Scenario

I am a data analyst at Cyclistic bike-share company that operates in Chicago. The Cyclistic's management wants to convert casual bike riders into members to generate more revenue. My task is identifying the differences between casual riders and members' riding preferences. This information will be used to start the new marketing company aimed at boosting casual riders to buy annual memberships.

Data sources

The data has been made available by Motivate International Inc under this license. The data has collected by Lyft Bikes and Scooters, LLC that operates the City of Chicago's Divvy bicycle sharing service. For the next analysis, I downloaded data for the past 12 months of trip data (from 2024 June to 2025 May). I saved data using consistent file naming conventions. Additionally, I made a backup of data before I started to process data. Data has been provided as .csv files.

Each file contains data for each ride during the specific month:

1. ride_id (unique for each ride)
2. rideable_type (type of bike used for ride– electric or classic)
3. starting time of each ride
4. ending time of each ride
5. start station id and name
6. end station id and name
7. start station longitude and latitude
8. end station longitude and latitude
9. member_casual (type of user who made a ride – casual user or member).

There are too many blank fields with data that are associated with start and end stations. So, the next analysis conducted without this information.

SQL queries

To create a more suitable table for analysis (that contains only necessary columns) the following query was used. The **EXTRACT() function** with DAYOFWEEK convert the date object and return it as a day of week with Sunday as '1' and Monday as '7'.

```
SELECT
  ride_id,
  rideable_type,
  `started_at`,
  `ended_at`,
  (`ended_at` - `started_at`) AS ride_length,
  EXTRACT(DAYOFWEEK from `started_at`) AS day_of_week,
  member_casual
FROM
  arcane-dolphin-462114-v2.Case_Study.june_2024
```

Then with **MAX function** I calculate the maximum of ride length.

```
SELECT
  MAX(ride_length) AS max_ride_length
FROM
  arcane-dolphin-462114-v2.Case_Study.June
```

By adding the WHERE clause with member_casual = ‘casual’ for casual users and member_casual = ‘member’ for members I identified the maximum of ride length by user type. This element was used in all necessary situations throughout the analysis.

```
SELECT
  MAX(ride_length) AS max_ride_length
FROM
  arcane-dolphin-462114-v2.Case_Study.June
WHERE
  member_casual = 'casual'
```

Similarly, the average ride length was calculated using **AVG()** function.

```
SELECT
  AVG(ride_length) AS avg_ride_length
FROM
  arcane-dolphin-462114-v2.Case_Study.June
WHERE
  member_casual = 'casual'
```

To calculate **the number of rides per day of week** by user type I used the following query. The next query can also work without HAVING clause. But we can use HAVING clause to select all weekdays starting from specific day (for Monday HAVING > 1, for Tuesday HAVING > 2 and so on).

```
SELECT
  day_of_week,
  COUNT(ride_id) AS rides_per_day
FROM
  arcane-dolphin-462114-v2.Case_Study.June
WHERE
  member_casual = 'casual'
GROUP BY
  day_of_week
HAVING
  day_of_week > 0
```

Then I identified the **average ride length by day of week** with the help of the next query.

```
SELECT
  day_of_week,
  AVG(ride_length) AS avg_ride_length_per_day
FROM
  arcane-dolphin-462114-v2.Case_Study.June
WHERE
  member_casual = 'casual'
GROUP BY
  day_of_week
```

To identify **the total number of rides by user type** and **the total ride length by user type** the following queries were used.

Total number of rides by user type:

```
SELECT
  member_casual,
  COUNT(ride_id) AS number_of_rides
FROM
  arcane-dolphin-462114-v2.Case_Study.June
GROUP BY
  member_casual
```

Total ride_length per users:

```
SELECT
  member_casual,
  SUM(ride_length) AS total_ride_length
FROM
  arcane-dolphin-462114-v2.Case_Study.June
GROUP BY
  member_casual
```

Finally, I reviewed the **bike type preferences by user type**. For this purpose, the following query was used:

```
SELECT
  rideable_type,
  COUNT (rideable_type) AS number_of_bikes,
  member_casual
FROM
  arcane-dolphin-462114-v2.Case_Study.July
GROUP BY
  member_casual,
  rideable_type
```

I analysed the data for the certain month to identify trends for the specific month. We can merge data from different months using UNION ALL operator. Then we can conduct the analysis in the scale of the year. The following query merge data for three spring months – March, April and May.

```
SELECT
  ride_id,
  rideable_type,
  `started_at`,
  `ended_at`,
  (`ended_at` - `started_at`) AS ride_length,
  EXTRACT(DAYOFWEEK from `started_at`) AS day_of_week,
  member_casual
FROM
  arcane-dolphin-462114-v2.Case_Study.March
UNION ALL
SELECT
  ride_id,
  rideable_type,
  `started_at`,
  `ended_at`,
  (`ended_at` - `started_at`) AS ride_length,
  EXTRACT(DAYOFWEEK from `started_at`) AS day_of_week,
  member_casual
FROM
  arcane-dolphin-462114-v2.Case_Study.April
UNION ALL
SELECT
  ride_id,
  rideable_type,
  `started_at`,
  `ended_at`,
  (`ended_at` - `started_at`) AS ride_length,
  EXTRACT(DAYOFWEEK from `started_at`) AS day_of_week,
  member_casual
FROM
  arcane-dolphin-462114-v2.Case_Study.May
```

We can continue this merging process by adding another month data with UNION ALL operator. But I collect the most insightful data for each month in the Summary file. Then I used the Summary file to review figures for the year.

I think **if I had more data I could give more recommendations about the organization of the future marketing campaign**. For example, if I was provided by data **rider_id,** that was individual for each Cyclistic user, I can give more insights. In current conditions we can only identify the maximum or the average ride length, the total number of rides and the number of rides per day of the week, the average ride length per day of the week. But with rider_id we can give more individual information. If we have individual for each user rider_id we can calculate the total number of Cyclistic users. Additionally, we can say how many of them are members and casual riders. We can use **COUNT DISTINCT** for this purpose:

SELECT

  member_casual,

  COUNT (DISTINCT rider_id) AS number_of_users

FROM

  Table name

GROUP BY

  member_casual