First thing will be to upload the csv files into bigquery

Create new project

Create dataset then create tables

Upload the files from local storage or google cloud (create buckets for files larger than 100MB)

After uploading all csv files, merge them using

INSERT INTO – SELECT – FROM query

In my case I used

```
INSERT INTO `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
SELECT *
FROM `vaulted-quarter-375910.Cyclistic.202202-divvy-tripdata`
```

The query above will copy all data from table named 202202-divvy-tripdata to table named 202201-divvy-tripdata.

Do the same for all other tables, make adjustments to the query above by changing the table names in the FROM clause

To confirm the total no. of observations after copying all the tables, use

```
SELECT COUNT (*)
FROM `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
```

The result is 5,667,717

To confirm the blanks in the data use:

```
SELECT *
FROM `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
WHERE ride_id IS NULL
```

The query above checks whether there are any rows missing information in the ride_id column.

Repeat the same for all other columns

833,064 observations were missing data in the start_station_name & start_station_id columns

We shall however proceed because there's start_lat & start_lng which could alternatively be used

892,742 observations were missing data in the end_station_name & end_station_id columns

Out of those 5,858 observations were missing data in the end_lng & end_lat columns

We'll proceed since the station names & id's don't have an impact on our analysis for now.

The rest of the columns had no missing information

However, we shall proceed because that we can still answer the business questions with the available information.

To get clean data (with all stations i.e all 5,667,717 observations) which we'll continue to explore, use the query below

```
SELECT
   ride_id,
   rideable_type,
   member_casual AS membership,
   start_station_name,
   end_station_name,
   EXTRACT(date FROM started_at) AS start_date,
   EXTRACT(time FROM started_at) AS start_time,
   EXTRACT(date FROM ended_at) AS end_date,
   EXTRACT(time FROM ended_at) AS end_time,
   date_diff(ended_at,started_at, MINUTE) AS ride_length,
FROM
   `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
```

I saved the results in a new table as clean-data

Clean data (without nulls)

```
SELECT
   ride_id,
   rideable_type,
   member_casual AS membership,
   start_station_name,
   end_station_name,
   EXTRACT(date FROM started_at) AS start_date,
   EXTRACT(time FROM started_at) AS start_time,
   EXTRACT(date FROM ended_at) AS end_date,
   EXTRACT(time FROM ended_at) AS end_time,
   date_diff(ended_at,started_at, MINUTE) AS ride_length,
FROM
   `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
WHERE
   start_station_name IS NOT NULL AND
   end_station_name IS NOT NULL;
```

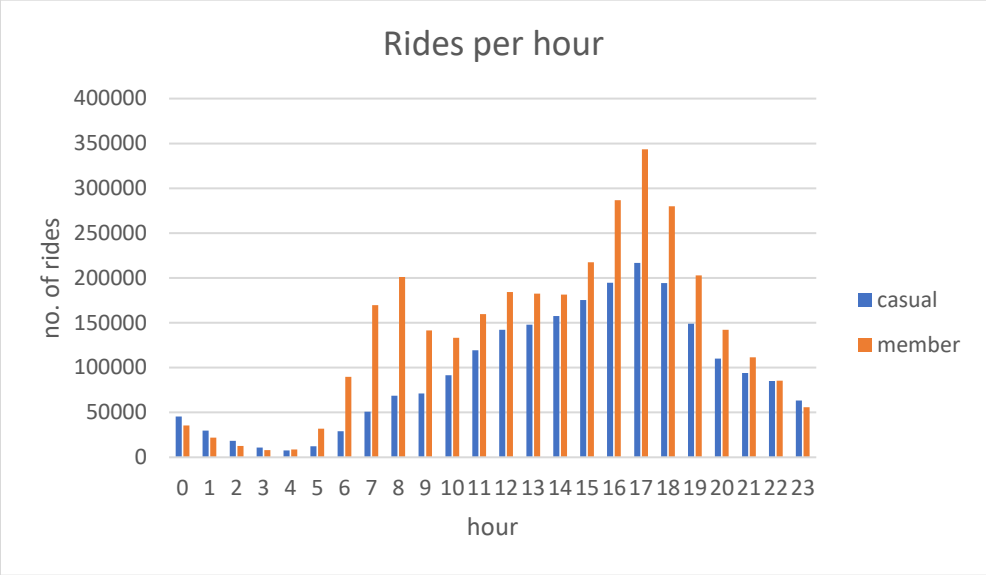
Explore the clean data

```
#Number of rides per start_hour

SELECT
  membership,
  EXTRACT(hour FROM start_time) AS start_hour,
  COUNT(*) AS rides_per_hour
FROM
```

```
  `vaulted-quarter-375910.Cyclistic.clean-data`
GROUP BY
  EXTRACT(hour FROM start_time),membership
```

Export to Excel, create pivot chart and table



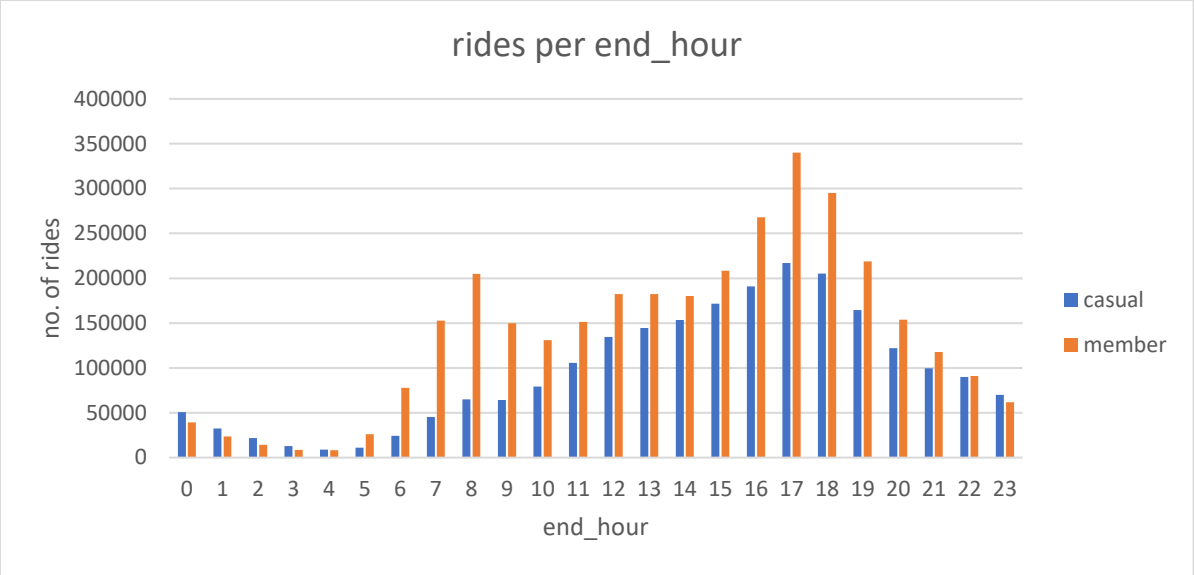
Interpretation: peak hours for annual members are 7-9 am & 4-6 pm. This could mean they mostly use the rides for commuting to work.

As for casual members, the number steadily increases throughout the day and peaks at 3-6 pm which could mean they use the bikes to for short runs e.g to run errands.

```
#Number of rides per end_hour

SELECT
  membership,
  EXTRACT(hour FROM end_time) AS end_hour,
  COUNT(*) AS rides_per_hour
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
GROUP BY
  EXTRACT(hour FROM end_time),membership
```

Results after exporting to Excel & visualizing

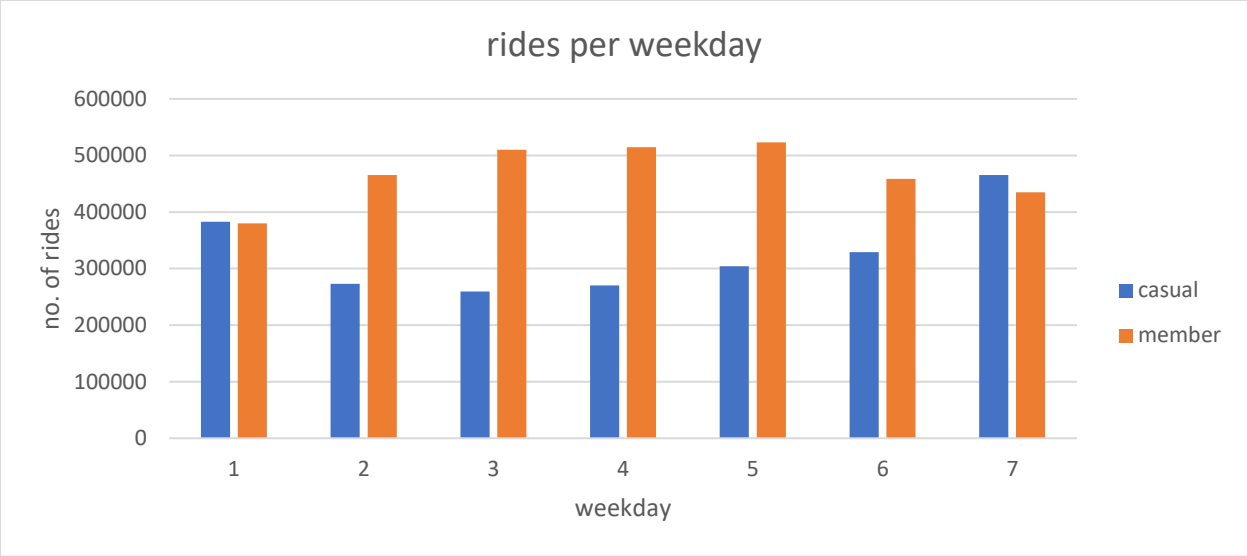


Interpretation: same as start_hour chart

```
#Number of rides according to weekday

SELECT
  EXTRACT(dayofweek FROM start_date) AS start_day,
  membership,
  COUNT(*)
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
GROUP BY
  EXTRACT(dayofweek FROM start_date),membership
```

After exporting and visualization in Excel



Where (in BigQuery) 1=Monday, 2=Tuesday, ..., etc.

Interpretation: the number of rides per week for annual members is almost constant throughtout the week but tends to reduce from day 6 to 1 i.e Saturday to Monday. This means most of them choose Cyclistic bikeshare for their mobility needs to commute to work.

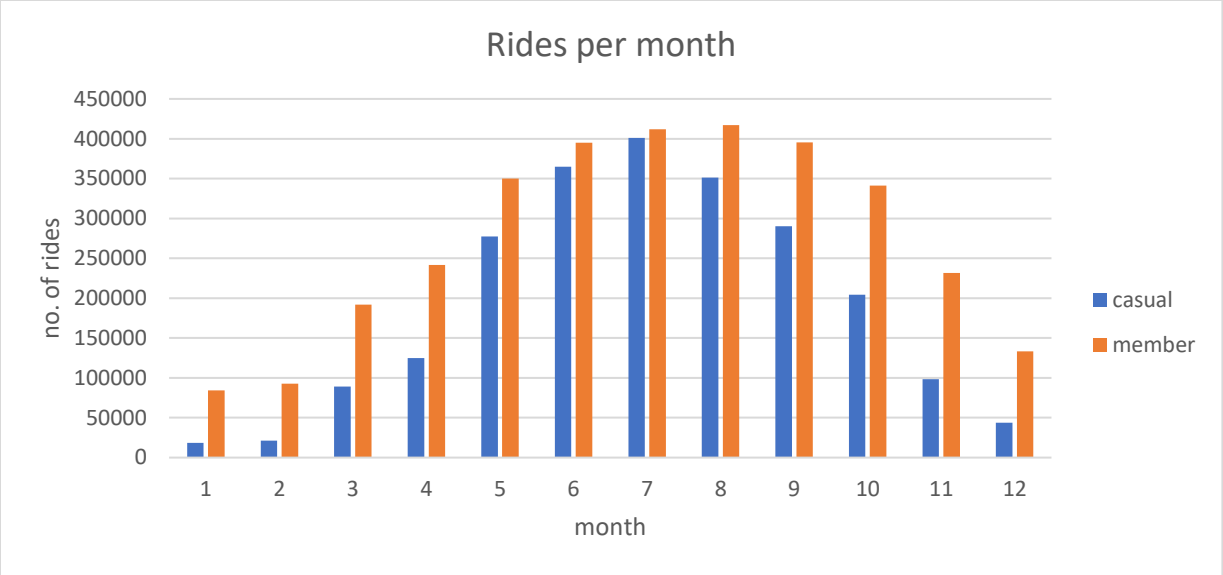
Whereas the number of rides per week for casual riders is a downward slope from the beginning of the week. It however increases on Saturday & Sunday meaning they use the bikes for leisure, to exercise, run errands, etc. i.e those activities which are easier to do during weekends.

Cyclistic could do more marketing towards end of the week to attract more casual riders.

```
#Number of rides according to month


SELECT
  EXTRACT(month FROM start_date) AS start_month,
  membership,
  COUNT(*) as num_of_rides
FROM
  `vaulted-quarter-375910.Cyclistic.double_clean`
GROUP BY
  EXTRACT(month FROM start_date),membership
```

After exporting and visualization in Excel



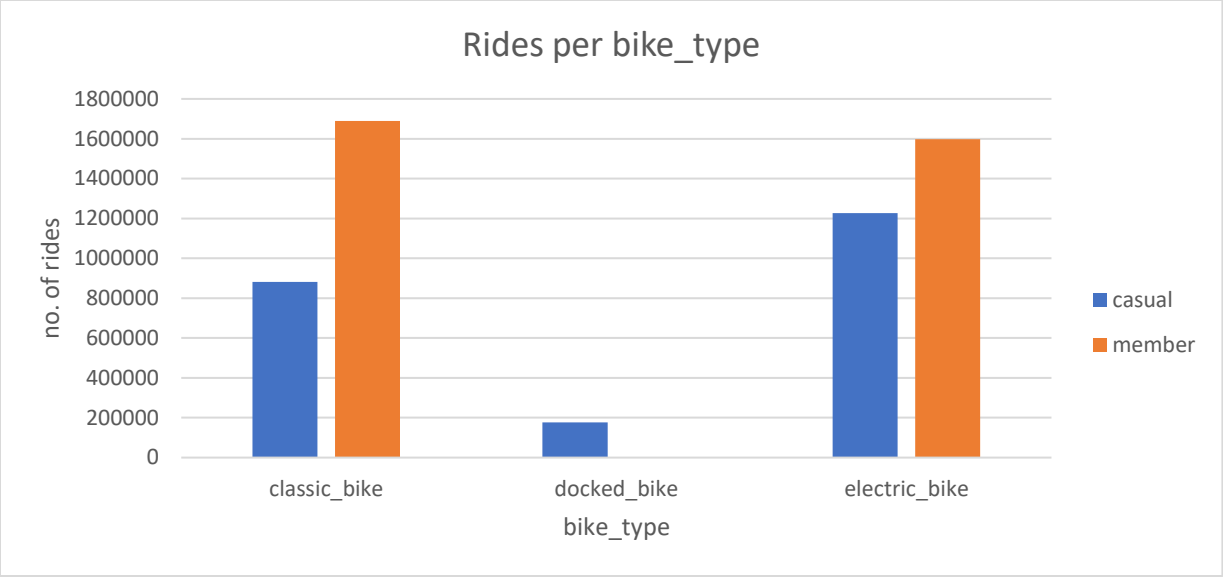
Where 1=Jan, 2=Feb,……, etc.

Interpretation: Weather is a factor in bike usage which explains why it's a bell curve. The normal distribution means that an increase in bike usage is experienced during summer months. Marketing team could take advantage of this period so they can attract casual riders who are tourists. They could start advertising a bit earlier, say in April to attract more local casual riders.

```
#Popularity of bikes according to type

SELECT
  membership,
  rideable_type,
  COUNT(*)
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
GROUP BY
  rideable_type, membership
```

After exporting and visualization in Excel



Interpretation: electric bikes are more preferred by the casual riders. Maybe because of their smooth transition, speed among other factors.

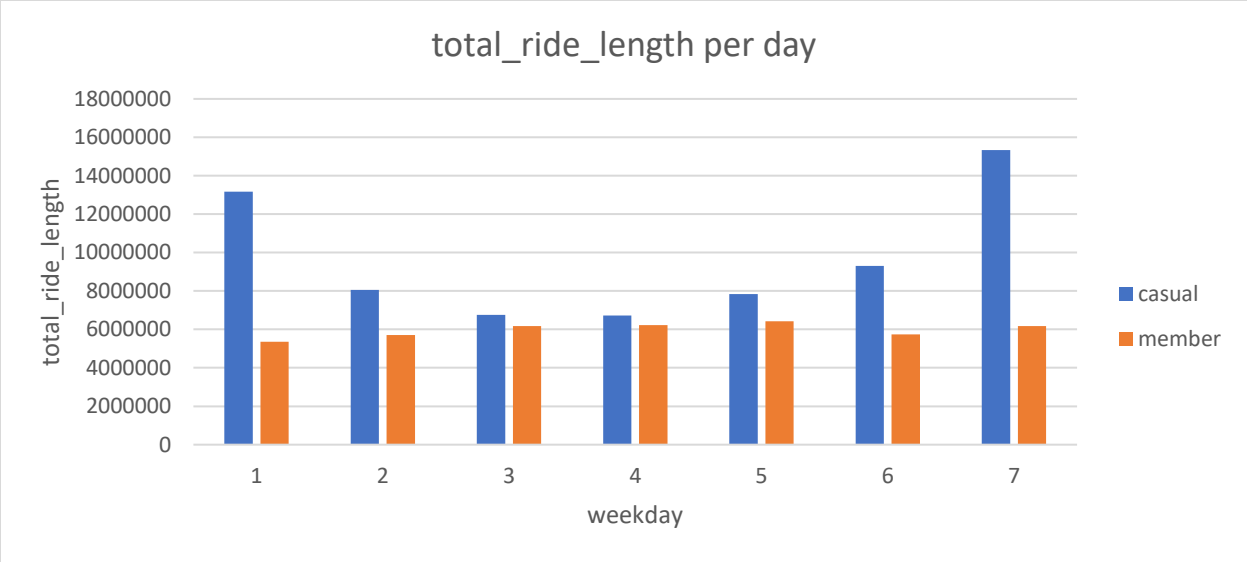
Docked bikes are only used by casual members because of their nature, they need to be locked and secured from unauthorized use. After a casual rider ends his/her ride it is locked at the station.

Preference is almost the same for annual members. I suggest maybe because they mostly use them to commute to work daily, they know the congestion and traffic patterns better and are pretty much confident regardless of bike type.

Cyclistic's marketing team could market the electric bikes more among the casual riders.

```
#Total ride length per day

SELECT
  EXTRACT(dayofweek FROM start_date) AS weekday,
  sum(ride_length) AS total_ride_length,
  membership
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
GROUP BY
  EXTRACT(dayofweek FROM start_date),membership
```

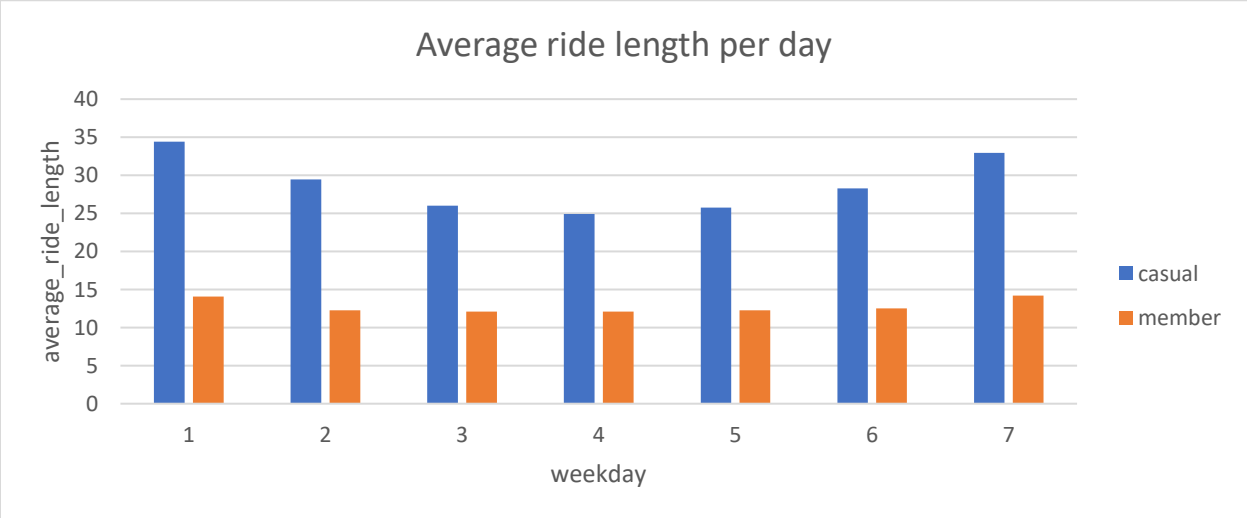


Interpretation: The total of ride_length per day of the week increase towards weekend since there are more casual riders who ride for leisure, errands, etc.

It's consistent for annual members maybe because their schedules are more consistent after commuting to work mostly between day 1 and 5 i.e Monday to Friday

```
#Average ride length per day

SELECT
  EXTRACT(dayofweek FROM start_date) AS weekday,
  avg(ride_length) AS average_ride_length,
  membership
FROM
  `vaulted-quarter-375910.Cyclistic.double_clean`
GROUP BY
  EXTRACT(dayofweek FROM start_date),membership;
```



Interpretation: The total of ride_length per day of the week increase towards weekend since there are more casual riders who ride for leisure, errands, etc. On average, casual riders ride the bikes longer than annual members.

It's consistent for annual members maybe because their schedules are more consistent after commuting to work mostly between day 1 and 5 i.e Monday to Friday

Descriptive statistics

- Maximum ride_length for each membership type
- Minimum ride_length for each membership type
- Average ride_length for each membership type

```
SELECT
  membership,
  max(ride_length) AS max,
  min(ride_length) AS min,
  avg(ride_length) AS average,
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
GROUP BY
  Membership
```

Results were as follows

membership	max	min	average
casual	41387	-138	29
member	1560	-10353	12

membership	max	min	average
member	1560	1	13
casual	41387	1	29

On average, casual riders ride bikes twice longer than annual members

Note, there's no negative time verify the 2 under min using the exact dates and times

```
#A closer look at observations with negative ride_length

      SELECT
        start_date,
        start_time,
        end_date,
        end_time,
```

```
  ride_length
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
WHERE
  ride_length = -138 OR ride_length = -10353;
```

start_date	start_time	end_date	end_time	ride_length
9/28/2022	11:04:32	9/21/2022	6:31:11	-10353
6/7/2022	19:23:00	6/7/2022	17:05:00	-138

A negative ride_length means ride ended a day (days later) at a time earlier than when it was started.

It seems to be an outlier since the next observations' ride_length is 168 mins

```
SELECT
  start_date,
  start_time,
  end_date,
  end_time,
  ride_length
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
WHERE
  ride_length <0
ORDER BY
  ride_length;
```

This showed some 77 observations. It seems that start and end times were reversed

In queries where ride_length is factor, we shall omit the 77.

The clause to be added to be added may look like

```
WHERE
  ride_length >0
```

In real world, this would have been amended in the original data after consultation/confirmation with stakeholders.

It would also be prudent to investigate observations with ride_length = 0. They were 95,957

I opted to clean the data again to omit those with ride_length equal to or less than <=0 after which I did the analysis again.

Rides per time intervals

- No. of rides less than an hour

```
SELECT
```

```
  COUNT(*),
  membership
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
WHERE
  ride_length <60 AND ride_length !=0
GROUP BY
  membership;
```

- No. of rides between 1 – 3 hours

```
SELECT

  COUNT(*),
  membership
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
WHERE
  ride_length BETWEEN 61 AND 180
GROUP BY
  membership;
```

- No. of rides between 3 – 6 hours

```
SELECT
  COUNT(*),
  membership
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
WHERE
  ride_length BETWEEN 181 AND 360
GROUP BY
  membership;
```

- No. of rides between 6 – 12 hours

```
SELECT
  COUNT(*),
  membership
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
WHERE
  ride_length BETWEEN 360 AND 720
GROUP BY
  membership;
```

- No. of rides between 12 – 24 hours

```
SELECT
  COUNT(*),
  membership
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
WHERE
  ride_length BETWEEN 721 AND 1440
GROUP BY
  membership;
```

- No. of rides between 24 – 72 hours

```
SELECT
  COUNT(*),
  membership
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
WHERE
  ride_length BETWEEN 1441 AND 4320
GROUP BY
  membership;
```

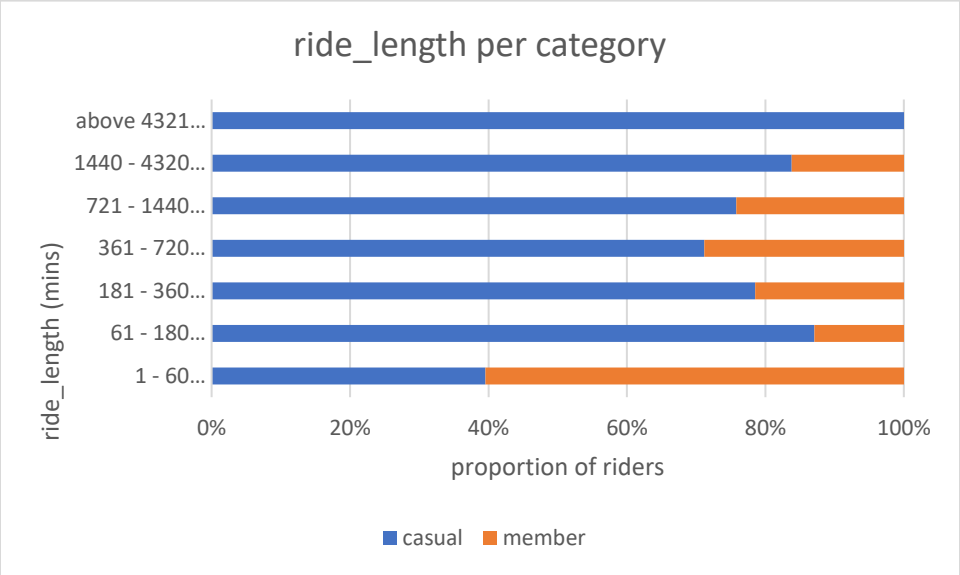
- No. of rides above 72 hours

```
SELECT
  COUNT(*),
  membership
FROM
  `vaulted-quarter-375910.Cyclistic.clean-data`
WHERE
  ride_length >4320
GROUP BY
  membership;
```

This information could be important for pricing flexibility and discounted prices (according to time intervals)

Results were exported to excel, combined and visualized as follows

membership	1 - 60 (0 - 1hr)	61 - 180 (1 - 3hrs)	181 - 360 3 - 6hrs)	361 - 720 6 - 12hrs)	721 - 1440 (12 - 24hrs)	1440 - 4320 (24 - 72hrs)	above 4321 (over 72hrs)
casual	2173992	128415	7425	1851	1556	3708	935
member	3321635	19065	2031	750	497	716	0



Marketing team could consider introduction of a weekly and monthly subscription for favorable pricing options since the data showed there are those who needed the bikes for more than a day. 100% of those who had the bikes for more than 3 days were casual riders.

Count the no. of start_stations

Top 20 stations for each membership type to find out of location is factor

```
#Top 20 start locations

      SELECT
        DISTINCT start_station_name,
        membership,
        count (*) AS count,
      FROM
        `vaulted-quarter-375910.Cyclistic.double_clean`
      WHERE
        start_station_name IS NOT NULL
      GROUP BY
        start_station_name, membership
      ORDER BY
        count DESC
      LIMIT 20;
```

Out of the top 20 start_stations where most rides were started 13/20 had casual riders and 7/20 annual members. 4/5 of the first 5 stations had casual riders.

Location is therefore a factor and marketing may consider targeting the environs near the first 5 start_stations where most casual riders started their rides i.e:

- ✓ Streeter Dr & Grand Ave
- ✓ DuSable Lake Shore Dr & Monroe St
- ✓ Millennium Park
- ✓ Michigan Ave & Oak St

- ✓ Kingsbury St & Kinzie St

With double_clean data

Now we have 5,571,683 observations

Assumptions:

It was assumed that the time intervals were spent riding the bike but it would be more accurate to adjust the tracker to show "rest time" if there's any between the start_time and end_time. Since it's almost unrealistic for one to ride a bike for 41387 mins (i.e 28 days) without stopping

Cyclistic could also collect more data that shows gender so that marketing could be well informed of target market and market segmentation.

Other codes

Separate date and time in the started_at & ended_at columns

Also find time difference using

```
SELECT
  ride_id,
  DATE(ended_at) AS end_date,
  TIME(ended_at) AS end_time,
  DATE(started_at) AS start_date,
  TIME(started_at) AS start_time,
  date_diff(ended_at,started_at, MINUTE) AS ride_length
FROM `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
```

This will be important to verify the differences when subtracting ended_at from started_at

Clean data

```
SELECT
  ride_id,
```

```
  start_station_name,
  end_station_name,
  CAST (started_at as date) AS start_date,
  CAST (ended_at as date) AS end_date,
  date_diff(ended_at,started_at, MINUTE) AS ride_length,
FROM
  `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
WHERE
  start_station_name IS NOT NULL AND
  end_station_name IS NOT NULL
```

Explore the clean data

```
WITH clean_data AS
(
 SELECT
   ride_id,
   member_casual AS membership,
   start_station_name,
   end_station_name,
   CAST (started_at as date) AS start_date,
   CAST (ended_at as date) AS end_date,
   date_diff(ended_at,started_at, MINUTE) AS ride_length,
 FROM
   `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
 WHERE
   start_station_name IS NOT NULL AND
   end_station_name IS NOT NULL
)
SELECT *
FROM clean_data
WHERE ride_length <0
```

59798 rides

The WITH clause gives the subquery an alias name

Save the results as a table or create view

```
SELECT COUNT (*)
FROM `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`

#Merge tables

INSERT INTO `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
SELECT *
```

```
FROM `vaulted-quarter-375910.Cyclistic.202212-divvy-tripdata`

#To get a review what the merged data looks like

SELECT *
FROM `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
WHERE end_station_name IS NULL AND end_lat IS NULL

#Separate date and time in the started_at and ended_at columns

SELECT
  ride_id,
  DATE(ended_at) AS end_date,
  TIME(ended_at) AS end_time,
  DATE(started_at) AS start_date,
  TIME(started_at) AS start_time,
FROM `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`

#Alternatively use the EXRACT() funtion i.e

SELECT
  ride_id,
  EXTRACT(date FROM started_at) AS start_date,
  EXTRACT(time FROM started_at) AS start_time,
  EXTRACT(date FROM ended_at) AS end_date,
  EXTRACT(time FROM started_at) AS end_time
FROM
  `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
```

Or use CONCAT( ) instead of EXTRACT( ) function

```
#Subtract ended_at from started_at to find ride_length

SELECT
  ride_id,
  date_diff(ended_at,started_at, MINUTE) AS ride_length
FROM `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`

#Find out day of the week

SELECT
  ride_id,
  rideable_type,
  EXTRACT(dayofweek FROM started_at) AS start_day,
  EXTRACT(dayofweek FROM ended_at) AS end_day,
  member_casual AS membership
FROM
  `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
```

```
#Clean Data

WITH clean_data AS
(
 SELECT
   ride_id,
   member_casual AS membership,
   start_station_name,
   end_station_name,
   CAST (started_at as date) AS start_date,
   CAST (ended_at as date) AS end_date,
   date_diff(ended_at,started_at, MINUTE) AS ride_length,
 FROM
   `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
 WHERE
   start_station_name IS NOT NULL AND
   end_station_name IS NOT NULL
)
SELECT *
FROM clean_data
WHERE ride_length <0
```

Save results as a table for further exploration

Or create view for your different results e.g

```
CREATE VIEW vaulted-quarter-375910.Cyclistic.bikeride AS
WITH clean_data AS
(
 SELECT
   ride_id,
   member_casual AS membership,
   start_station_name,
   end_station_name,
   CAST (started_at as date) AS start_date,
   CAST (ended_at as date) AS end_date,
   date_diff(ended_at,started_at, MINUTE) AS ride_length,
 FROM
   `vaulted-quarter-375910.Cyclistic.202201-divvy-tripdata`
 WHERE
   start_station_name IS NOT NULL AND
   end_station_name IS NOT NULL
)
SELECT *
FROM clean_data
WHERE ride_length >0
```

I proceeded with the first option since I couldn't preview the data in second option. It had some billing implications (my subscription was sandbox/free with some limited access)