Cyclistic Bike Share

BUSINESS STATEMENT

Cyclistic operate a bike sharing program where uses can be broken down into two main groups: those who pay for annual memberships (members) and those who use bikes on an ad-hoc basis through single ride passes or single day passes (casual riders). The purpose of this analysis is to determine how these two groups use the bikes differently using data provided directly by cyclistic from the previous 12 months.

SUMMARY AND RECCOMENDATIONS

The analysis results given below paint a picture of two distinct groups of cyclistic customers. Casual users appear far more likely to view the service as there for recreational activity with longer trips peaking on the weekends in warm weather. Members appear to view the service more as a replacement for walking or driving: taking shorter trips, more frequently at commuting times, and almost always returning the bike to a different station than where it was collected.

1) Communicate to casual users on the benefits of commuting by bike that the classic bike does not require returning to a docking station.
2) Communicate to casual users on the variety of drop off locations when they hire a bike.
3) Encourage more of a 'little and often' mentality by communicating the unlimited hiring available when a member.

ANALYSIS RESULTS

Casual users used the bike share service fewer times in the past year than bike share members (Figure 1).

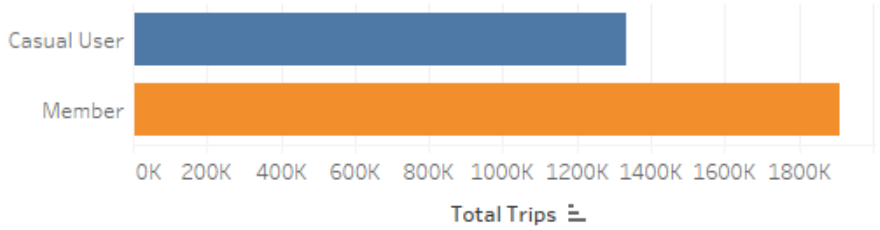

Figure 1 – Total trips taken by members and casual users April 2020 - March 2021

However casual users used the bikes for longer (Figure 2). The average trip time for casual users was approximately three times as long as that for members.

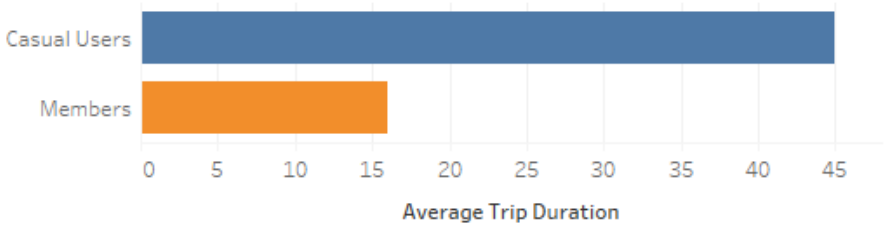

Figure 2 – Average trip duration in minutes for casual users and members.

Members and casual users also showed divergent habbits when comparing the time of day and day of the week most trips took place. Casual users preferred the weekend, particularly around 2pm on a Saturday. Members most used times were during commuting hours on weekdays (Figure 3).

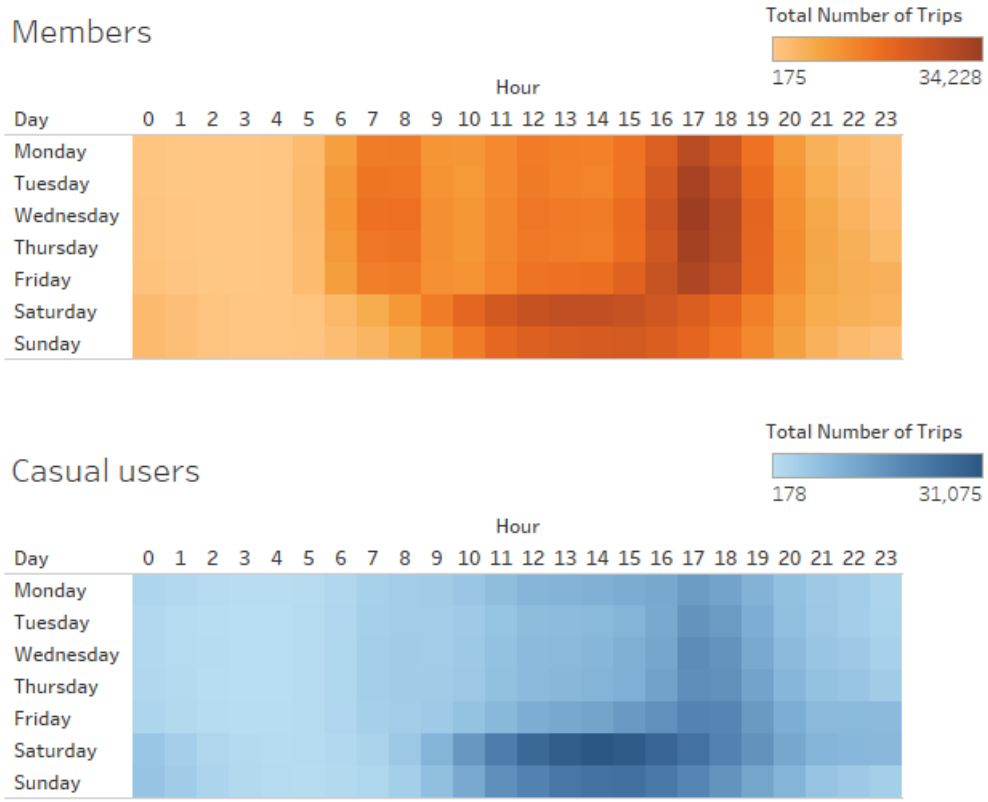

Figure 3 – Heatmaps detailing the number of trips starting during each hour of each day of the week.

There were also more subtle variations between members and causal users. While the majority of both groups used docked bicycles members were more far more likely than casual users to opt for the classic bike (Figure 4).

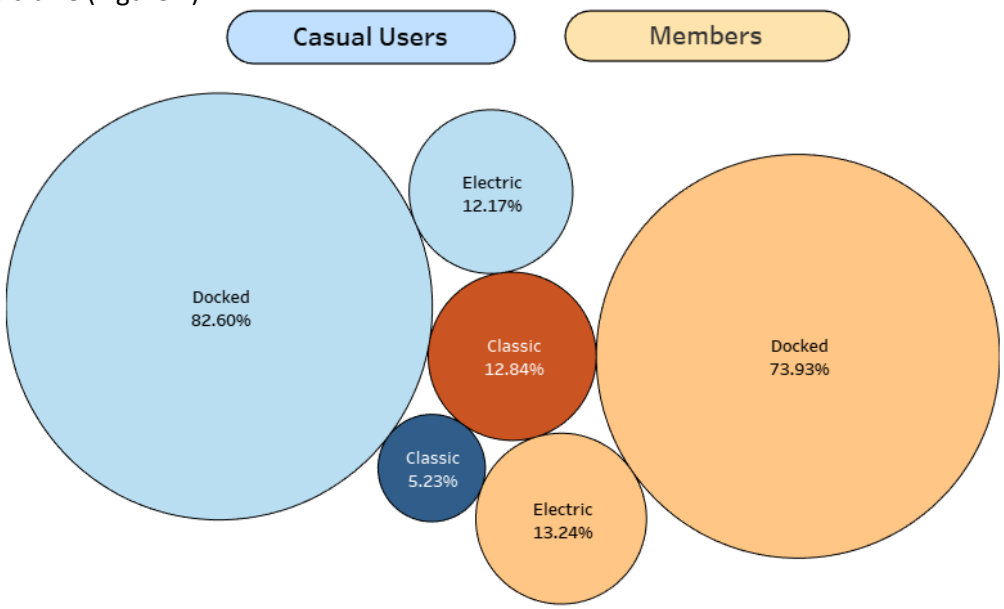

Figure 4 – Preferences by vehicle type for members and casual users.

Both casual users and members are likely to return the bike to a different place than where they collected it, casual users three times more likely than members to go against this trend – returning the bike to the same position as they collected it (Figure 5).

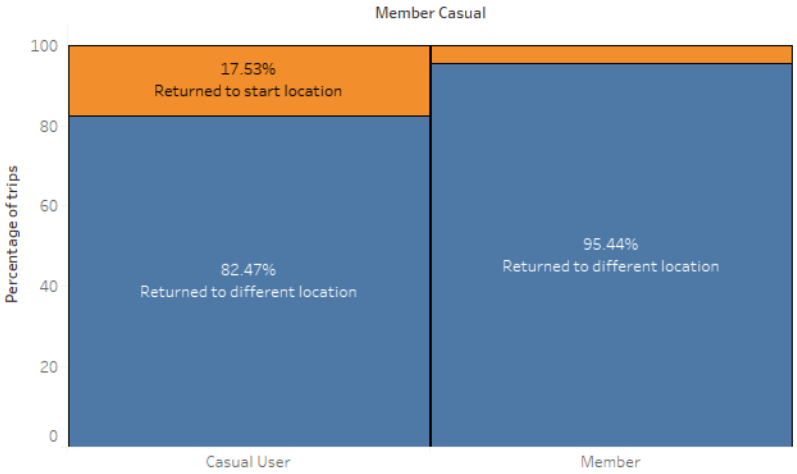

Figure 5 - Percentage of users returning the bike to the start position.

Finally, both members and casual users are affected significantly by lower temperatures, however, casual users are faster to reduce the number of trips they take when temperatures are falling (Figure 6).

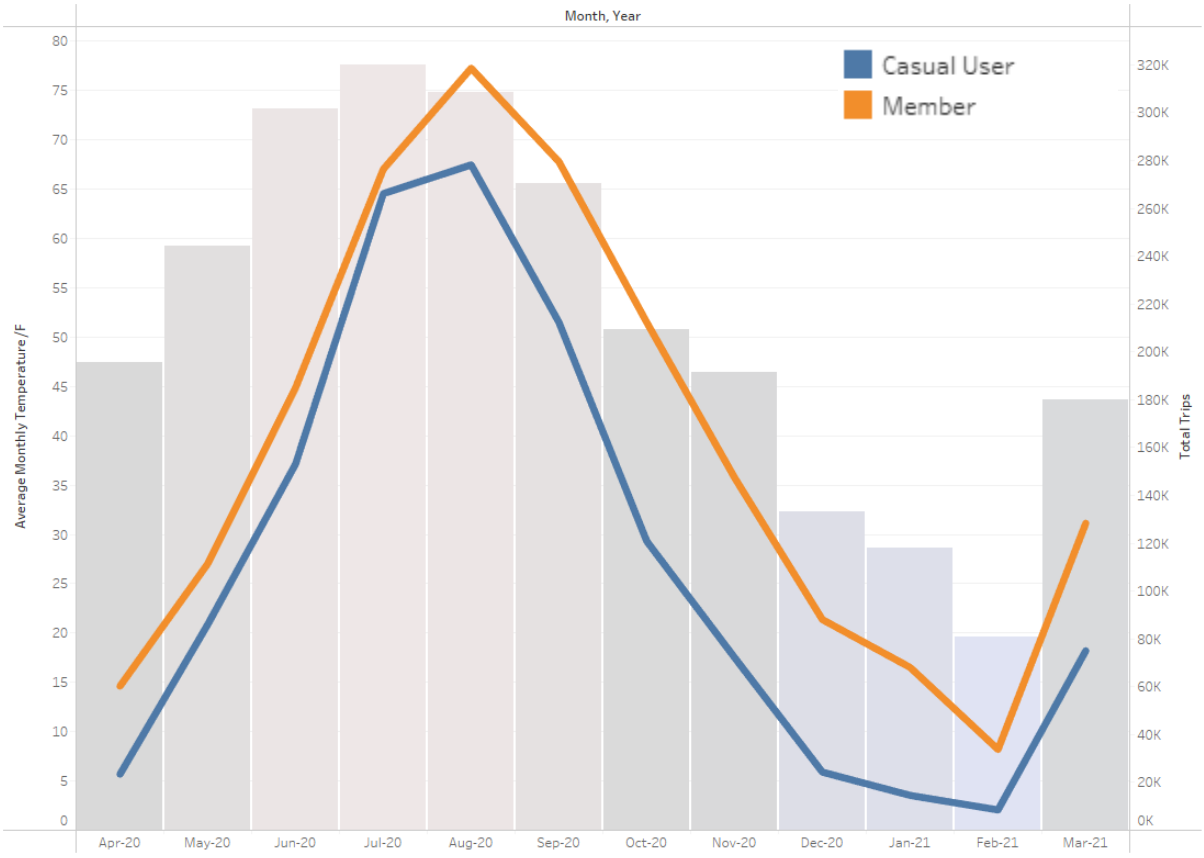

Figure 6 – The total number of trips members and casual users made for each month of the year, compared with the average temperature for each month.

DATA SET

Data provided is stored as zipped csv files on the company's own website (https://divvy-tripdata.s3.amazonaws.com/index.html). The data set consisted of the previous 12 months of cyclistic trip data (Apr-2020 – Mar-2021). This data set totalled 3,489,748 rows of data organised by the following columns:

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

Data appeared reliable, seemingly collected automatically as each ride began and ended. Originally in that it is a primary source from cyclistic. Data appeared to be comprehensive covering every journey from the previous 12 months and current, being regularly updated.

The Licence for the data is provided at https://www.divvybikes.com/data-license-agreement allowing 'non-exclusive, royalty-free, limited, perpetual license to access, reproduce, analyze, copy, modify, distribute in your product or service and use the Data for any lawful purpose'

DATA CLEANSING

INITIAL DATA SET

Initial inspection of the data revealed:

- Missing values for start_station_name, end_station name, start_station_id, end_station_id, start_lat, start_lng, end_lat, end_lng.
- Duplicated ride_id values
- Started_at values being later than ended_at values.

DATA CLEANSING STEPS

Data cleansing was completed using MySQL, the exact code given in Appendix 1, data cleansing sql.

10552 records removed due to start time being later than end time.

1024 records healed that were missing start_station_id or end_station_id where the corresponding data was available within other records.

194,087 records removed where data was missing for start_station_name, start_station_id, end_station_name, or end_station_id.

40,882 records removed where the bike was collected and returned to the same station within 1 minute.

DATA VALIDATION

Data validation checked for duplicated ride_id values, blank entities, null entites, and total remaining records, MySQL code given in

Appendix 2, data validation sql. Further a sampling of (414) records representing a 95% confidence interval and 5% margin of error was exported and checked manually.

Duplicates	Nulls	Blanks	Total Records
0	0	0	3244226

The remaining data represented 93% of the original source.

APPENDIX 1, DATA CLEANSING SQL

```
#REMOVING VALUES WHERE START TIME IS GREATER THAN END TIME.
CREATE TABLE IF NOT EXISTS DeletedTime_StartLaterthanEnd AS (SELECT * FROM
  trip_data
WHERE
  TIMESTAMPDIFF(SECOND,
    started_at,
    ended_at) < 0);

DELETE FROM trip_data
WHERE
  TIMESTAMPDIFF(SECOND,
  started_at,
  ended_at) < 0;

#HEALING DATA WHERE STATION IDS ARE MISSING
UPDATE trip_data
SET
  start_station_id = 20252
WHERE
  start_station_name = 'W Oakdale Ave & N Broadway'
    AND start_station_id = '';
UPDATE trip_data
SET
  end_station_id = 20252
WHERE
  end_station_name = 'W Oakdale Ave & N Broadway'
    AND end_station_id = '';

UPDATE trip_data
SET
  start_station_id = 20254
WHERE
  start_station_name = 'W Armitage Ave & N Sheffield Ave'
    AND start_station_id = '';
UPDATE trip_data
SET
  end_station_id = 20254
WHERE
  end_station_name = 'W Armitage Ave & N Sheffield Ave'
    AND end_station_id = '';

#REMOVING DATA WHERE START AND END LOCATIONS ARE NOT SUFFICIENTLY KNOWN.
CREATE TABLE IF NOT EXISTS deleted_location_info AS (SELECT * FROM
  trip_data
WHERE
  start_station_name = '0'
    OR end_station_name = '0');

DELETE FROM trip_data
WHERE
```

```
   start_station_name = '0'
   OR end_station_name = '0'
;
#REMOVING DATA WHERE BICYCLE WAS RETURNED TO THE PLACE OF ORIGIN IN UNDER A MINUTE
CREATE TABLE IF NOT EXISTS immediate_returns AS (SELECT * FROM
   trip_data
WHERE
   TIMESTAMPDIFF(SECOND,
      started_at,
      ended_at) / 60 < 1
      AND start_station_name = end_station_name)
;
DELETE FROM trip_data
WHERE
   TIMESTAMPDIFF(SECOND,
   started_at,
   ended_at) / 60 < 1
   AND start_station_name = end_station_name;
```

APPENDIX 2, DATA VALIDATION SQL

```
create table if not exists data_validation (
        duplicates integer,
   nulls integer,
   blanks integer,
   total_distinct integer
   );

#TEST 1: DUPLICATES
insert into data_validation(duplicates)
        select count(Q.cnt) from (
   SELECT ride_id,count(ride_id) as cnt
        FROM trip_data
        GROUP BY ride_id
        HAVING (cnt> 1)) as Q
;


#TEST 2: CHECK FOR NULL CELLS ANYWHERE
update data_validation set data_validation.nulls =
(SELECT
   count(ride_id)
FROM
   trip_data
WHERE
   ride_id IS NULL OR rideable_type IS NULL
      OR started_at IS NULL
      OR ended_at IS NULL
      OR start_station_id IS NULL
      OR start_station_name IS NULL
      OR end_station_name IS NULL
      OR end_station_id IS NULL
      OR start_lat IS NULL
      OR start_lng IS NULL
      OR end_lat IS NULL
      OR end_lng IS NULL
      OR member_casual IS NULL)
      ;


#TEST 3: CHECK FOR BLANK CELLS ANYWHERE
update data_validation set data_validation.blanks =
(SELECT
   count(ride_id)
FROM
   trip_data
WHERE
   rideable_type = ''
      OR start_station_id = ''
      OR start_station_name = ''
      OR end_station_name = ''
```

```
    OR end_station_id = ''
    OR start_lat = ''
    OR start_lng = ''
    OR end_lat = ''
    OR end_lng = ''
    OR member_casual = '');
```

#TEST 4: TOTAL NUMBER OF DISTINCT VALUES REMAINING

```
update data_validation set data_validation.total_distinct =
(select count(ride_id)
from trip_data);
```

#TEST 5: SHOW RANDOM SAMPLING OF DATA FOR MANUAL VERIFICATION (SAMPLE SIZE WITH 95% CI AND 5% MARGIN OF ERROR

```
select * from trip_data where rand()<0.000119;
```

APPENDIX 3, ANALYSIS SQL

```
#MONTHS OF THE YEAR
#RETRIVING THE NUMBER OF TRIPS, AVERAGE LENGTH OF EACH TRIP, AND AVERAGE TRIP DISTANCE(BY CO-ORDINATES WHEN START STATION IS DIFFERENT TO END STATION)
#FOR EACH MONTH IN THE YEAR

SELECT
  a.member_casual,
  a.month_number,
  a.month_text,
  a.total_trips,
  a.average_trip_duration,
  b.distance_ridden_km
FROM
  (SELECT
    member_casual,
      MONTH(started_at) AS month_number,
      MONTHNAME(started_at) AS month_text,
      COUNT(ride_id) AS total_trips,
      ROUND(AVG(TIMESTAMPDIFF(MINUTE, started_at, ended_at)), 0) AS average_trip_duration
  FROM
    trip_data
  GROUP BY member_casual , MONTH(started_at)
  ORDER BY member_casual , month_number ASC) AS a
    LEFT JOIN
  (SELECT
    member_casual,
      MONTH(started_at) AS month_number,
      MONTHNAME(started_at) AS month_text,
      ROUND(AVG(ST_DISTANCE_SPHERE(POINT(start_lng, start_lat), POINT(end_lng, end_lat))) /
1000, 1) AS distance_ridden_km
  FROM
    trip_data
  WHERE
    ST_DISTANCE_SPHERE(POINT(start_lng, start_lat), POINT(end_lng, end_lat)) > 0
  GROUP BY member_casual , MONTH(started_at)
  ORDER BY member_casual , month_number ASC) AS b ON a.member_casual = b.member_casual
    AND a.month_number = b.month_number
;



#AVERAGE TRIP DURATION FOR CASUAL USERS AND MEMBERS
SELECT
    member_casual,
      COUNT(ride_id) AS total_trips,
      ROUND(AVG(TIMESTAMPDIFF(MINUTE, started_at, ended_at)), 0) AS average_trip_duration
  FROM
    trip_data
  GROUP BY member_casual
  ORDER BY member_casual;
```

```
#DAYS OF THE WEEK
#MONTHS OF THE YEAR
#RETRIVING THE NUMBER OF TRIPS, AVERAGE LENGTH OF EACH TRIP, AND AVERAGE TRIP DISTANCE(BY CO-ORDINATES WHEN START STATION IS DIFFERENT TO END STATION)
#FOR EACH DAY OF THE WEEK

SELECT
  a.member_casual,
  a.day_number,
  a.day_text,
  a.total_trips,
  a.average_trip_duration,
  b.distance_ridden_km
FROM
  (SELECT
    member_casual,
      weekday(started_at) AS day_number,
      dayNAME(started_at) AS day_text,
      COUNT(ride_id) AS total_trips,
      ROUND(AVG(TIMESTAMPDIFF(MINUTE, started_at, ended_at)), 0) AS average_trip_duration
  FROM
    trip_data
  GROUP BY member_casual , weekday(started_at)
  ORDER BY member_casual , day_number ASC) AS a
    LEFT JOIN
  (SELECT
    member_casual,
      weekday(started_at) AS day_number,
      dayNAME(started_at) AS day_text,
      ROUND(AVG(ST_DISTANCE_SPHERE(POINT(start_lng, start_lat), POINT(end_lng, end_lat))) /
1000, 1) AS distance_ridden_km
  FROM
    trip_data
  WHERE
    ST_DISTANCE_SPHERE(POINT(start_lng, start_lat), POINT(end_lng, end_lat)) > 0
  GROUP BY member_casual , weekday(started_at)
  ORDER BY member_casual , day_number ASC) AS b ON a.member_casual = b.member_casual
    AND a.day_number = b.day_number
;



#RETRIVING THE NUMBER OF TRIPS
#FOR EACH DAY OF THE WEEK AND EACH HOUR OF THE DAY

SELECT
  a.member_casual,
  a.day_number,
  a.day_text,
  a.hour_number,
```

```
  a.total_trips,
  a.average_trip_duration,
  b.distance_ridden_km
FROM
  (SELECT
    member_casual,
      hour(started_at) AS hour_number,
      weekday(started_at) as day_number,
      dayNAME(started_at) AS day_text,
      COUNT(ride_id) AS total_trips,
      ROUND(AVG(TIMESTAMPDIFF(MINUTE, started_at, ended_at)), 0) AS average_trip_duration
  FROM
    trip_data
  GROUP BY member_casual , day_number, hour(started_at)
  ORDER BY member_casual , day_number, hour_number ASC) AS a
    LEFT JOIN
  (SELECT
    member_casual,
      hour(started_at) AS hour_number,
                          weekday(started_at) as day_number,
      ROUND(AVG(ST_DISTANCE_SPHERE(POINT(start_lng, start_lat), POINT(end_lng, end_lat))) /
1000, 1) AS distance_ridden_km
  FROM
    trip_data
  WHERE
    ST_DISTANCE_SPHERE(POINT(start_lng, start_lat), POINT(end_lng, end_lat)) > 0
  GROUP BY member_casual ,day_number, hour(started_at)
  ORDER BY member_casual ,day_number, hour_number ASC) AS b ON a.member_casual =
b.member_casual and a.day_number=b.day_number
    AND a.hour_number = b.hour_number
;



#VEHICLE PREFERENCE,
#SHOWING THE TYPE OF BIKE PER USER TYPE

select a.member_casual, a.rideable_type, round(100*a.total_trips/b.all_trips,2) as percentage
from
(select member_casual, rideable_type, count(rideable_type) as total_trips
from trip_data
group by member_casual, rideable_type
order by member_casual, rideable_type) as a
left join
(select member_casual, count(rideable_type) as all_trips
        from trip_data
        group by member_casual) as b
on a.member_casual=b.member_casual
;
```