**Introduction**

This is my first case study as a Data Analyst, and it serves as the capstone project for the Google Data Analytics course. I will be analyzing a fictional company called Cyclistic: a bike sharing program based in Chicago, and the marketing director wants to maximize the number of annual memberships, and we are tasked with determining the behavior of casual riders and members and must present insights supported by professional data visualizations.

Throughout the Gogle Data Analytics program, we follow five phases of data analysis: ask, prepare, process, analyze and share. These will help you work step by step thoughout the analysis.

**ASK**

The task is to analyze the behavior of casual riders and members for the team to think of a marketing strategy to promote annual membership to casual riders.

**How do annual members and casual riders use Cyclistic bikes differently?**

**PREPARE**

I will use Cyclistic’s trip data for the past 12 months (November 2021 to October 2022) to analyze and identify trends. Datasets are available in this [link](https://divvy-tripdata.s3.amazonaws.com/index.html).

ROCCC Analysis

Reliable - The data has been made available by Motivate International Inc. under this [license](https://ride.divvybikes.com/data-license-agreement).

Original - Lyft Bikes and Scooters, LLC, operates the City of Chicago’s, subject to the terms and conditions of the Data License Agreement.

Comprehensive - The data is organized in .cvs files by trip month.

Current - This data is being updated monthly.

Cited – It is a public data and the datasets are available in this [link](https://divvy-tripdata.s3.amazonaws.com/index.html).

A lot of students who finished the same case study used Rstudio. Just to be unique with the others, I used pgAdmin4 - leading Open Source management tool for PostgresSQL, the world's most advanced Open Source database.

To import the CSV files in pgAdmin4, first you need to create tables and then import the twelve CSV files one by one. Here’s the code in creating tables, 12 of these should be written, you just need to change the table name after the CREATE TABLE function. As I have checked, all files have the same column name, therefore we can proceed in creating tables and merging them later on.

CREATE TABLE nov21 (

ride\_id varchar,

rideable\_type varchar,

started\_at timestamp,

ended\_at timestamp,

start\_station\_name varchar,

start\_station\_id varchar,

end\_station\_name varchar,

end\_station\_id varchar,

start\_lat numeric,

start\_lng numeric,

end\_lat numeric,

end\_lng numeric,

member\_casual varchar

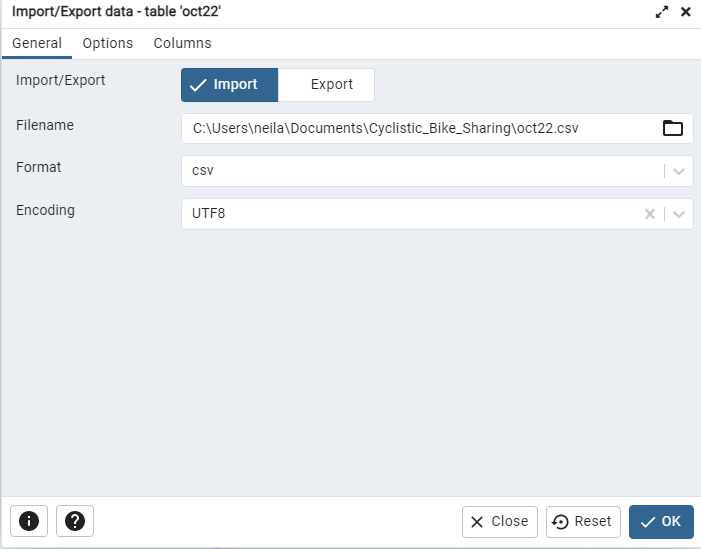
);

After executing the code, just right click the *Tables* on the side panel and hit Refresh. All the created tables should appear.

Table

Description automatically generated

To import the downloaded CSV files, right click on a table and then click *Import/Export Data*. Click on the folder icon and locate the CSV file that you want to import. The format should be CSV and the encoding is UTF8.



After importing all the files, we need to merge all tables into one dataset. We can do that by using UNION. The UNION operator is used to combine the result-set of two or more SELECT statements.

CREATE TABLE cyclistic\_data

AS

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

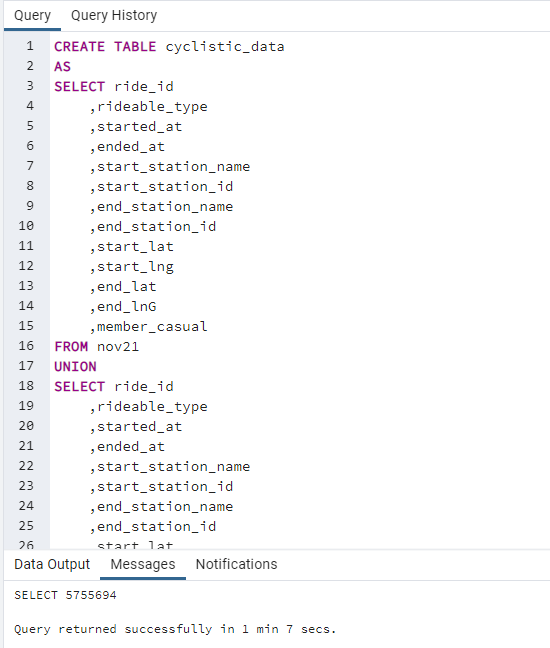
,end\_lng

,member\_casual

FROM nov21

UNION

SELECT *… (and so on)*

Running the code may take some time because we are combining twelve CSV files with thousands of rows; the speed is also dependent on your CPU. In my case, the query took 1 minute and 7 seconds to complete with a total of 5,755,694 rows.

**CLEAN DATA**

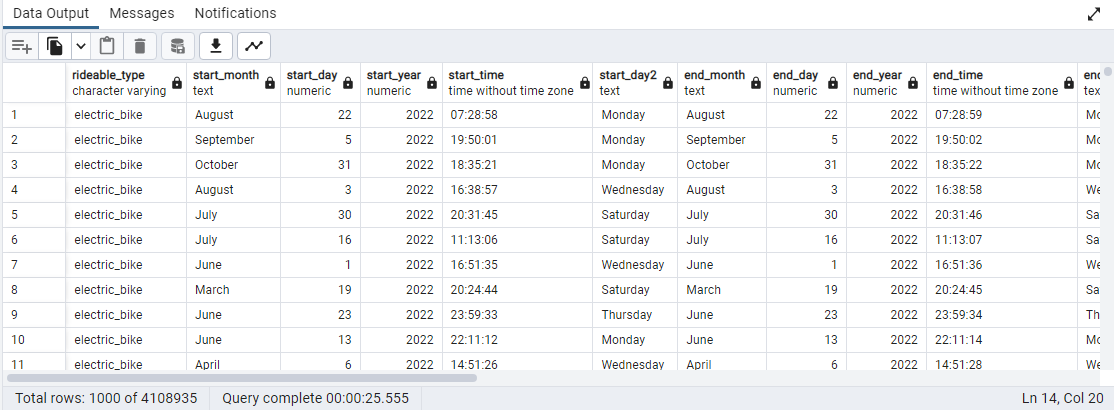
I have observed in the dataset that there are negative values for the trip duration and 0 values. Also, some trips have same locations for the station and coordinates (latitude and longitude). I came up with the condition:

WHERE (start\_lat != end\_lat AND start\_lng != end\_lng)

AND start\_station\_name!=end\_station\_name

AND (ended\_at - started\_at)>'00:00:00'

The first line explains that the starting latitude and ending latitude should not be the same, as should the longitudes. The second line explains that the initial station and the final station should not be the same. The final line indicates that the trip duration should be greater than 0.



After the execution of the given statements, we are now left with 4,108,935 valid trips. This may be further corrected by the client if ever some conditions have been overlooked.

**ANALYZE**

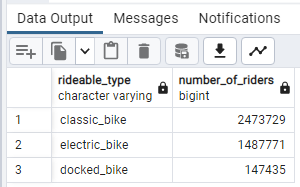
Cyclistic is offering three kinds of bikes: classic, docked and electric bikes. First, I want to know which bike is mostly used by people.

SELECT rideable\_type, COUNT(\*) AS number\_of\_riders

FROM cyclistic\_cleaned

GROUP BY rideable\_type

ORDER BY COUNT(\*) DESC



There are **2,473,729** people using the classic bike, while **147,435** are using the docked bike and **1,487,771** are using the electric bike.

Next, let’s determine how many are casual riders and how many members does Cyclistic have.

SELECT member\_casual, COUNT(\*) AS total

FROM cyclistic\_cleaned

GROUP BY member\_casual

ORDER BY COUNT(\*) DESC

Graphical user interface, text, application, table, Excel

Description automatically generated

Total members are **2,521,356** while casual riders are **1,587,579**.

Let’s determine how many casual riders and members use different types of bikes.

SELECT rideable\_type

, COUNT(rideable\_type) AS number\_of\_rides

, member\_casual

FROM cyclistic\_cleaned

GROUP BY rideable\_type, member\_casual

Table

Description automatically generated

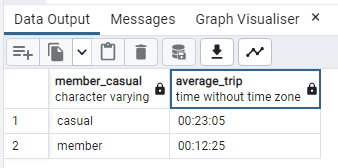
Let’s get the average trip duration of members and casual riders.

SELECT member\_casual

, AVG(trip\_duration)::time(0) AS average\_trip

FROM cyclistic\_cleaned

GROUP BY member\_casual



Average trip duration of casual riders is **23 minutes 5 seconds** while the average trip duration of members is **12 minutes 25 seconds**.

Let’s also get the overall average trip duration.

SELECT AVG(trip\_duration)::time(0) AS average\_trip

FROM cyclistic\_cleaned

Graphical user interface, text, application

Description automatically generated

The average trip of all users is **16 minutes 33 seconds**.

Let’s get the average trip durations per month

SELECT CONCAT(start\_month, ' ',start\_year) AS month\_year

,AVG(trip\_duration)::time(0) AS average\_per\_month

FROM cyclistic\_cleaned

GROUP BY month\_year

ORDER BY average\_per\_month DESC

Table

Description automatically generated

I also want to know the average trip duration per month with types of riders.

SELECT CONCAT(start\_month, ' ',start\_year) AS month\_year

,AVG(trip\_duration)::time(0) AS average\_per\_month

, member\_casual

FROM cyclistic\_cleaned

GROUP BY month\_year, member\_casual

ORDER BY average\_per\_month DESC Table

Description automatically generated

Let’s get the total number of rides per month.

SELECT CONCAT(start\_month, ' ',start\_year) AS month\_year

,COUNT(trip\_duration) AS count\_per\_month

FROM cyclistic\_cleaned

GROUP BY month\_year

ORDER BY COUNT(trip\_duration) DESC

Table

Description automatically generated

Let’s get the total number of rides per month with types of riders.

SELECT CONCAT(start\_month, ' ',start\_year) AS month\_year

,COUNT(\*) AS count\_per\_month

, member\_casual

FROM cyclistic\_cleaned

GROUP BY month\_year, member\_casual

ORDER BY count\_per\_month DESC

Table

Description automatically generated with medium confidence

Let’s get the average trip duration per day.

SELECT start\_day2 AS day\_of\_month

,AVG(trip\_duration)::time(0) AS average\_trip

FROM cyclistic\_cleaned

GROUP BY start\_day2

ORDER BY AVG(trip\_duration) DESC

Table

Description automatically generated

Let’s get the total number of rides per day

SELECT start\_day2 AS day\_of\_month

,COUNT(trip\_duration) AS total\_trip

FROM cyclistic\_cleaned

GROUP BY start\_day2

ORDER BY COUNT(trip\_duration) DESC

Table

Description automatically generated

Total trip per day of members.

SELECT start\_day2 AS day\_of\_week

,COUNT(trip\_duration) AS total\_trip

FROM cyclistic\_cleaned

WHERE member\_casual = ‘member’

GROUP BY start\_day2

ORDER BY COUNT(trip\_duration) DESC

Table

Description automatically generated

Average trip per day of members

SELECT start\_day2 AS day\_of\_week

,AVG(trip\_duration)::time(0) AS total\_trip

FROM cyclistic\_cleaned

WHERE member\_casual = 'member'

GROUP BY start\_day2

ORDER BY AVG(trip\_duration) DESC

Table

Description automatically generated

Total trip per day of casual riders.

SELECT start\_day2 AS day\_of\_month

,COUNT(trip\_duration) AS total\_trip

FROM cyclistic\_cleaned

WHERE member\_casual = 'casual'

GROUP BY start\_day2

ORDER BY COUNT(trip\_duration) DESC

Table

Description automatically generated

Average trip per day of casual riders

SELECT start\_day2 AS day\_of\_week

,AVG(trip\_duration)::time(0) AS total\_trip

FROM cyclistic\_cleaned

WHERE member\_casual = 'casual'

GROUP BY start\_day2

ORDER BY AVG(trip\_duration) DESC

Table

Description automatically generated

Let’s see how many riders per time of the day

SELECT start\_day2 AS day\_of\_month, COUNT(start\_day) AS total\_trip, (CASE

WHEN start\_time >= '06:00:00' AND start\_time < '12:00:00' THEN 'Morning'

WHEN start\_time >= '12:00:00' AND start\_time < '17:00:00' THEN 'Afternoon'

WHEN start\_time >= '17:00:00' AND start\_time < '20:00:00' THEN 'Evening'

ELSE 'Night'

END ) AS time\_day

FROM cyclistic\_cleaned

GROUP BY start\_day2, (CASE

WHEN start\_time >= '06:00:00' AND start\_time < '12:00:00' THEN 'Morning'

WHEN start\_time >= '12:00:00' AND start\_time < '17:00:00' THEN 'Afternoon'

WHEN start\_time >= '17:00:00' AND start\_time < '20:00:00' THEN 'Evening'

ELSE 'Night'

END)

ORDER BY total\_trip DESC

Table

Description automatically generated

I also want to know the top 20 stations with the highest number of trips.

SELECT start\_station\_name AS station

, COUNT(start\_station\_name) AS total\_trips

FROM cyclistic\_cleaned

GROUP BY start\_station\_name

ORDER BY total\_trips DESC

LIMIT 20

Table

Description automatically generated

I also want to get the longest trip of members and casual riders by bike type.

SELECT rideable\_type

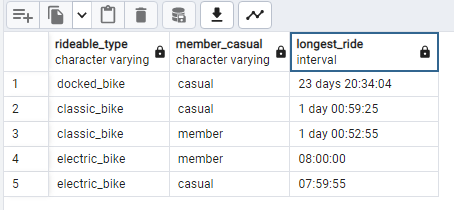
, member\_casual

, MAX(trip\_duration) AS longest\_ride

FROM cyclistic\_cleaned

GROUP BY rideable\_type, member\_casual

ORDER BY longest\_ride DESC



Lastly, I created a table with the latitudes and longitudes so that I can import it in tableau and create visualizations.

CREATE TABLE location AS

SELECT rideable\_type

, member\_casual

, start\_lat

, start\_lng

, end\_lat

, end\_lng

FROM cyclistic\_cleaned