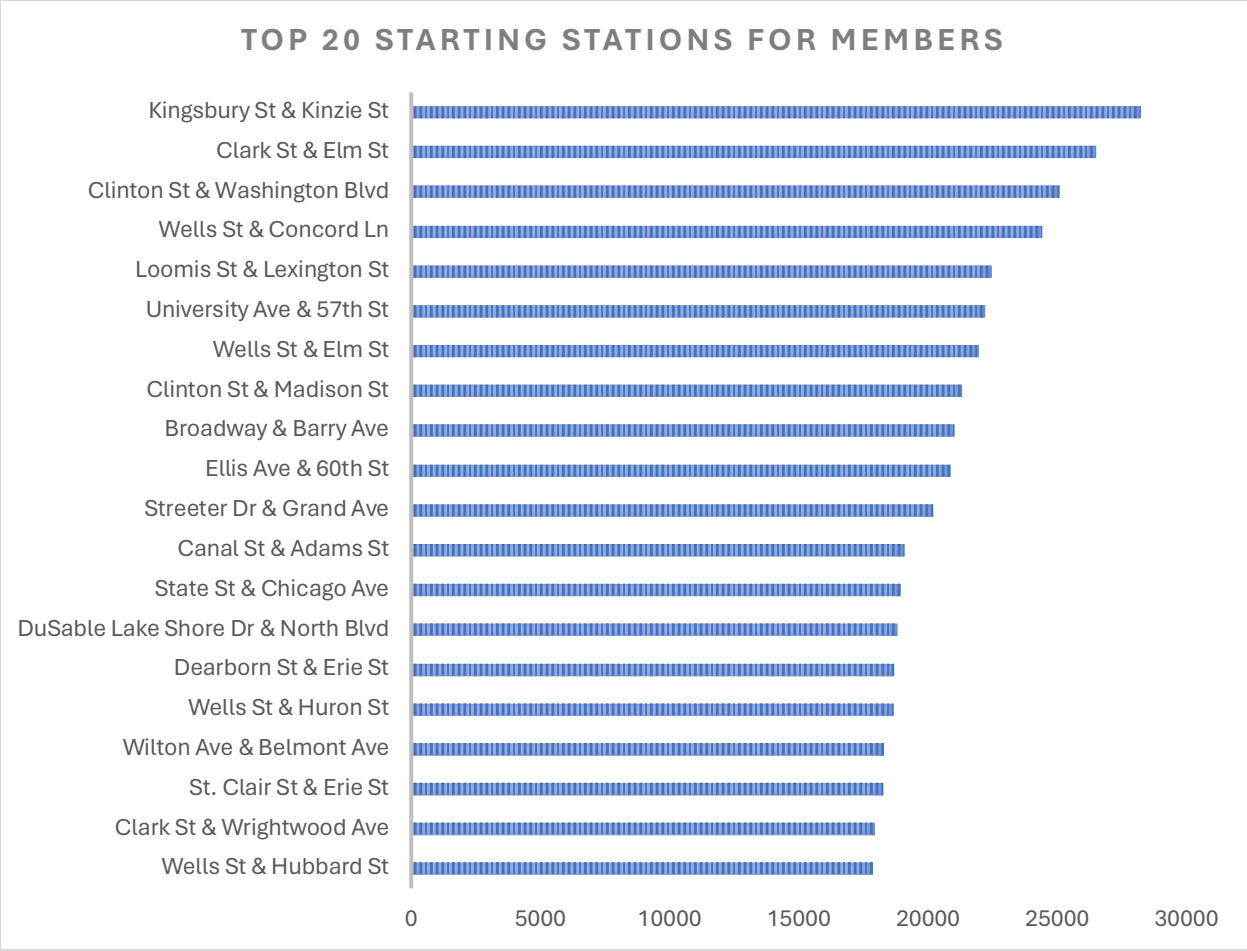


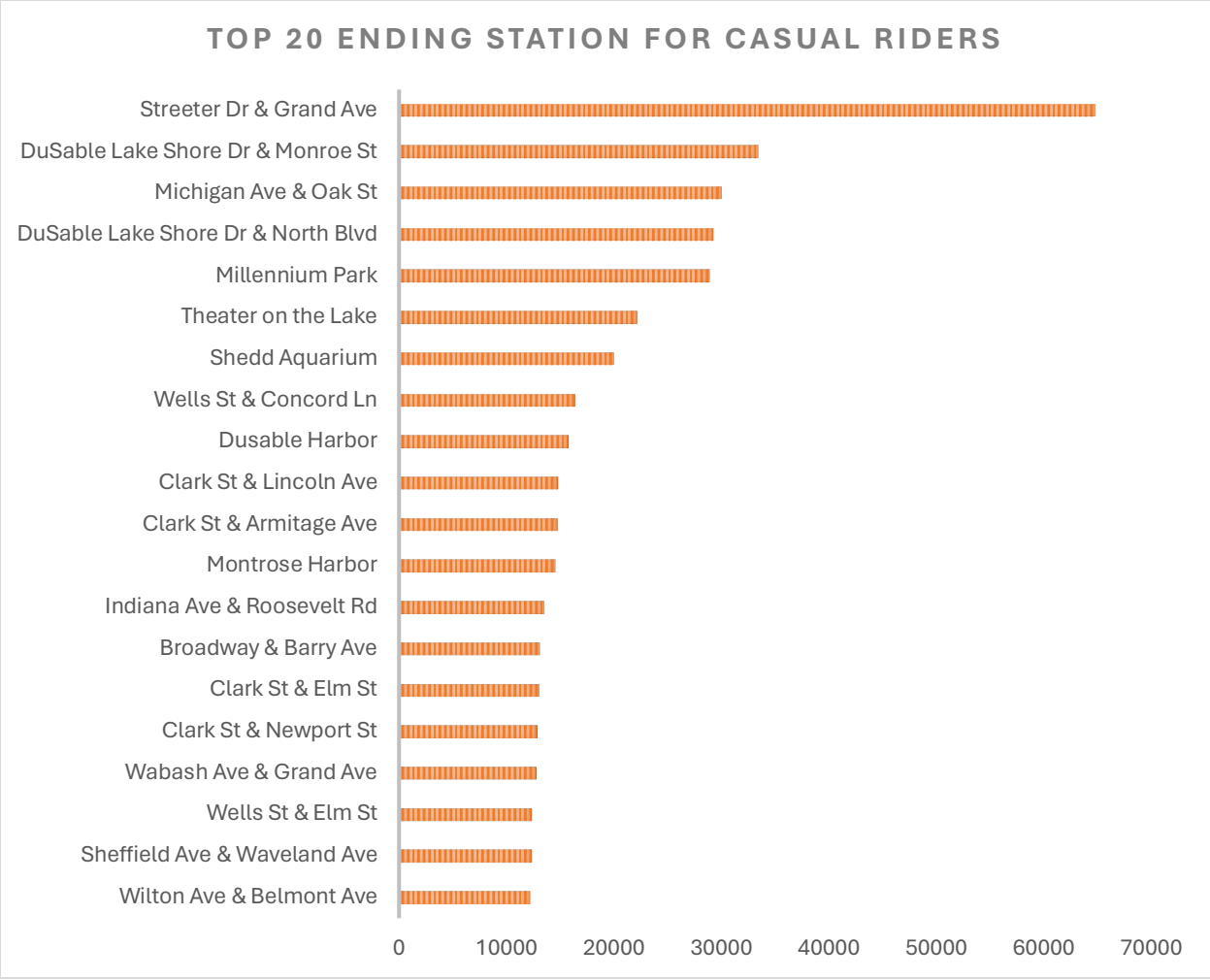
Visualization 10: monthly average ride time in minutes for each bike type

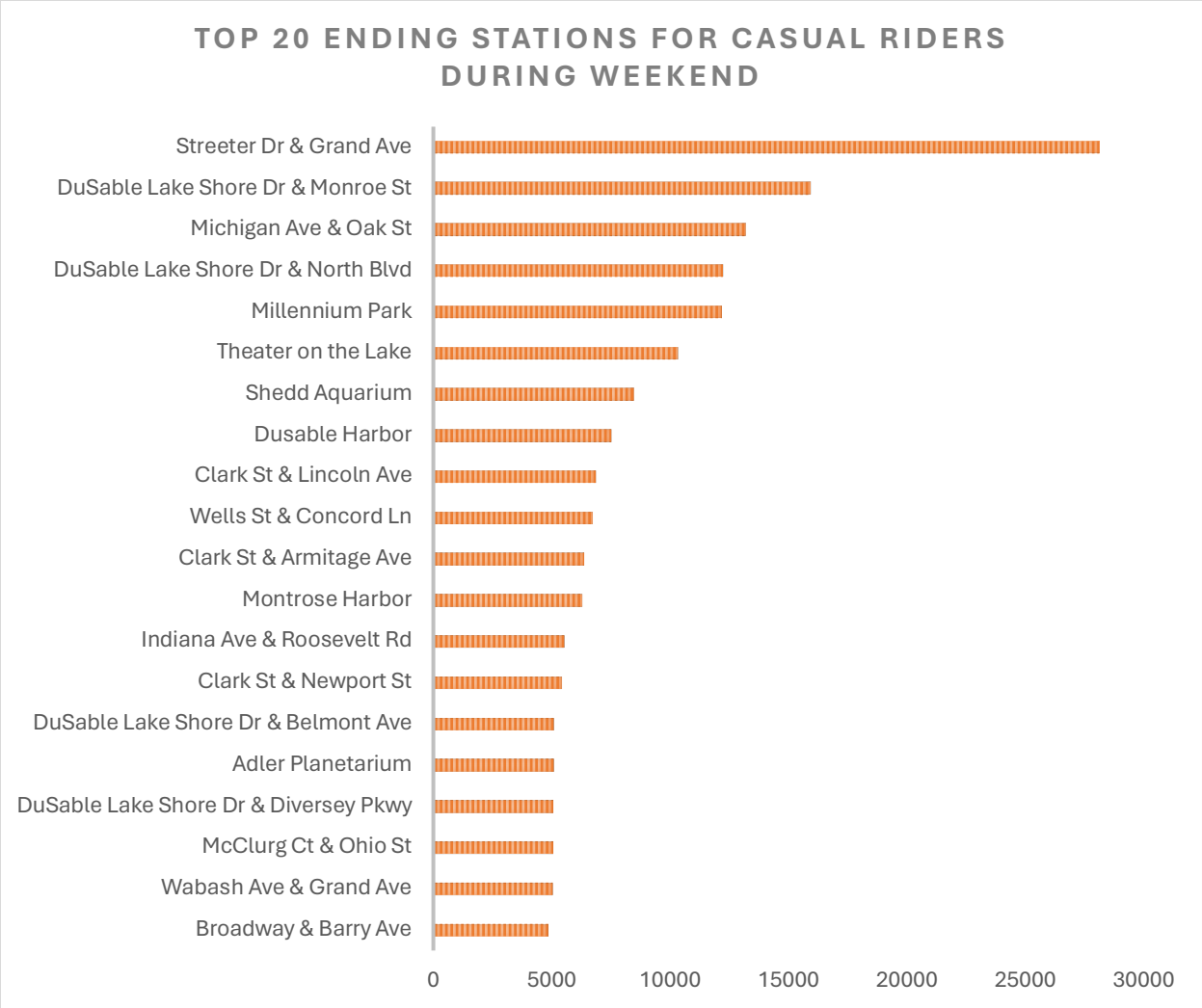


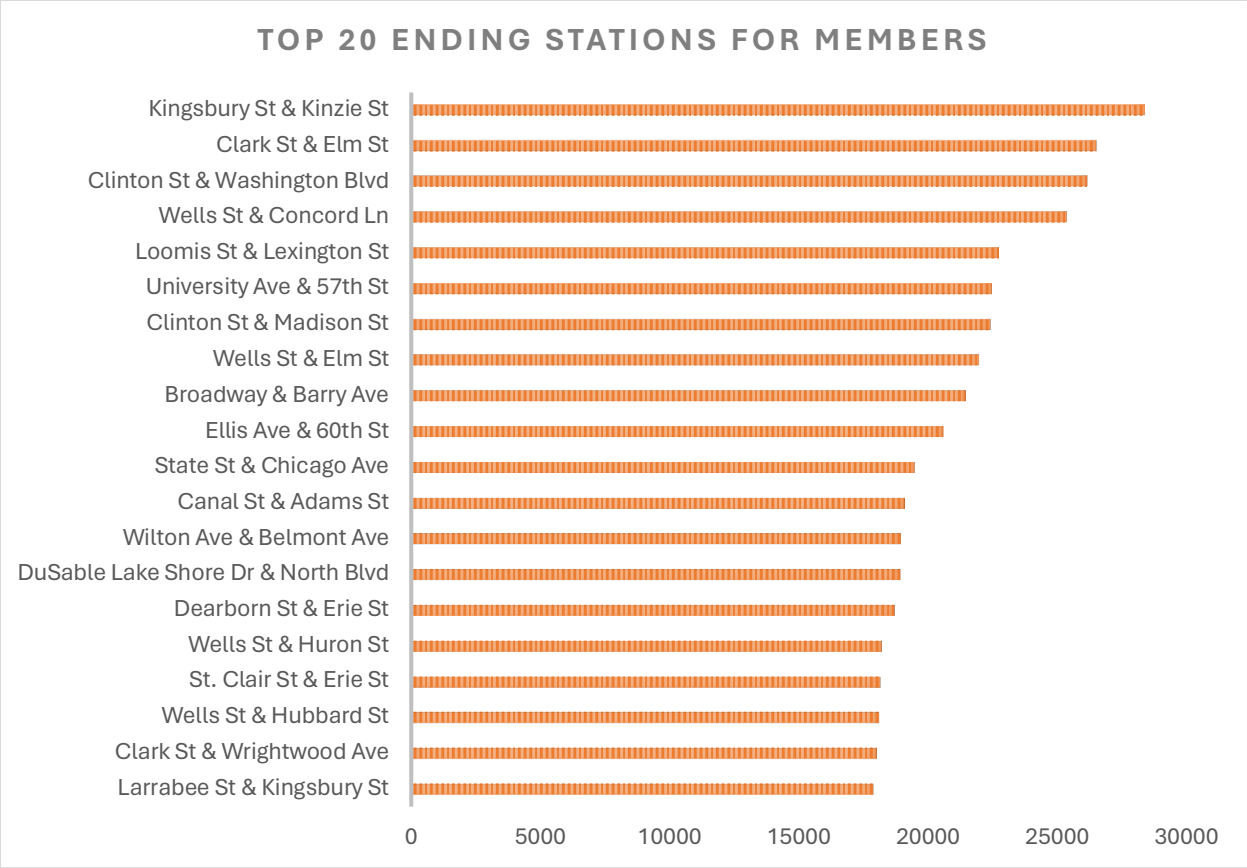

TOP 20 STARTING STATION FOR CASUAL RIDERS
Streeter Dr & Grand Ave
DuSable Lake Shore Dr & Monroe St
Michigan Ave & Oak St
Millennium Park
DuSable Lake Shore Dr & North Blvd
Shedd Aquarium
Theater on the Lake
Wells St & Concord Ln
Dusable Harbor
Montrose Harbor
Indiana Ave & Roosevelt Rd
Clark St & Armitage Ave
Clark St & Lincoln Ave
Clark St & Elm St
Adler Planetarium
Broadway & Barry Ave
Wilton Ave & Belmont Ave
Wells St & Elm St
Clark St & Newport St
Wabash Ave & Grand Ave
0
10000
20000
30000
40000
50000
60000
70000

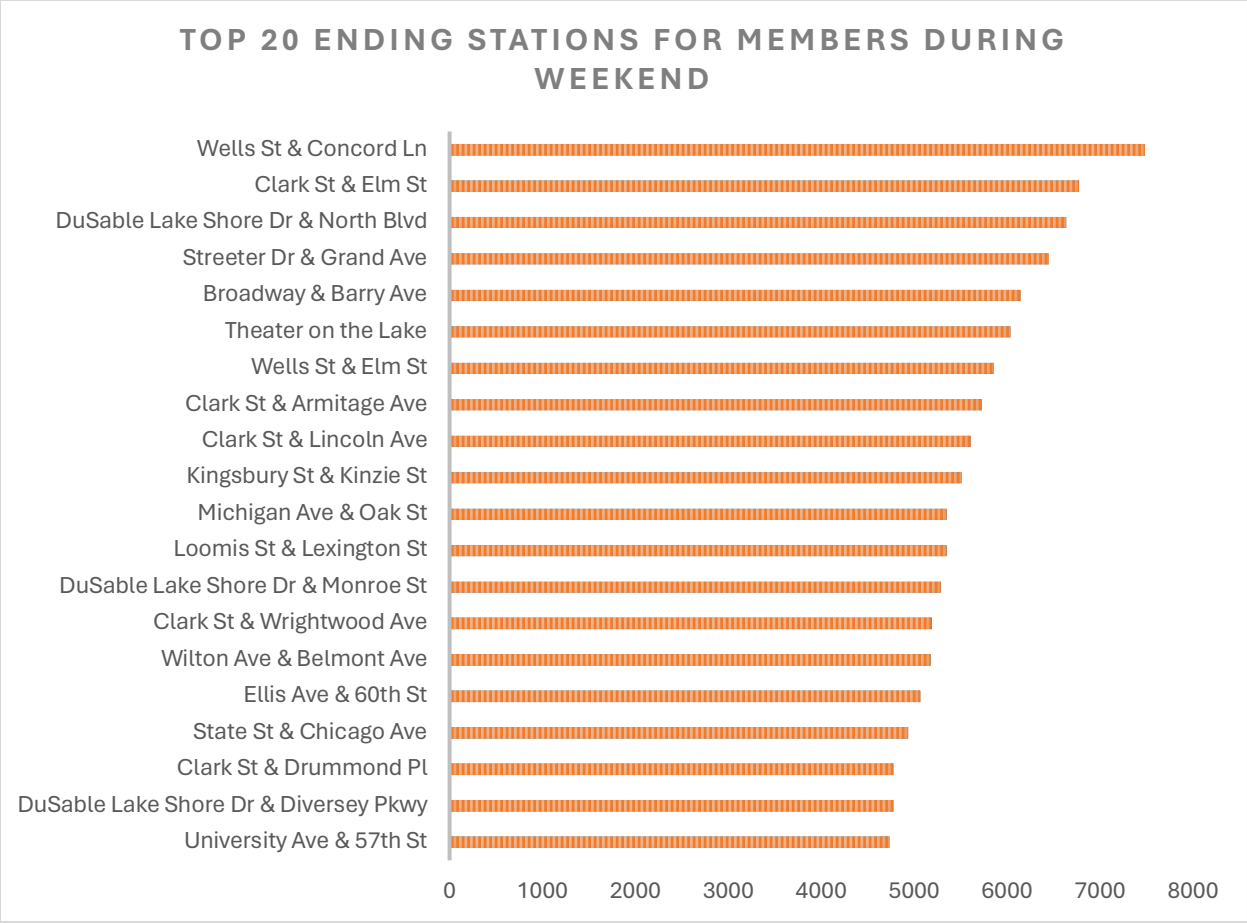

TOP 20 STARTING STATIONS FOR MEMBERS DURING WEEKEND
Wells St & Concord Ln
Streeter Dr & Grand Ave
Clark St & Elm St
DuSable Lake Shore Dr & North Blvd
Theater on the Lake
Broadway & Barry Ave
Wells St & Elm St
Clark St & Lincoln Ave
Clark St & Armitage Ave
Michigan Ave & Oak St
Clark St & Wrightwood Ave
Loomis St & Lexington St
Kingsbury St & Kinzie St
Ellis Ave & 60th St
Wilton Ave & Belmont Ave
Halsted St & Roscoe St
State St & Chicago Ave
Broadway & Waveland Ave
University Ave & 57th St
DuSable Lake Shore Dr & Monroe St
0
1000
2000
3000
4000
5000
6000
7000
8000


TOP 20 STARTING STATIONS FOR MEMBERS
Kingsbury St & Kinzie St
Clark St & Elm St
Clinton St & Washington Blvd
Wells St & Concord Ln
Loomis St & Lexington St
University Ave & 57th St
Wells St & Elm St
Clinton St & Madison St
Broadway & Barry Ave
Ellis Ave & 60th St
Streeter Dr & Grand Ave
Canal St & Adams St
State St & Chicago Ave
DuSable Lake Shore Dr & North Blvd
Dearborn St & Erie St
Wells St & Huron St
Wilton Ave & Belmont Ave
St. Clair St & Erie St
Clark St & Wrightwood Ave
Wells St & Hubbard St
0
5000
10000
15000
20000
25000
30000


TOP 20 ENDING STATION FOR CASUAL RIDERS
Streeter Dr & Grand Ave
DuSable Lake Shore Dr & Monroe St
Michigan Ave & Oak St
DuSable Lake Shore Dr & North Blvd
Millennium Park
Theater on the Lake
Shedd Aquarium
Wells St & Concord Ln
Dusable Harbor
Clark St & Lincoln Ave
Clark St & Armitage Ave
Montrose Harbor
Indiana Ave & Roosevelt Rd
Broadway & Barry Ave
Clark St & Elm St
Clark St & Newport St
Wabash Ave & Grand Ave
Wells St & Elm St
Sheffield Ave & Waveland Ave
Wilton Ave & Belmont Ave
0
10000
20000
30000
40000
50000
60000
70000

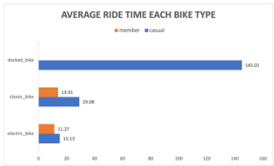

TOP 20 ENDING STATIONS FOR CASUAL RIDERS DURING WEEKEND
Streeter Dr & Grand Ave
DuSable Lake Shore Dr & Monroe St
Michigan Ave & Oak St
DuSable Lake Shore Dr & North Blvd
Millennium Park
Theater on the Lake
Shedd Aquarium
Dusable Harbor
Clark St & Lincoln Ave
Wells St & Concord Ln
Clark St & Armitage Ave
Montrose Harbor
Indiana Ave & Roosevelt Rd
Clark St & Newport St
DuSable Lake Shore Dr & Belmont Ave
Adler Planetarium
DuSable Lake Shore Dr & Diversey Pkwy
McClurg Ct & Ohio St
Wabash Ave & Grand Ave
Broadway & Barry Ave
0
5000
10000
15000
20000
25000
30000

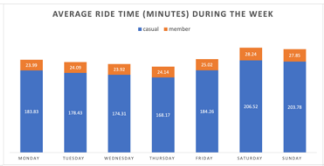

TOP 20 ENDING STATIONS FOR MEMBERS
Kingsbury St & Kinzie St
Clark St & Elm St
Clinton St & Washington Blvd
Wells St & Concord Ln
Loomis St & Lexington St
University Ave & 57th St
Clinton St & Madison St
Wells St & Elm St
Broadway & Barry Ave
Ellis Ave & 60th St
State St & Chicago Ave
Canal St & Adams St
Wilton Ave & Belmont Ave
DuSable Lake Shore Dr & North Blvd
Dearborn St & Erie St
Wells St & Huron St
St. Clair St & Erie St
Wells St & Hubbard St
Clark St & Wrightwood Ave
Larrabee St & Kingsbury St
0
5000
10000
15000
20000
25000
30000

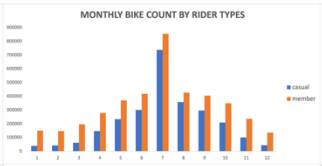

TOP 20 ENDING STATIONS FOR MEMBERS DURING WEEKEND
Wells St & Concord Ln
Clark St & Elm St
DuSable Lake Shore Dr & North Blvd
Streeter Dr & Grand Ave
Broadway & Barry Ave
Theater on the Lake
Wells St & Elm St
Clark St & Armitage Ave
Clark St & Lincoln Ave
Kingsbury St & Kinzie St
Michigan Ave & Oak St
Loomis St & Lexington St
DuSable Lake Shore Dr & Monroe St
Clark St & Wrightwood Ave
Wilton Ave & Belmont Ave
Ellis Ave & 60th St
State St & Chicago Ave
Clark St & Drummond Pl
DuSable Lake Shore Dr & Diversey Pkwy
University Ave & 57th St
0
1000
2000
3000
4000
5000
6000
7000
8000

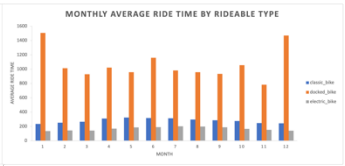
PHASE 6: ACT

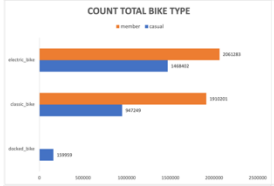
6.1. FINDINGS

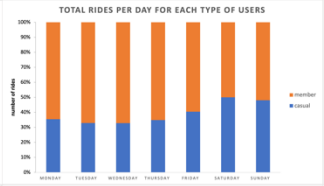


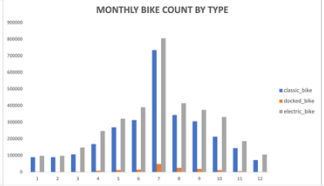


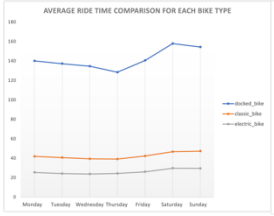


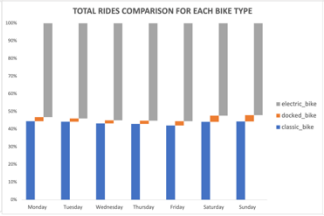


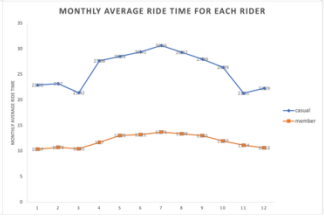


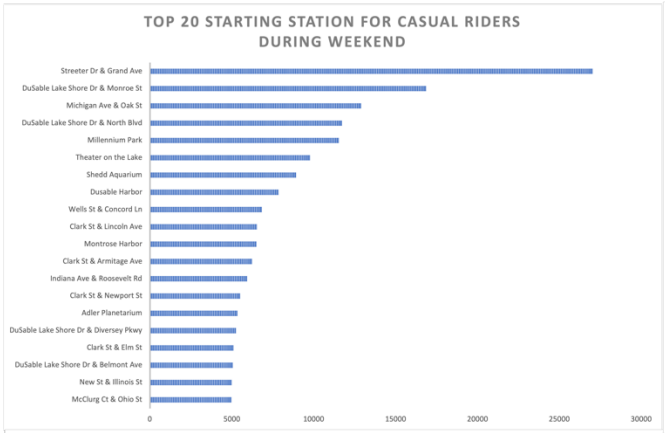


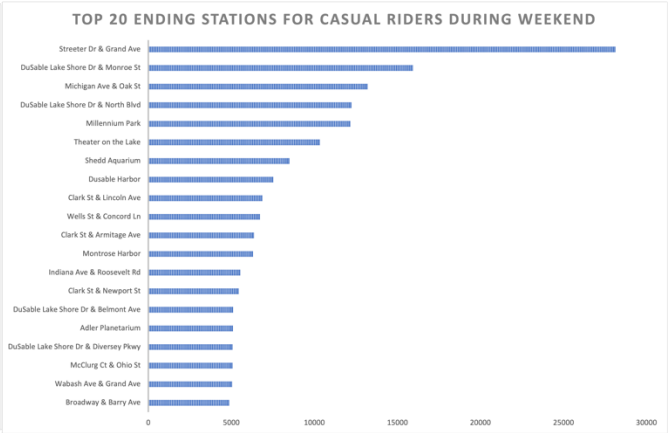


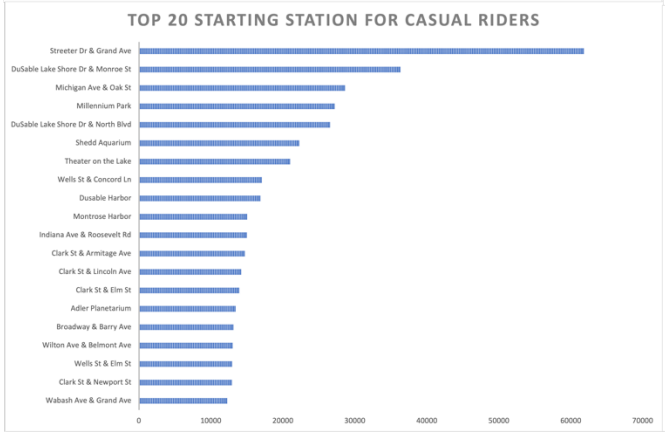


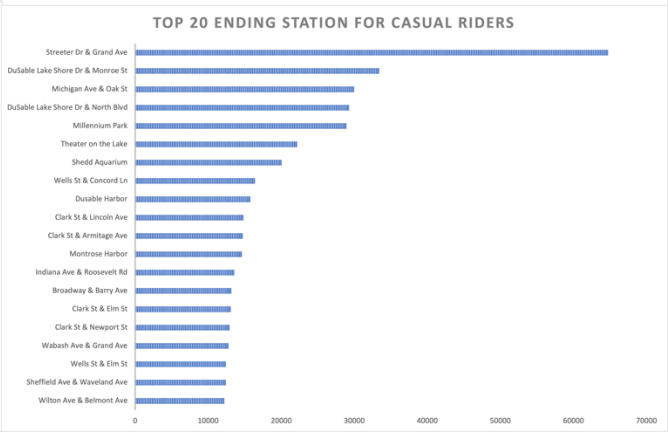


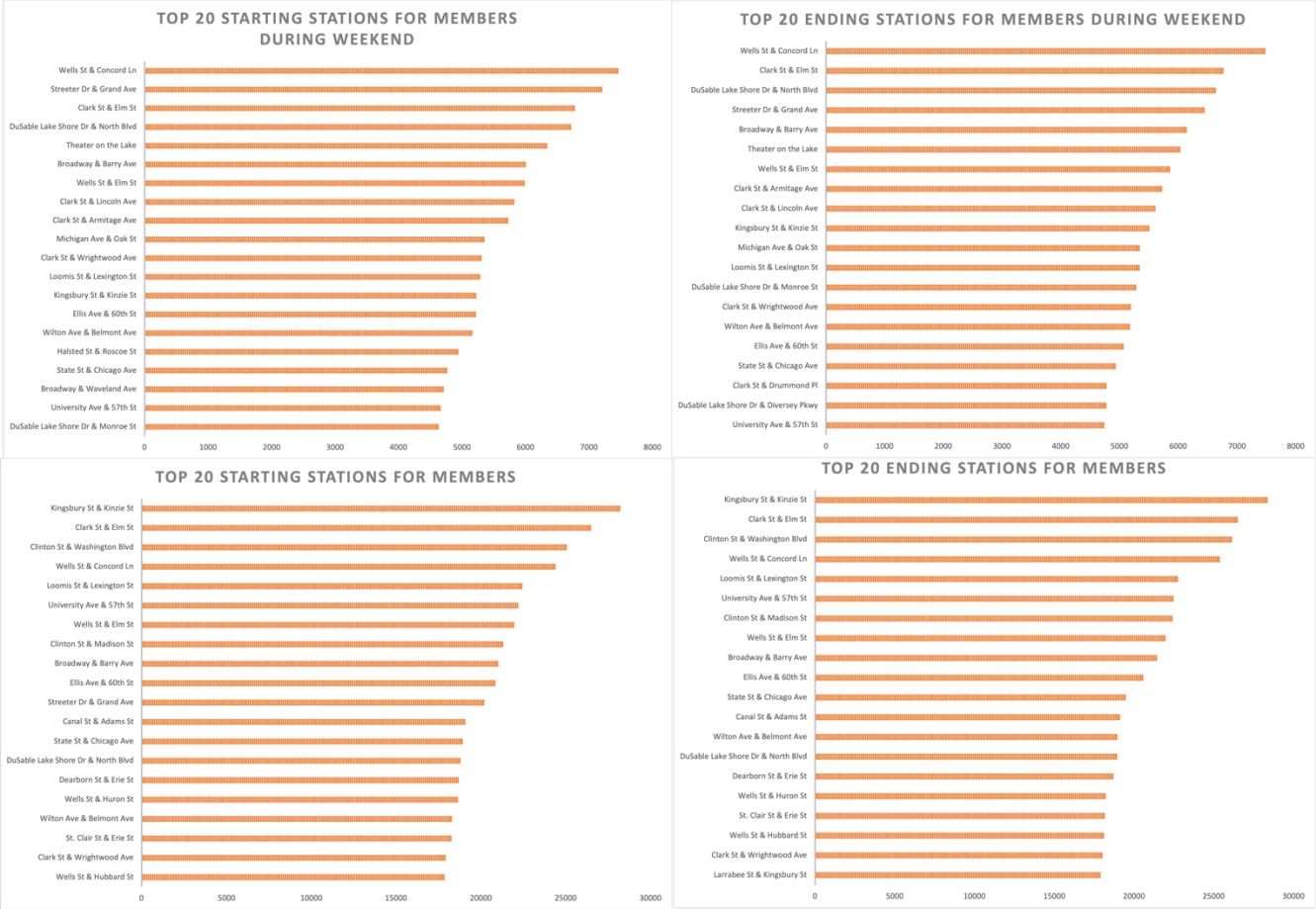












- Members of Cyclistic did not use docked_bike to travel, whereas this type of bike was popular among casual ones to travel great distances.

- Since docked bikes were used to travel far, this rideable type had the highest average hours of ride time in minutes (over 2 hours per ride) in monthly and weekly report

- However, the number of rides for said rideable type was the lowest, while classic and electric bikes were significantly higher in both casual riders and members

- For 12-month report on the frequency usage of the service, most casual riders prefer to use Cyclistic services during Summer (June to August) since these months reported the highest number of rides in classic and electric bikes.

- According to weekly report of the number of rides and riding time, Members tend to use the service more often than Casuals during Weekdays, while the number of casual riders increase during the weekend, which increase by more than 10% during Saturday and Sunday. This can

suggest that casual riders are ones that use the service for tourism purposes. This idea can be further investigated with the to 20 starting and ending station between members and casual riders.

Many Casual rides start and end near popular tourist attractions in Downtown and Lake Shore Chicago. Members seem to like bus stops near Metro stops and places with more office buildings than tourist attractions. A majority of casual riders appear to use the bikes for tourism, whereas members primarily use them for public transportation on a daily basis.

6.2. RECOMMENDATIONS

- Promotions should happen more often during the weekend as Casual riders are ones that use the service to travel to tourist attractions, and this frequencies increases during the weekend.

- Promotion for the service exclusively for new members in Summer should be prioritize as the number of people using the service will peak during that time.

- Focus more on the credit for how long a person use the service to travel to tourist attractions in Chicago

Hence, I recommend that tourists and locals be targeted separately in a two-pronged marketing campaign.

My recommendation for the tourism marketing is to highlight the value of repeat trips and offer an alternative "Weekend Membership" or multiple day pass to entice visitors to spend more. The data suggests tourism focused media (e.g. visitors guides, weekend guides, specific event guides) emphasizing weekends in the spring and summer months will be most effective, and promoting value and how "exploring the city by bike" is a great way to see the city.

According to the data, marketing to Chicago locals should focus on convenience and value. It is recommended to time the marketing in the spring, when the value proposition is the greatest, although further research into membership sign up timing could provide valuable insights. Also, further research into concerns locals may have about using the service (e.g. safety, stress, etc.) could help the marketing effort to overcome the most common objections to membership. I recommend geographically focused marketing near the most used stations, as well as further research into why these stations are so popular in hopes of developing other stations and their surrounding population.

APPENDIX

```
CREATE TABLE tripdata_2022_7
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

```
CREATE TABLE tripdata_2022_8
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```
CREATE TABLE tripdata_2022_9
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```sql
CREATE TABLE tripdata_2022_10
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```sql
CREATE TABLE tripdata_2022_11
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```sql
CREATE TABLE tripdata_2022_12
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```sql
CREATE TABLE tripdata_2023_1
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```sql
CREATE TABLE tripdata_2023_2
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```sql
CREATE TABLE tripdata_2023_3
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```sql
CREATE TABLE tripdata_2023_4
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```sql
CREATE TABLE tripdata_2023_5
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```sql
CREATE TABLE tripdata_2023_6
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```

SQL Copy Caption

```sql
CREATE TABLE tripdata_2023_7
(ride_id VARCHAR(225),rideable_type VARCHAR(225),started_at TIMESTAMP,
ended_at TIMESTAMP,start_station_name VARCHAR(225),start_station_id VARCHAR(225),
end_station_name VARCHAR(225),end_station_id VARCHAR(225),start_lat DOUBLE PRECISION,
start_lng DOUBLE PRECISION,end_lat DOUBLE PRECISION,end_lng DOUBLE PRECISION,
member_casual VARCHAR(225));
```