Google Data Analytics Project: Cyclistic

Tulio de la Cruz

2024-02-26

Content

Introduction

As part of the first case study in the final project of the Google Data Analytics Certificate, the task involves addressing a hypothetical business challenge for a fictitious bicycle company. The steps of the comprehensive data analysis process: ask, prepare, process, analyze, share, and act, will be followed with the objective of demonstrating the skills and knowledge acquired to effectively fulfill the role of a data analyst.

Scenario

Cyclistic is a company that launched a successful bike-sharing program and has grown to operate a fleet of 5,824 georeferenced bicycles locked in a network of 692 stations across Chicago. Bikes can be unlocked from one station and returned to any other station in the system at any time.

So far, Cyclistic's marketing strategy has been based on building general brand awareness and attracting broad consumer segments. One of the approaches that helped make this possible was the flexibility of its pricing plans: single-ride passes, full-day passes, and annual memberships. Customers who purchase single-ride passes or full-day passes are referred to as casual riders. Customers who purchase annual memberships are called Cyclistic members.

Cyclistic's financial analysts have concluded that annual members are much more profitable than casual riders. Although the pricing flexibility helps Cyclistic attract more customers. Lily Moreno, the marketing director and manager, believes that maximizing the number of annual members will be key to future growth. Instead of creating a marketing campaign targeting all new customers, Moreno believes there is a great opportunity to convert casual riders into members. She points out that casual riders are already familiar with Cyclistic's program and have chosen Cyclistic for their mobility needs.

Moreno also set a clear goal: to design marketing strategies aimed at converting casual riders into annual members. However, to do that, the marketing analytics team needs to better understand how annual members and casual riders differ, why casual riders would consider purchasing a membership, and how digital media might impact their marketing tactics. Moreno and her team are interested in analyzing Cyclistic's historical bike trip data to identify trends.

Three questions will guide the future marketing program:

1. How do annual members and casual riders differ in their use of Cyclistic bikes?
2. Why would casual riders purchase annual memberships from Cyclistic?
3. How can Cyclistic utilize digital media to influence casual riders to become members?

The main actors and stakeholders are made up of:

- Cyclistic: A bike-sharing program that includes around 5,800 bicycles and 600 stations. It stands out for also offering various types of bikes.
- Lily Moreno: The marketing director and manager. Moreno is responsible for developing campaigns and initiatives to promote the bike-sharing program. The campaigns may include email, social media, and other channels.
- Cyclistic's Marketing Data Analytics Team: A team of data analysts responsible for collecting, analyzing, and reporting data that helps drive Cyclistic's marketing strategy.
- Cyclistic's Executive Team: This highly detail-oriented group will decide whether to approve the recommended marketing program.

Description

Cyclistic is a Chicago-based bike-sharing company whose marketing director believes that the future success of the company depends on maximizing the number of annual memberships. Therefore, the marketing analytics team aims to understand the differences in Cyclistic bike usage between casual riders and annual members. With this knowledge, the marketing data analytics team will design a new marketing strategy to convert casual riders into annual members.

Objective

Design a marketing program for Lily Moreno and the Cyclistic executive team, with strategies aimed at converting casual riders into annual members.

Research Question

To design a marketing program with certain characteristics, it is essential to know, among other things, the following: How do annual members and casual riders differ in their use of Cyclistic bikes?

Methodology

Data Preparation

For the practical purposes of the project, the course guides the use of historical data provided by Motivate International Inc, stored in the Index of bucket “divvy-tripdata” under the Data License Agreement | Divvy Bikes. This means that Motivate International Inc is considered a reliable data source in the field of bike sharing, and the associated license provides a clear legal framework for data usage. These data are obtained directly from the original source, in this case, Motivate International Inc. This ensures their originality as they have not been altered or misinterpreted by secondary sources.

Looking at the data at first glance, it is clear that they have the critical and necessary information to perform the appropriate analysis. This data set can be considered current as it contains files corresponding to all of 2023, so it is possible to detect relevant patterns and trends up to date of writing this project.

Description of Source Data

Twelve files corresponding to each month of 2023 will be used. These data are in tabular format, vertical, and in CSV format contained within ZIP files. It is important to emphasize that apart from being used within the permitted limits of the license, the data do not expose any sensitive information of any third party.

Data Processing

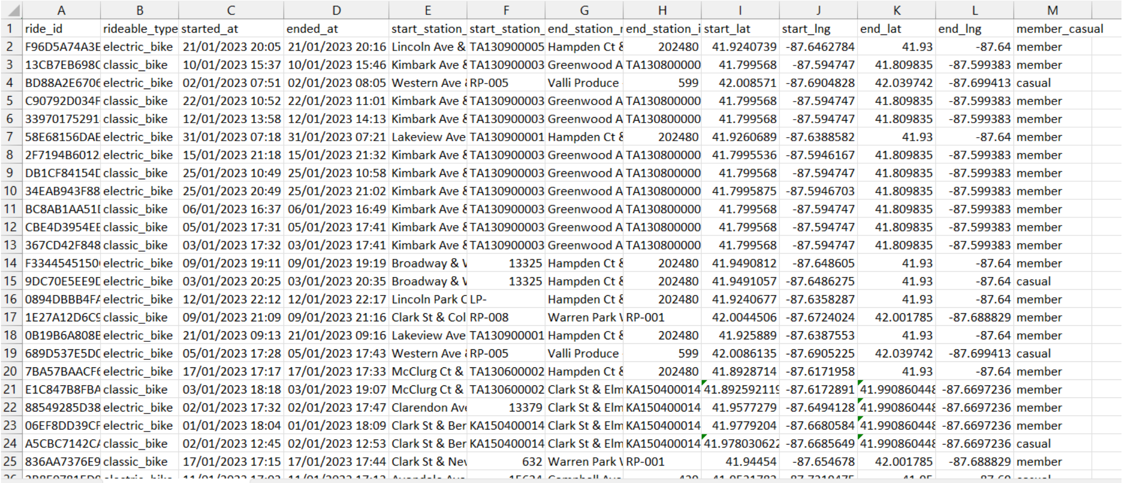
SQL is chosen for data processing, specifically SQL Server, as the data size when combined is around 5 million rows, exceeding the approximate one million row limit that Excel can handle. While it is possible to view the number of rows of each file individually with Excel, SQL is the appropriate tool for carrying out the transformation and analysis of the data.

As mentioned earlier, twelve files corresponding exclusively to the year 2023 are used, with each file representing a month of captured data. The twelve selected files are:

- 202301-divvy-tripdata
- 202302-divvy-tripdata
- 202303-divvy-tripdata
- 202304-divvy-tripdata
- 202305-divvy-tripdata
- 202306-divvy-tripdata
- 202307-divvy-tripdata
- 202308-divvy-tripdata
- 202309-divvy-tripdata
- 202310-divvy-tripdata

- 202311-divvy-tripdata
- 202312-divvy-tripdata

As a superficial observation in Excel, it appears that the overall structure is consistent across all files. This means that the number, name, and data type of columns seem to be the same across all files, except for the number of rows.

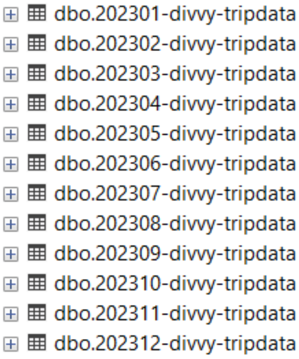


	A	B	C	D	E	F	G	H	I	J	K	L	M
1	ride_id	rideable_type	started_at	ended_at	start_station_	start_station_	end_station_r	end_station_i	start_lat	start_lng	end_lat	end_lng	member_casual
2	F96D5A74A3E	electric_bike	21/01/2023 20:05	21/01/2023 20:16	Lincoln Ave &	TA130900005	Hampden Ct &	202480	41.9240739	-87.6462784	41.93	-87.64	member
3	13CB7EB698C	classic_bike	10/01/2023 15:37	10/01/2023 15:46	Kimbark Ave &	TA130900003	Greenwood A	TA130800000	41.799568	-87.594747	41.809835	-87.599383	member
4	BD88A2E6706	electric_bike	02/01/2023 07:51	02/01/2023 08:05	Western Ave &	RP-005	Valli Produce	599	42.008571	-87.6904828	42.039742	-87.699413	casual
5	C90792D034F	classic_bike	22/01/2023 10:52	22/01/2023 11:01	Kimbark Ave &	TA130900003	Greenwood A	TA130800000	41.799568	-87.594747	41.809835	-87.599383	member
6	33970175291	classic_bike	12/01/2023 13:58	12/01/2023 14:13	Kimbark Ave &	TA130900003	Greenwood A	TA130800000	41.799568	-87.594747	41.809835	-87.599383	member
7	58E68156DAE	electric_bike	31/01/2023 07:18	31/01/2023 07:21	Lakeview Ave	TA130900001	Hampden Ct &	202480	41.9260689	-87.6388582	41.93	-87.64	member
8	2F7194B6012	electric_bike	15/01/2023 21:18	15/01/2023 21:32	Kimbark Ave &	TA130900003	Greenwood A	TA130800000	41.7995536	-87.5946167	41.809835	-87.599383	member
9	DB1CF84154D	classic_bike	25/01/2023 10:49	25/01/2023 10:58	Kimbark Ave &	TA130900003	Greenwood A	TA130800000	41.799568	-87.594747	41.809835	-87.599383	member
10	34EAB943F88	electric_bike	25/01/2023 20:49	25/01/2023 21:02	Kimbark Ave &	TA130900003	Greenwood A	TA130800000	41.7995875	-87.5946703	41.809835	-87.599383	member
11	BC8AB1AA51I	classic_bike	06/01/2023 16:37	06/01/2023 16:49	Kimbark Ave &	TA130900003	Greenwood A	TA130800000	41.799568	-87.594747	41.809835	-87.599383	member
12	CBE4D3954EE	classic_bike	05/01/2023 17:31	05/01/2023 17:41	Kimbark Ave &	TA130900003	Greenwood A	TA130800000	41.799568	-87.594747	41.809835	-87.599383	member
13	367CD42F848	classic_bike	03/01/2023 17:32	03/01/2023 17:41	Kimbark Ave &	TA130900003	Greenwood A	TA130800000	41.799568	-87.594747	41.809835	-87.599383	member
14	F3344545150	electric_bike	09/01/2023 19:11	09/01/2023 19:19	Broadway & \	13325	Hampden Ct &	202480	41.9490812	-87.648605	41.93	-87.64	member
15	9DC70E5EE9D	electric_bike	03/01/2023 20:25	03/01/2023 20:35	Broadway & \	13325	Hampden Ct &	202480	41.9491057	-87.6486275	41.93	-87.64	casual
16	0894DBBB4FA	electric_bike	12/01/2023 22:12	12/01/2023 22:17	Lincoln Park C	LP-	Hampden Ct &	202480	41.9240677	-87.6358287	41.93	-87.64	member
17	1E27A12D6C9	classic_bike	09/01/2023 21:09	09/01/2023 21:16	Clark St & Col	RP-008	Warren Park \	RP-001	42.0044506	-87.6724024	42.001785	-87.688829	member
18	0B19B6A808E	electric_bike	21/01/2023 09:13	21/01/2023 09:16	Lakeview Ave	TA130900001	Hampden Ct &	202480	41.925889	-87.6387553	41.93	-87.64	member
19	689D537E5DC	electric_bike	05/01/2023 17:28	05/01/2023 17:43	Western Ave &	RP-005	Valli Produce	599	42.0086135	-87.6905225	42.039742	-87.699413	casual
20	7BA57BAACF6	electric_bike	17/01/2023 17:17	17/01/2023 17:33	McClurg Ct &	TA130600002	Hampden Ct &	202480	41.8928714	-87.6171958	41.93	-87.64	member
21	E1C847B8FBA	classic_bike	03/01/2023 18:18	03/01/2023 19:07	McClurg Ct &	TA130600002	Clark St & Elm	KA150400014	41.89259211	-87.6172891	41.99086044	-87.6697236	member
22	88549285D38	electric_bike	02/01/2023 17:32	02/01/2023 17:47	Clarendon Ave	13379	Clark St & Elm	KA150400014	41.9577279	-87.6494128	41.99086044	-87.6697236	member
23	06EF8DD39CF	electric_bike	01/01/2023 18:04	01/01/2023 18:09	Clark St & Ber	KA150400014	Clark St & Elm	KA150400014	41.9779204	-87.6680584	41.99086044	-87.6697236	member
24	A5CBC7142CA	classic_bike	02/01/2023 12:45	02/01/2023 12:53	Clark St & Ber	KA150400014	Clark St & Elm	KA150400014	41.97803062	-87.6685649	41.99086044	-87.6697236	casual
25	836AA7376E9	classic_bike	17/01/2023 17:15	17/01/2023 17:44	Clark St & Nev	632	Warren Park \	RP-001	41.94454	-87.654678	42.001785	-87.688829	member

File 202301-divvy-tripdata

Data Importation

Once the tool to be used is clear, the files to be analyzed are determined, and there is a general idea of how the data looks, they are imported into SQL Server, taking care of the data type of each column. As confirmation of a successful import, the tables will be visible in the SQL Server Object Explorer panel.



```
⊞ dbo.202301-divvy-tripdata
⊞ dbo.202302-divvy-tripdata
⊞ dbo.202303-divvy-tripdata
⊞ dbo.202304-divvy-tripdata
⊞ dbo.202305-divvy-tripdata
⊞ dbo.202306-divvy-tripdata
⊞ dbo.202307-divvy-tripdata
⊞ dbo.202308-divvy-tripdata
⊞ dbo.202309-divvy-tripdata
⊞ dbo.202310-divvy-tripdata
⊞ dbo.202311-divvy-tripdata
⊞ dbo.202312-divvy-tripdata
```

Data imported into SQL Server

As a next step, the files are reviewed again one by one through a general query to corroborate any inconsistency or error in the import process. The following query shows, as a first review, all the columns and all the rows of the table 202301-divvy-tripdata.

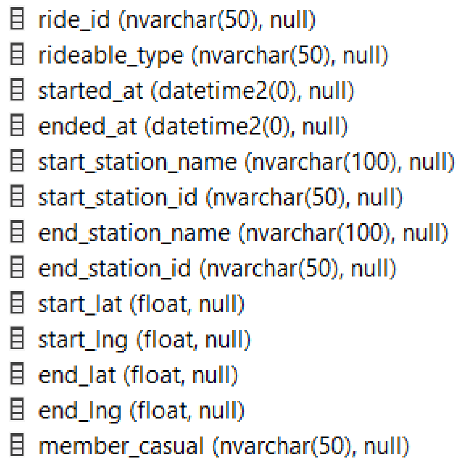
```
SELECT
*
FROM
"202301-divvy-tripdata"
```

Modification of Data Types and Data Type Formats

The corresponding data type for each column of each file is definitively determined, and the data type can be changed through queries or graphically in the SQL Server interface. Next, it is decided to edit the data type format of the "started_at" and "ended_at" columns to improve readability in each of the tables. The following queries remove the 7 decimal digits from the milliseconds that are generated by default in the DATETIME2 data type of the table 202301-divvy-tripdata.

```
ALTER TABLE
"202301-divvy-tripdata"
ALTER COLUMN
started_at DATETIME2(0)

ALTER TABLE
"202301-divvy-tripdata"
ALTER COLUMN
ended_at DATETIME2(0)
```



ride_id (nvarchar(50), null)
rideable_type (nvarchar(50), null)
started_at (datetime2(0), null)
ended_at (datetime2(0), null)
start_station_name (nvarchar(100), null)
start_station_id (nvarchar(50), null)
end_station_name (nvarchar(100), null)
end_station_id (nvarchar(50), null)
start_lat (float, null)
start_lng (float, null)
end_lat (float, null)
end_lng (float, null)
member_casual (nvarchar(50), null)

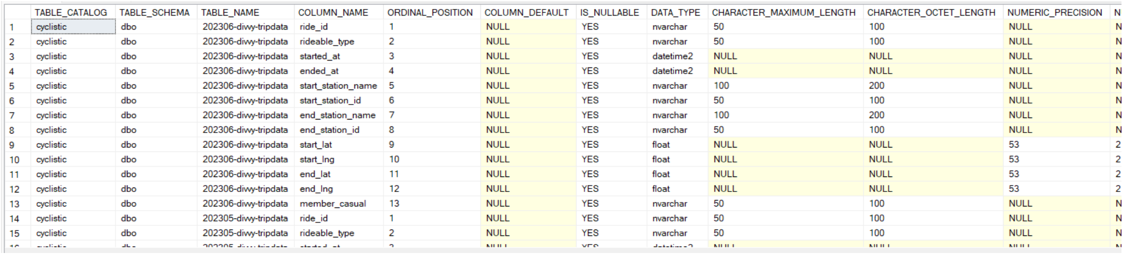
Data type used in each table

Review of rows and columns

Once the data types and their formats are standardized, the column names that are the same in each table are verified to facilitate any necessary processing or analysis.

The following query provides general information about the imported tables.

```
SELECT
*
FROM
INFORMATION_SCHEMA.COLUMNS
```

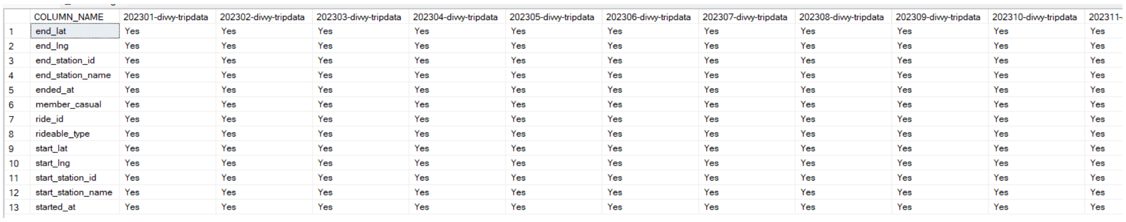


	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	CHARACTER_OCTET_LENGTH	NUMERIC_PRECISION	N
1	cyclistic	dbo	202306-divvy-tripdata	ride_id	1	NULL	YES	nvarchar	50	100	NULL	N
2	cyclistic	dbo	202306-divvy-tripdata	rideable_type	2	NULL	YES	nvarchar	50	100	NULL	N
3	cyclistic	dbo	202306-divvy-tripdata	started_at	3	NULL	YES	datetime2	NULL	NULL	NULL	N
4	cyclistic	dbo	202306-divvy-tripdata	ended_at	4	NULL	YES	datetime2	NULL	NULL	NULL	N
5	cyclistic	dbo	202306-divvy-tripdata	start_station_name	5	NULL	YES	nvarchar	100	200	NULL	N
6	cyclistic	dbo	202306-divvy-tripdata	start_station_id	6	NULL	YES	nvarchar	50	100	NULL	N
7	cyclistic	dbo	202306-divvy-tripdata	end_station_name	7	NULL	YES	nvarchar	100	200	NULL	N
8	cyclistic	dbo	202306-divvy-tripdata	end_station_id	8	NULL	YES	nvarchar	50	100	NULL	N
9	cyclistic	dbo	202306-divvy-tripdata	start_lat	9	NULL	YES	float	NULL	NULL	53	2
10	cyclistic	dbo	202306-divvy-tripdata	start_lng	10	NULL	YES	float	NULL	NULL	53	2
11	cyclistic	dbo	202306-divvy-tripdata	end_lat	11	NULL	YES	float	NULL	NULL	53	2
12	cyclistic	dbo	202306-divvy-tripdata	end_lng	12	NULL	YES	float	NULL	NULL	53	2
13	cyclistic	dbo	202306-divvy-tripdata	member_casual	13	NULL	YES	nvarchar	50	100	NULL	N
14	cyclistic	dbo	202305-divvy-tripdata	ride_id	1	NULL	YES	nvarchar	50	100	NULL	N
15	cyclistic	dbo	202305-divvy-tripdata	rideable_type	2	NULL	YES	nvarchar	50	100	NULL	N

Data displayed by INFORMATION_SCHEMA.COLUMNS

Thanks to the information provided by INFORMATION_SCHEMA.COLUMNS, it is possible to generate a query that helps compare column names in each table and determine which tables share column names.

```
SELECT
COLUMN_NAME,
MAX(CASE
    WHEN TABLE_NAME = '202301-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202301-divvy-tripdata',
MAX(CASE
    WHEN TABLE_NAME = '202302-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202302-divvy-tripdata',
MAX(CASE
    WHEN TABLE_NAME = '202303-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202303-divvy-tripdata',
MAX(CASE
    WHEN
    TABLE_NAME = '202304-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202304-divvy-tripdata',
MAX(CASE
    WHEN TABLE_NAME = '202305-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202305-divvy-tripdata',
MAX(CASE
    WHEN TABLE_NAME = '202306-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202306-divvy-tripdata',
MAX(CASE
    WHEN TABLE_NAME = '202307-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202307-divvy-tripdata',
MAX(CASE
    WHEN TABLE_NAME = '202308-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202308-divvy-tripdata',
MAX(CASE
    WHEN TABLE_NAME = '202309-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202309-divvy-tripdata',
MAX(CASE
    WHEN TABLE_NAME = '202310-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202310-divvy-tripdata',
MAX(CASE
    WHEN TABLE_NAME = '202311-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202311-divvy-tripdata',
MAX(CASE
    WHEN TABLE_NAME = '202312-divvy-tripdata' THEN 'Yes' ELSE 'No'
    END) AS '202312-divvy-tripdata'
FROM
(
SELECT
*
FROM
INFORMATION_SCHEMA.COLUMNS
) AS Columns
GROUP BY
    COLUMN_NAME
```



	COLUMN_NAME	202301-divvy-tripdata	202302-divvy-tripdata	202303-divvy-tripdata	202304-divvy-tripdata	202305-divvy-tripdata	202306-divvy-tripdata	202307-divvy-tripdata	202308-divvy-tripdata	202309-divvy-tripdata	202310-divvy-tripdata	202311-
1	end_lat	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	end_lng	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3	end_station_id	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4	end_station_name	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	ended_at	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6	member_casual	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
7	ride_id	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
8	rideable_type	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
9	start_lat	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
10	start_lng	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
11	start_station_id	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
12	start_station_name	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
13	started_at	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Column Name Comparison

Upon confirming that the column names in all tables are the same, it is possible to proceed directly to review the number of rows that exist across all tables with the following query. SELECT allows displaying the numeric value given by the COUNT function placed in subqueries, and this numeric value is summed up, resulting in a total of 5,719,877 rows.

```
SELECT
(
SELECT
COUNT(*)
FROM
"202301-divvy-tripdata"
) +
(
SELECT
COUNT(*)
FROM
"202302-divvy-tripdata"
) +
(
SELECT
COUNT(*)
FROM
"202303-divvy-tripdata"
) +
(
SELECT
COUNT(*)
FROM
"202304-divvy-tripdata"
) +
(
SELECT
COUNT(*)
FROM
"202305-divvy-tripdata"
) +
(
SELECT
COUNT(*)
FROM
"202306-divvy-tripdata"
) +
(
SELECT
COUNT(*)
FROM
"202307-divvy-tripdata"
) +
(
SELECT
COUNT(*)
FROM
"202308-divvy-tripdata"
) +
(
SELECT
COUNT(*)
FROM
"202309-divvy-tripdata"
) +
(
SELECT
COUNT(*)
FROM
"202310-divvy-tripdata"
) +
(
SELECT
COUNT(*)
FROM
"202311-divvy-tripdata"
) +
(
SELECT
COUNT(*)
FROM
"202312-divvy-tripdata"
) AS total_rows
```

Combining Tables

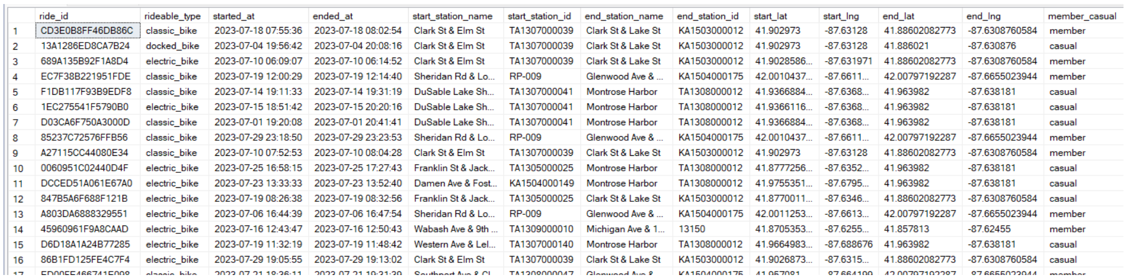
Confirming that all tables have the same columns and these columns have the same name, the creation of the first temporary table will be started. This table will be the starting point for the creation of the final table with clean data and the necessary columns for analysis and calculations. The use of temporary tables is essential from this moment because it will be necessary to perform queries of tables created earlier by other queries, this, in addition to facilitating the differentiation of each step of the process, also allows for appropriate backtracking in case it is necessary.

For the creation of this first table, it is necessary to combine the rows of all tables into a single table, in this way the data cleaning will be applied uniformly. The following query performs this combination and generates a single temporary table called #cyclistic_raw with 5,719,877 rows.

```
SELECT
*
INTO
#cyclistic_raw
FROM
(
SELECT
*
FROM
"202301-divvy-tripdata"
UNION ALL
SELECT
*
```

```
FROM
"202302-divvy-tripdata"
UNION ALL
SELECT
*
FROM
"202303-divvy-tripdata"
UNION ALL
SELECT
*
FROM
"202304-divvy-tripdata"
UNION ALL
SELECT
*
FROM
"202305-divvy-tripdata"
UNION ALL
SELECT
*
FROM
"202306-divvy-tripdata"
UNION ALL
SELECT
*
FROM
"202307-divvy-tripdata"
UNION ALL
SELECT
*
FROM
"202308-divvy-tripdata"
UNION ALL
SELECT
*
FROM
"202309-divvy-tripdata"
UNION ALL
SELECT
*
FROM
"202310-divvy-tripdata"
UNION ALL
SELECT
*
FROM
"202311-divvy-tripdata"
UNION ALL
```

```
SELECT
*
FROM
"202312-divvy-tripdata"
) AS tables_rows_combined
---------------------------------------
SELECT
*
FROM
#cyclistic_raw
```



	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
1	CD3E0B8FF46DB86C	classic_bike	2023-07-18 07:55:36	2023-07-18 08:02:54	Clark St & Elm St	TA1307000039	Clark St & Lake St	KA1503000012	41.902973	-87.63128	41.88602082773	-87.6308760584	member
2	13A1286ED8CA7B24	docked_bike	2023-07-04 19:56:42	2023-07-04 20:08:16	Clark St & Elm St	TA1307000039	Clark St & Lake St	KA1503000012	41.902973	-87.63128	41.886021	-87.630876	casual
3	689A135B92F1A8D4	electric_bike	2023-07-10 06:09:07	2023-07-10 06:14:52	Clark St & Elm St	TA1307000039	Clark St & Lake St	KA1503000012	41.9028586...	-87.631971	41.88602082773	-87.6308760584	member
4	EC7F38B221951FDE	classic_bike	2023-07-19 12:00:29	2023-07-19 12:14:40	Sheridan Rd & Lo...	RP-009	Glenwood Ave & ...	KA1504000175	42.0010437...	-87.6611...	42.00797192287	-87.6655023944	member
5	F1DB117F93B9EDF8	classic_bike	2023-07-14 19:11:33	2023-07-14 19:31:19	DuSable Lake Sh...	TA1307000041	Montrose Harbor	TA1308000012	41.9366884...	-87.6368...	41.963982	-87.638181	casual
6	1EC275541F5790B0	electric_bike	2023-07-15 18:51:42	2023-07-15 20:20:16	DuSable Lake Sh...	TA1307000041	Montrose Harbor	TA1308000012	41.9366116...	-87.6368...	41.963982	-87.638181	casual
7	D03CA6F750A3000D	classic_bike	2023-07-01 19:20:08	2023-07-01 20:41:41	DuSable Lake Sh...	TA1307000041	Montrose Harbor	TA1308000012	41.9366884...	-87.6368...	41.963982	-87.638181	casual
8	85237C72576FFB56	classic_bike	2023-07-29 23:18:50	2023-07-29 23:23:53	Sheridan Rd & Lo...	RP-009	Glenwood Ave & ...	KA1504000175	42.0010437...	-87.6611...	42.00797192287	-87.6655023944	member
9	A27115CC44080E34	classic_bike	2023-07-10 07:52:53	2023-07-10 08:04:28	Clark St & Elm St	TA1307000039	Clark St & Lake St	KA1503000012	41.902973	-87.63128	41.88602082773	-87.6308760584	member
10	0060951C02440D4F	electric_bike	2023-07-25 16:58:15	2023-07-25 17:27:43	Franklin St & Jack...	TA1305000025	Montrose Harbor	TA1308000012	41.8777256...	-87.6352...	41.963982	-87.638181	casual
11	DCCED51A061E67A0	electric_bike	2023-07-23 13:33:33	2023-07-23 13:52:40	Damen Ave & Fost...	KA1504000149	Montrose Harbor	TA1308000012	41.9755351...	-87.6795...	41.963982	-87.638181	casual
12	847B5A6F688F121B	electric_bike	2023-07-19 08:26:38	2023-07-19 08:32:56	Franklin St & Jack...	TA1305000025	Clark St & Lake St	KA1503000012	41.8770011...	-87.6346...	41.88602082773	-87.6308760584	casual
13	A803DA6888329551	electric_bike	2023-07-06 16:44:39	2023-07-06 16:47:54	Sheridan Rd & Lo...	RP-009	Glenwood Ave & ...	KA1504000175	42.0011253...	-87.6613...	42.00797192287	-87.6655023944	member
14	45960961F9A8CAAD	electric_bike	2023-07-16 12:43:47	2023-07-16 12:50:43	Wabash Ave & 9th ...	TA1309000010	Michigan Ave & 1...	13150	41.8705353...	-87.6255...	41.857813	-87.62455	member
15	D6D18A1A24B77285	electric_bike	2023-07-19 11:32:19	2023-07-19 11:48:42	Western Ave & Lel...	TA1307000140	Montrose Harbor	TA1308000012	41.9664983...	-87.688676	41.963982	-87.638181	casual
16	86B1FD125FE4C7F4	electric_bike	2023-07-29 19:05:55	2023-07-29 19:13:02	Clark St & Elm St	TA1307000039	Clark St & Lake St	KA1503000012	41.9026873...	-87.6315...	41.88602082773	-87.6308760584	casual
17	ED00FF466741E008	classic_bike	2023-07-21 18:36:11	2023-07-21 19:31:39	Southport Ave & Cl...	TA1308000047	Glenwood Ave & ...	KA1504000175	41.957081	-87.664199	42.00797192287	-87.6655023944	member

#cyclistic_raw

To track the temporary tables created in SQL Server, you can use the following query.

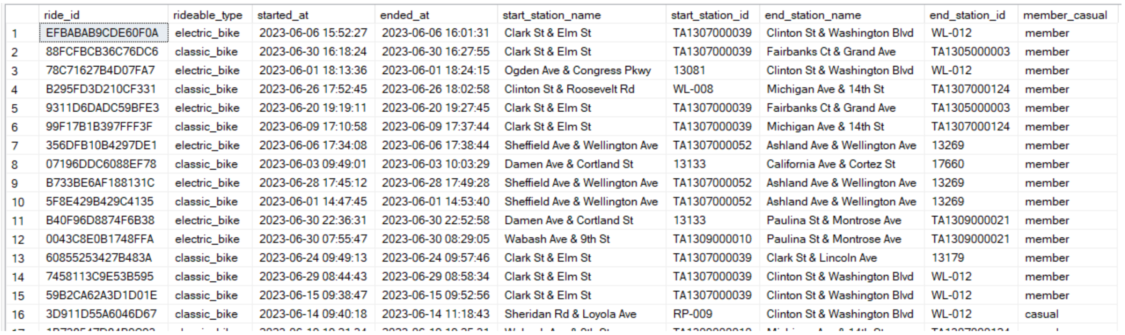
```
USE tempdb;
GO
SELECT
*
FROM
sys.tables
WHERE name LIKE '#%';
```

Removal of Columns and Trimming of Text Strings

Upon inspecting the data, it has been determined that certain columns are not relevant for analysis. Therefore, these columns will now be removed from the table. Additionally, any leading and trailing white spaces in text string values will be removed simultaneously.

The following query specifies which columns from the table #cyclistic_raw will be displayed, and at the same time, columns containing text string values will be processed by the TRIM function to remove any leading and trailing white spaces. The new table created by the query will be named #cyclistic_trim.

```
SELECT
TRIM(ride_id) AS ride_id,
TRIM(rideable_type) AS rideable_type,
started_at AS started_at,
ended_at AS ended_at,
TRIM(start_station_name) AS start_station_name,
TRIM(start_station_id) AS start_station_id,
TRIM(end_station_name) AS end_station_name,
TRIM(end_station_id) AS end_station_id,
TRIM(member_casual) AS member_casual
INTO
#cyclistic_trim
FROM
#cyclistic_raw
------------------------------------------------------------------
SELECT
*
FROM
#cyclistic_trim
```



	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	member_casual
1	EFBABAB9CDE60F0A	electric_bike	2023-06-06 15:52:27	2023-06-06 16:01:31	Clark St & Elm St	TA1307000039	Clinton St & Washington Blvd	WL-012	member
2	88FCFBCB36C76DC6	classic_bike	2023-06-30 16:18:24	2023-06-30 16:27:55	Clark St & Elm St	TA1307000039	Fairbanks Ct & Grand Ave	TA1305000003	member
3	78C71627B4D07FA7	electric_bike	2023-06-01 18:13:36	2023-06-01 18:24:15	Ogden Ave & Congress Pkwy	13081	Clinton St & Washington Blvd	WL-012	member
4	B295FD3D210CF331	classic_bike	2023-06-26 17:52:45	2023-06-26 18:02:58	Clinton St & Roosevelt Rd	WL-008	Michigan Ave & 14th St	TA1307000124	member
5	9311D6DADC59BFE3	electric_bike	2023-06-20 19:19:11	2023-06-20 19:27:45	Clark St & Elm St	TA1307000039	Fairbanks Ct & Grand Ave	TA1305000003	member
6	99F17B1B397FFF3F	classic_bike	2023-06-09 17:10:58	2023-06-09 17:37:44	Clark St & Elm St	TA1307000039	Michigan Ave & 14th St	TA1307000124	member
7	356DFB10B4297DE1	electric_bike	2023-06-06 17:34:08	2023-06-06 17:38:44	Sheffield Ave & Wellington Ave	TA1307000052	Ashland Ave & Wellington Ave	13269	member
8	07196DDC6088EF78	classic_bike	2023-06-03 09:49:01	2023-06-03 10:03:29	Damen Ave & Cortland St	13133	California Ave & Cortez St	17660	member
9	B733BE6AF188131C	electric_bike	2023-06-28 17:45:12	2023-06-28 17:49:28	Sheffield Ave & Wellington Ave	TA1307000052	Ashland Ave & Wellington Ave	13269	member
10	5F8E429B429C4135	classic_bike	2023-06-01 14:47:45	2023-06-01 14:53:40	Sheffield Ave & Wellington Ave	TA1307000052	Ashland Ave & Wellington Ave	13269	member
11	B40F96D8874F6B38	electric_bike	2023-06-30 22:36:31	2023-06-30 22:52:58	Damen Ave & Cortland St	13133	Paulina St & Montrose Ave	TA1309000021	member
12	0043C8E0B1748FFA	electric_bike	2023-06-30 07:55:47	2023-06-30 08:29:05	Wabash Ave & 9th St	TA1309000010	Paulina St & Montrose Ave	TA1309000021	member
13	60855253427B483A	classic_bike	2023-06-24 09:49:13	2023-06-24 09:57:46	Clark St & Elm St	TA1307000039	Clark St & Lincoln Ave	13179	member
14	7458113C9E53B595	classic_bike	2023-06-29 08:44:43	2023-06-29 08:58:34	Clark St & Elm St	TA1307000039	Clinton St & Washington Blvd	WL-012	member
15	59B2CA62A3D1D01E	classic_bike	2023-06-15 09:38:47	2023-06-15 09:52:56	Clark St & Elm St	TA1307000039	Clinton St & Washington Blvd	WL-012	member
16	3D911D55A6046D67	classic_bike	2023-06-14 09:40:18	2023-06-14 11:18:43	Sheridan Rd & Loyola Ave	RP-009	Clinton St & Washington Blvd	WL-012	casual

#cyclistic_trim

Removal of Rows Filled with NULL Values

As a precautionary measure, rows that are completely filled with NULL values will now be removed. To do this, the table #cyclistic_temp is created, which is a copy of #cyclistic_trim. Then, it is specified to delete rows that meet the condition given by the WHERE clause, that is, those in which all columns have a NULL value.

```
SELECT
*
INTO
#cyclistic_temp
FROM
#cyclistic_trim
-----------------------------
DELETE FROM
#cyclistic_temp
WHERE
ride_id IS NULL AND
rideable_type IS NULL AND
started_at IS NULL AND
ended_at IS NULL AND
start_station_name IS NULL AND
start_station_id IS NULL AND
```

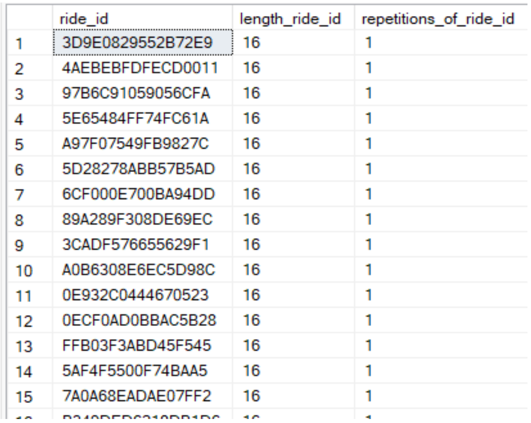
```
end_station_name IS NULL AND
end_station_id IS NULL AND
member_casual IS NULL
```

After executing the query, the message "(0 rows affected)" confirms that none of the more than 5 million rows are entirely empty. However, rows with partial NULL values will be retained, as they may still contain useful information for analysis.

Text Strings Correction

In addition to correcting errors in text strings, their length will be reviewed in columns where it should be the same across all rows. To start, the column "ride_id" is observed. Then, a query is executed to count the number of characters and an additional column called "repetitions_of_ride_id" is added with a COUNT to indicate if there are any repeated text strings.

```
SELECT
ride_id,
LEN(ride_id) AS length_ride_id,
COUNT(ride_id) AS repetitions_of_ride_id
FROM
#cyclistic_temp
GROUP BY
ride_id
```



	ride_id	length_ride_id	repetitions_of_ride_id
1	3D9E0829552B72E9	16	1
2	4AEBEBFDFECD0011	16	1
3	97B6C91059056CFA	16	1
4	5E65484FF74FC61A	16	1
5	A97F07549FB9827C	16	1
6	5D28278ABB57B5AD	16	1
7	6CF000E700BA94DD	16	1
8	89A289F308DE69EC	16	1
9	3CADF576655629F1	16	1
10	A0B6308E6EC5D98C	16	1
11	0E932C0444670523	16	1
12	0ECF0AD0BBAC5B28	16	1
13	FFB03F3ABD45F545	16	1
14	5AF4F5500F74BAA5	16	1
15	7A0A68EADAE07FF2	16	1

At first glance, it appears that the length of the text string in the "ride_id" column is 16 characters, and there may be no repeated rows. However, this is confirmed with another query where all rows with a character length different from 16 are filtered.

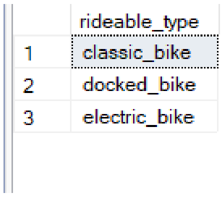
```
SELECT
ride_id,
LEN(ride_id) AS length_ride_id,
COUNT(ride_id) AS repetitions_of_ride_id
FROM
#cyclistic_temp
```

```
WHERE
LEN(ride_id) <> 16
GROUP BY
ride_id
```

The result is an empty table, which means that there are no values with a length different from 16 characters. Therefore, all rows contain the correct length, and there is no need for additional corrections.

Moving on to the "rideable_type" column, we can verify if there are any categories that need correction by selecting the unique values using SELECT DISTINCT as follows.

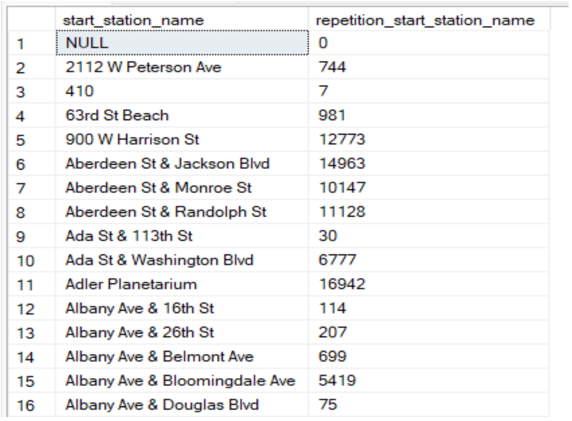
```
SELECT DISTINCT
rideable_type
FROM
#cyclistic_temp
```



	rideable_type
1	classic_bike
2	docked_bike
3	electric_bike

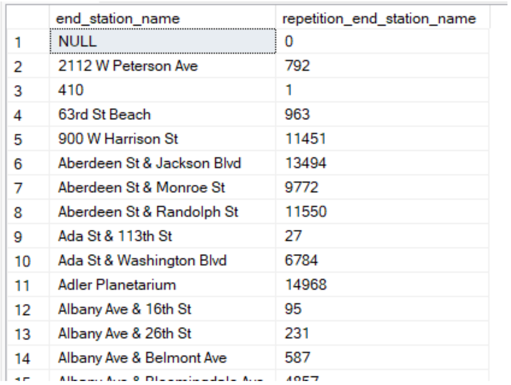
The result confirms that all rows are correctly written and that there are three categories in this column. Then, the same procedure is applied to the text strings in "start_station_name" and "end_station_name".

```
SELECT DISTINCT
start_station_name,
COUNT(start_station_name) AS repetition_start_station_name
FROM
#cyclistic_temp
GROUP BY
start_station_name
ORDER BY
start_station_name
```



	start_station_name	repetition_start_station_name
1	NULL	0
2	2112 W Peterson Ave	744
3	410	7
4	63rd St Beach	981
5	900 W Harrison St	12773
6	Aberdeen St & Jackson Blvd	14963
7	Aberdeen St & Monroe St	10147
8	Aberdeen St & Randolph St	11128
9	Ada St & 113th St	30
10	Ada St & Washington Blvd	6777
11	Adler Planetarium	16942
12	Albany Ave & 16th St	114
13	Albany Ave & 26th St	207
14	Albany Ave & Belmont Ave	699
15	Albany Ave & Bloomingdale Ave	5419
16	Albany Ave & Douglas Blvd	75

```
SELECT DISTINCT
end_station_name,
COUNT(end_station_name) AS repetition_end_station_name
FROM
#cyclistic_temp
GROUP BY
end_station_name
ORDER BY
end_station_name
```

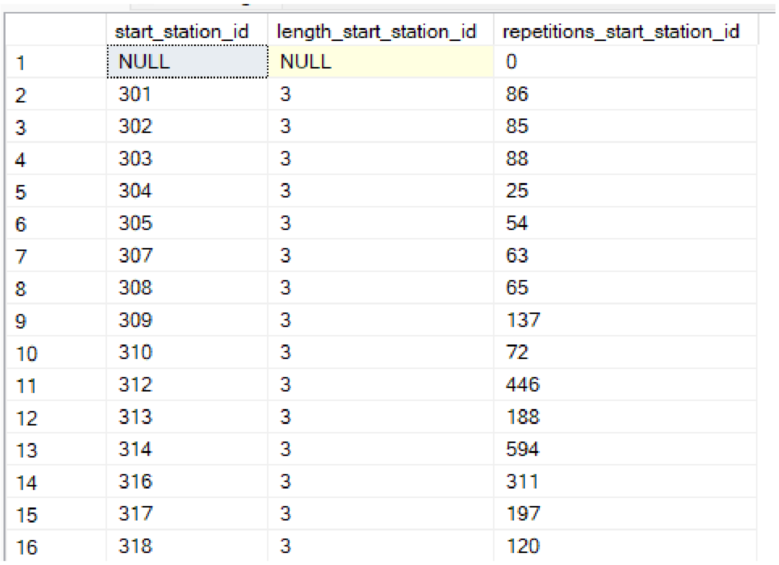


	end_station_name	repetition_end_station_name
1	NULL	0
2	2112 W Peterson Ave	792
3	410	1
4	63rd St Beach	963
5	900 W Harrison St	11451
6	Aberdeen St & Jackson Blvd	13494
7	Aberdeen St & Monroe St	9772
8	Aberdeen St & Randolph St	11550
9	Ada St & 113th St	27
10	Ada St & Washington Blvd	6784
11	Adler Planetarium	14968
12	Albany Ave & 16th St	95
13	Albany Ave & 26th St	231
14	Albany Ave & Belmont Ave	587
15	Albany Ave & Bloomingdale Ave	4857

When the result of both queries is observed, it can be determined that there is no need for correction in any of the rows, however, it is also observed how many times these stations are repeated, unlike “ride_id” there is no relevance with these values being repeated.

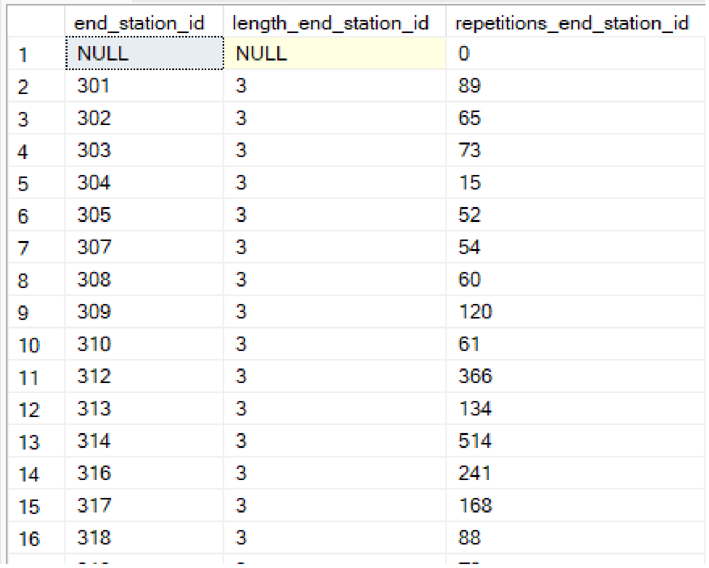
Now it is checked whether correction is necessary in “start_station_id” and “end_station_id”.

```
SELECT
start_station_id,
LEN(start_station_id) AS length_start_station_id,
COUNT(start_station_id) AS repetitions_start_station_id
FROM
#cyclistic_temp
GROUP BY
start_station_id
ORDER BY
length_start_station_id, start_station_id
```



	start_station_id	length_start_station_id	repetitions_start_station_id
1	NULL	NULL	0
2	301	3	86
3	302	3	85
4	303	3	88
5	304	3	25
6	305	3	54
7	307	3	63
8	308	3	65
9	309	3	137
10	310	3	72
11	312	3	446
12	313	3	188
13	314	3	594
14	316	3	311
15	317	3	197
16	318	3	120

```
SELECT
end_station_id,
LEN(end_station_id) AS length_end_station_id,
COUNT(end_station_id) AS repetitions_end_station_id
FROM
#cyclistic_trim
GROUP BY
end_station_id
ORDER BY
length_end_station_id, end_station_id
```

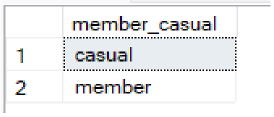


	end_station_id	length_end_station_id	repetitions_end_station_id
1	NULL	NULL	0
2	301	3	89
3	302	3	65
4	303	3	73
5	304	3	15
6	305	3	52
7	307	3	54
8	308	3	60
9	309	3	120
10	310	3	61
11	312	3	366
12	313	3	134
13	314	3	514
14	316	3	241
15	317	3	168
16	318	3	88

In the same way, it is determined that no changes are necessary, and the repetition of the values also does not impact the analysis. It is confirmed that the length of the characters in both columns changes, so this length is not constant in the rows of the two columns.

Finally, the category of user types is reviewed. Upon seeing the result of the query, it is concluded that no text string needs changes and it confirms that there are indeed two types of users.

```
SELECT DISTINCT
member_casual
FROM
#cyclistic_temp
```



	member_casual
1	casual
2	member

Removal of Duplicate Rows

Having corrected the text strings, it is time to remove duplicate rows. This is achieved by using SELECT DISTINCT and the resulting table will be the temporary table #cyclistic_distinct.

```
SELECT DISTINCT
*
INTO
#cyclistic_distinct
FROM
#cyclistic_temp
```

The query returns the message "(5719877 rows affected)," confirming the absence of duplicate rows. However, this temporary table created will be used for the subsequent processes.

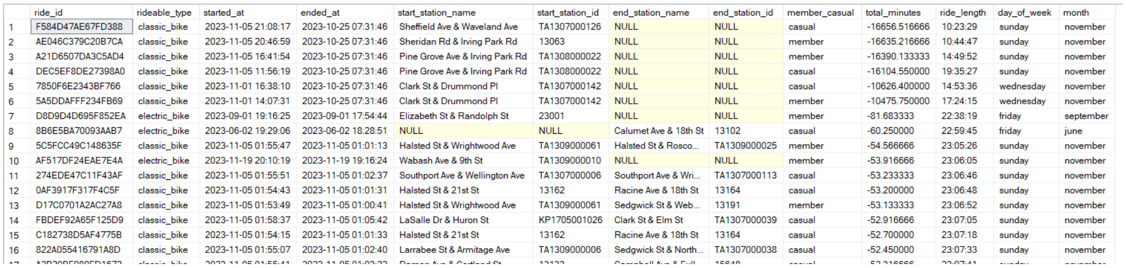
Addition of Calculated Columns

With the columns having their proper data type, format, and corrected text strings, it is possible to add columns whose calculations and information can enhance the analysis.

According to the information given by the table so far, it will be necessary to include the length of the rides converted into minutes, as well as the exact day and month when the rides were made. The following query precisely adds the necessary columns.

The first column converts the length of the rides into minutes using the numerical value provided by DATEDIFF, which is then divided by 60. Then, as a complement and illustration of the previous column, another column is added with these minutes in time format. Next, using DATENAME, another column is added to display the day of the week when the ride started, with the resulting text string being forced to lowercase using LOWER. Finally, a column is added with the same process as before but now showing the month.

```
SELECT
*,
DATEDIFF(SECOND, started_at, ended_at)/60.0 AS total_minutes,
FORMAT(CONVERT(DATETIME, ended_at)-CONVERT(DATETIME, started_at),
'HH:mm:ss') AS ride_length,
LOWER(DATENAME(dw, started_at)) AS day_of_week,
LOWER(DATENAME(month, started_at)) AS "month"
INTO
#cyclistic_timedate
FROM
#cyclistic_distinct
ORDER BY
total_minutes, ride_length
--------------------------------------------------------------
SELECT
*
FROM
#cyclistic_timedate
ORDER BY
total_minutes, ride_length
```



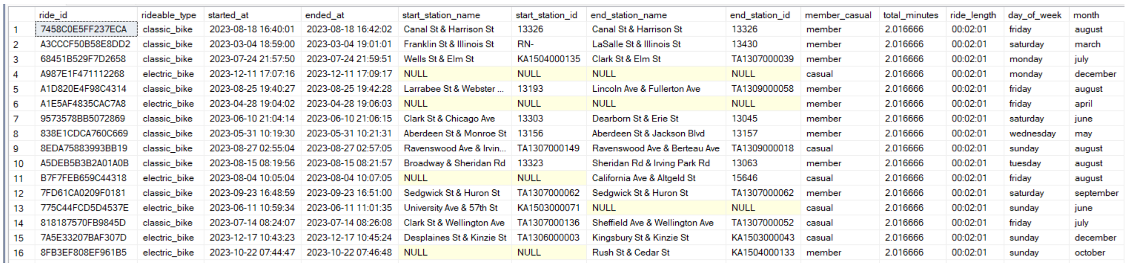
	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	member_casual	total_minutes	ride_length	day_of_week	month
1	F584D47AE67FD388	classic_bike	2023-11-05 21:08:17	2023-10-25 07:31:46	Sheffield Ave & Waveland Ave	TA1307000126	NULL	NULL	casual	-16656.516666	10:23:29	sunday	november
2	AE046C379C20B7CA	classic_bike	2023-11-05 20:46:59	2023-10-25 07:31:46	Sheridan Rd & Irving Park Rd	13063	NULL	NULL	member	-16635.216666	10:44:47	sunday	november
3	A21D6507DA3C5AD4	classic_bike	2023-11-05 16:41:54	2023-10-25 07:31:46	Pine Grove Ave & Irving Park Rd	TA1308000022	NULL	NULL	member	-16390.133333	14:49:52	sunday	november
4	DEC5EF8DE27398A0	classic_bike	2023-11-05 11:56:19	2023-10-25 07:31:46	Pine Grove Ave & Irving Park Rd	TA1308000022	NULL	NULL	casual	-16104.550000	19:35:27	sunday	november
5	7850F6E2343BF766	classic_bike	2023-11-01 16:38:10	2023-10-25 07:31:46	Clark St & Drummond Pl	TA1307000142	NULL	NULL	casual	-10626.400000	14:53:36	wednesday	november
6	5A5DDAFFF234FB69	classic_bike	2023-11-01 14:07:31	2023-10-25 07:31:46	Clark St & Drummond Pl	TA1307000142	NULL	NULL	member	-10475.750000	17:24:15	wednesday	november
7	D8D9D4D695F852EA	electric_bike	2023-09-01 19:16:25	2023-09-01 17:54:44	Elizabeth St & Randolph St	23001	NULL	NULL	member	-81.683333	22:38:19	friday	september
8	8B6E5BA70093AAB7	electric_bike	2023-06-02 19:29:06	2023-06-02 18:28:51	NULL	NULL	Calumet Ave & 18th St	13102	casual	-60.250000	22:59:45	friday	june
9	5C5FCC49C148635F	classic_bike	2023-11-05 01:55:47	2023-11-05 01:01:13	Halsted St & Wrightwood Ave	TA1309000061	Halsted St & Rosco...	TA1309000025	member	-54.566666	23:05:26	sunday	november
10	AF517DF24EAE7E4A	electric_bike	2023-11-19 20:10:19	2023-11-19 19:16:24	Wabash Ave & 9th St	TA1309000010	NULL	NULL	member	-53.916666	23:06:05	sunday	november
11	274EDE47C11F43AF	classic_bike	2023-11-05 01:55:51	2023-11-05 01:02:37	Southport Ave & Wellington Ave	TA1307000006	Southport Ave & Wri...	TA1307000113	casual	-53.233333	23:06:46	sunday	november
12	0AF3917F317F4C5F	classic_bike	2023-11-05 01:54:43	2023-11-05 01:01:31	Halsted St & 21st St	13162	Racine Ave & 18th St	13164	casual	-53.200000	23:06:48	sunday	november
13	D17C0701A2AC27A8	classic_bike	2023-11-05 01:53:49	2023-11-05 01:00:41	Halsted St & Wrightwood Ave	TA1309000061	Sedgwick St & Web...	13191	member	-53.133333	23:06:52	sunday	november
14	FBDEF92A65F125D9	classic_bike	2023-11-05 01:58:37	2023-11-05 01:05:42	LaSalle Dr & Huron St	KP1705001026	Clark St & Elm St	TA1307000039	casual	-52.916666	23:07:05	sunday	november
15	C182738D5AF4775B	classic_bike	2023-11-05 01:54:15	2023-11-05 01:01:33	Halsted St & 21st St	13162	Racine Ave & 18th St	13164	casual	-52.700000	23:07:18	sunday	november
16	822A055416791A8D	classic_bike	2023-11-05 01:55:07	2023-11-05 01:02:40	Larrabee St & Armitage Ave	TA1309000006	Sedgwick St & North...	TA1307000038	casual	-52.450000	23:07:33	sunday	november
17	A2B20BF080FD1672	classic_bike	2023-11-05 01:55:41	2023-11-05 01:03:22	Damen Ave & Cortland St	13133	Campbell Ave & Full...	15648	casual	-52.216666	23:07:41	sunday	november

#cyclistic_timedate

Upon reviewing the data in the "total_minutes” column, it is decided to remove negative values and constrain the values where the ride duration is greater than one minute but less than one day. This avoids an analysis that lacks sense and increases coherence. The query applies these conditions simply by adding them to the WHERE clause, and the temporary table created by the query will be named #cyclistic_table.

```
SELECT
*
INTO
#cyclistic_table
FROM
#cyclistic_timedate
WHERE
total_minutes > 2 and
total_minutes < 1440
ORDER BY
total_minutes, ride_length
------------------------------------------
SELECT
*
FROM
#cyclistic_table
ORDER BY
total_minutes, ride_length
```

The query throws the message “(5447295 rows affected)”, confirming the creation of the temporary table with a reduced number of rows compared to the previous table. This reduction is due to the specified conditions being applied. Therefore, the #cyclistic_table is the final table that will be created, containing a total of 5,447,295 trips relevant for analysis.



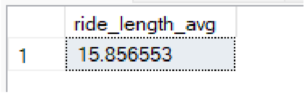
	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	member_casual	total_minutes	ride_length	day_of_week	month
1	7458C0E5FF237ECA	classic_bike	2023-08-18 16:40:01	2023-08-18 16:42:02	Canal St & Harrison St	13326	Canal St & Harrison St	13326	member	2.016666	00:02:01	friday	august
2	A3CCCF50B58E8DD2	classic_bike	2023-03-04 18:59:00	2023-03-04 19:01:01	Franklin St & Illinois St	RN-	LaSalle St & Illinois St	13430	member	2.016666	00:02:01	saturday	march
3	68451B529F7D2658	classic_bike	2023-07-24 21:57:50	2023-07-24 21:59:51	Wells St & Elm St	KA1504000135	Clark St & Elm St	TA1307000039	member	2.016666	00:02:01	monday	july
4	A987E1F471112268	electric_bike	2023-12-11 17:07:16	2023-12-11 17:09:17	NULL	NULL	NULL	NULL	casual	2.016666	00:02:01	monday	december
5	A1D820E4F98C4314	classic_bike	2023-08-25 19:40:27	2023-08-25 19:42:28	Larrabee St & Webster ...	13193	Lincoln Ave & Fullerton Ave	TA1309000058	member	2.016666	00:02:01	friday	august
6	A1E5AF4835CAC7A8	electric_bike	2023-04-28 19:04:02	2023-04-28 19:06:03	NULL	NULL	NULL	NULL	member	2.016666	00:02:01	friday	april
7	9573578BB5072869	classic_bike	2023-06-10 21:04:14	2023-06-10 21:06:15	Clark St & Chicago Ave	13303	Dearborn St & Erie St	13045	member	2.016666	00:02:01	saturday	june
8	838E1CDCA760C669	classic_bike	2023-05-31 10:19:30	2023-05-31 10:21:31	Aberdeen St & Monroe St	13156	Aberdeen St & Jackson Blvd	13157	member	2.016666	00:02:01	wednesday	may
9	8EDA75883993BB19	classic_bike	2023-08-27 02:55:04	2023-08-27 02:57:05	Ravenswood Ave & Irvin...	TA1307000149	Ravenswood Ave & Berteau Ave	TA1309000018	casual	2.016666	00:02:01	sunday	august
10	A5DEB5B3B2A01A0B	classic_bike	2023-08-15 08:19:56	2023-08-15 08:21:57	Broadway & Sheridan Rd	13323	Sheridan Rd & Irving Park Rd	13063	member	2.016666	00:02:01	tuesday	august
11	B7F7FEB659C44318	electric_bike	2023-08-04 10:05:04	2023-08-04 10:07:05	NULL	NULL	California Ave & Altgeld St	15646	casual	2.016666	00:02:01	friday	august
12	7FD61CA0209F0181	classic_bike	2023-09-23 16:48:59	2023-09-23 16:51:00	Sedgwick St & Huron St	TA1307000062	Sedgwick St & Huron St	TA1307000062	member	2.016666	00:02:01	saturday	september
13	775C44FCD5D4537E	electric_bike	2023-06-11 10:59:34	2023-06-11 11:01:35	University Ave & 57th St	KA1503000071	NULL	NULL	casual	2.016666	00:02:01	sunday	june
14	818187570FB9845D	classic_bike	2023-07-14 08:24:07	2023-07-14 08:26:08	Clark St & Wellington Ave	TA1307000136	Sheffield Ave & Wellington Ave	TA1307000052	casual	2.016666	00:02:01	friday	july
15	7A5E33207BAF307D	electric_bike	2023-12-17 10:43:23	2023-12-17 10:45:24	Desplaines St & Kinzie St	TA1306000003	Kingsbury St & Kinzie St	KA1503000043	casual	2.016666	00:02:01	sunday	december
16	8FB3EF808EF961B5	electric_bike	2023-10-22 07:44:47	2023-10-22 07:46:48	NULL	NULL	Rush St & Cedar St	KA1504000133	member	2.016666	00:02:01	sunday	october

#cyclistic_table

Analysis

Starting from the #cyclistic_table, it is possible to begin the analysis.To start getting a general idea of the length of the rides, the average length is first calculated. This way, a certain range can be assumed when observing the duration of the rides for each type of user.

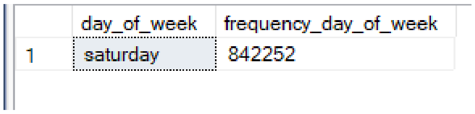
```
SELECT
AVG(total_minutes) AS ride_length_avg
FROM
#cyclistic_table
```



	ride_length_avg
1	15.856553

Now, the day with the highest number of rides made is determined. This is useful as it indicates the general preference or need of users to use the service.

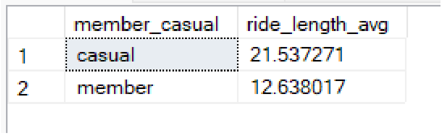
```
SELECT TOP 1
day_of_week,
COUNT(day_of_week) AS frequency_day_of_week
FROM
#cyclistic_table
GROUP BY
day_of_week
ORDER BY
frequency_day_of_week DESC
```



	day_of_week	frequency_day_of_week
1	saturday	842252

To get the first glimpses of the difference between both types of users, the average ride duration is calculated for each one.

```
SELECT
member_casual,
AVG(total_minutes) AS ride_length_avg
FROM
#cyclistic_table
GROUP BY
member_casual
```

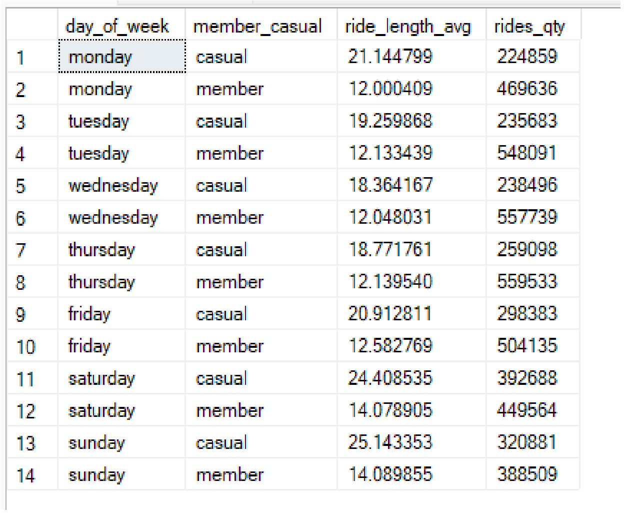


	member_casual	ride_length_avg
1	casual	21.537271
2	member	12.638017

It is observed that casual users have longer rides than member users. Users' weekly behavior is now observed to determine their differences more deeply and detailed.

The following query shows on each day of the week, the type of user, their average ride length, and the number of rides made. From here, it is necessary to organize the data in the correct order, that is, from Monday to Sunday.

```
WITH ordered_days AS
(
SELECT
*,
CASE
WHEN day_of_week = 'monday' THEN 1
WHEN day_of_week = 'tuesday' THEN 2
WHEN day_of_week = 'wednesday' THEN 3
WHEN day_of_week = 'thursday' THEN 4
WHEN day_of_week = 'friday' THEN 5
WHEN day_of_week = 'saturday' THEN 6
WHEN day_of_week = 'sunday' THEN 7
END AS day_of_week_order
FROM
#cyclistic_table
)
SELECT
day_of_week,
member_casual,
AVG(total_minutes) AS ride_length_avg,
COUNT(ride_id) AS rides_qty
FROM
ordered_days
GROUP BY
day_of_week, member_casual, day_of_week_order
ORDER BY
day_of_week_order
```

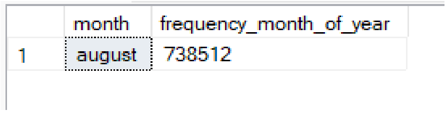


	day_of_week	member_casual	ride_length_avg	rides_qty
1	monday	casual	21.144799	224859
2	monday	member	12.000409	469636
3	tuesday	casual	19.259868	235683
4	tuesday	member	12.133439	548091
5	wednesday	casual	18.364167	238496
6	wednesday	member	12.048031	557739
7	thursday	casual	18.771761	259098
8	thursday	member	12.139540	559533
9	friday	casual	20.912811	298383
10	friday	member	12.582769	504135
11	saturday	casual	24.408535	392688
12	saturday	member	14.078905	449564
13	sunday	casual	25.143353	320881
14	sunday	member	14.089855	388509

It can be seen that the highest number of rides belongs to member users and the longest rides to casual users.

Having obtained the information on a weekly basis, it can be obtained in the same way on a monthly basis. First, the month with the most rides is determined.

```
SELECT TOP 1
"month",
COUNT("month") AS frequency_month_of_year
FROM
#cyclistic_table
GROUP BY
"month"
ORDER BY
frequency_month_of_year DESC
```

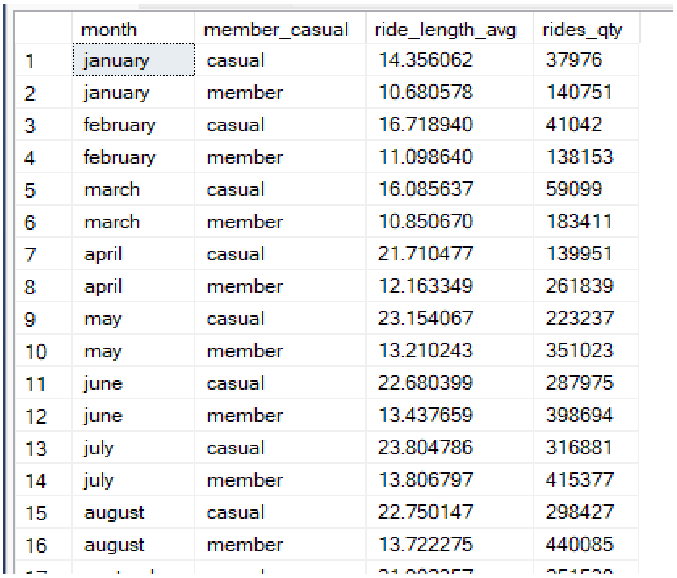


	month	frequency_month_of_year
1	august	738512

A query is performed again where the average ride duration and the number of rides made by each user type are displayed, but this time, the information is shown for each month. Also, the data is organized from January to December.

```
WITH ordered_months AS
(
SELECT
*,
CASE
WHEN "month" = 'january' THEN 1
WHEN "month" = 'february' THEN 2
WHEN "month" = 'march' THEN 3
WHEN "month" = 'april' THEN 4
WHEN "month" = 'may' THEN 5
WHEN "month" = 'june' THEN 6
WHEN "month" = 'july' THEN 7
WHEN "month" = 'august' THEN 8
WHEN "month" = 'september' THEN 9
WHEN "month" = 'october' THEN 10
WHEN "month" = 'november' THEN 11
WHEN "month" = 'december' THEN 12
END AS month_of_year
FROM #cyclistic_table
)
SELECT
"month",
member_casual,
AVG(total_minutes) AS ride_length_avg,
COUNT(ride_id) AS rides_qty
FROM
ordered_months
```

```
GROUP BY
"month", member_casual, month_of_year
ORDER BY
month_of_year
```

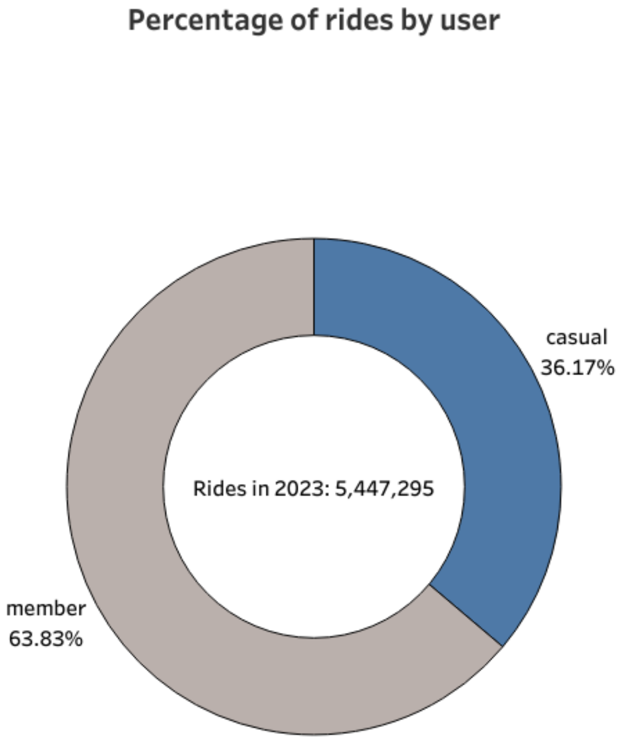


	month	member_casual	ride_length_avg	rides_qty
1	january	casual	14.356062	37976
2	january	member	10.680578	140751
3	february	casual	16.718940	41042
4	february	member	11.098640	138153
5	march	casual	16.085637	59099
6	march	member	10.850670	183411
7	april	casual	21.710477	139951
8	april	member	12.163349	261839
9	may	casual	23.154067	223237
10	may	member	13.210243	351023
11	june	casual	22.680399	287975
12	june	member	13.437659	398694
13	july	casual	23.804786	316881
14	july	member	13.806797	415377
15	august	casual	22.750147	298427
16	august	member	13.722275	440085

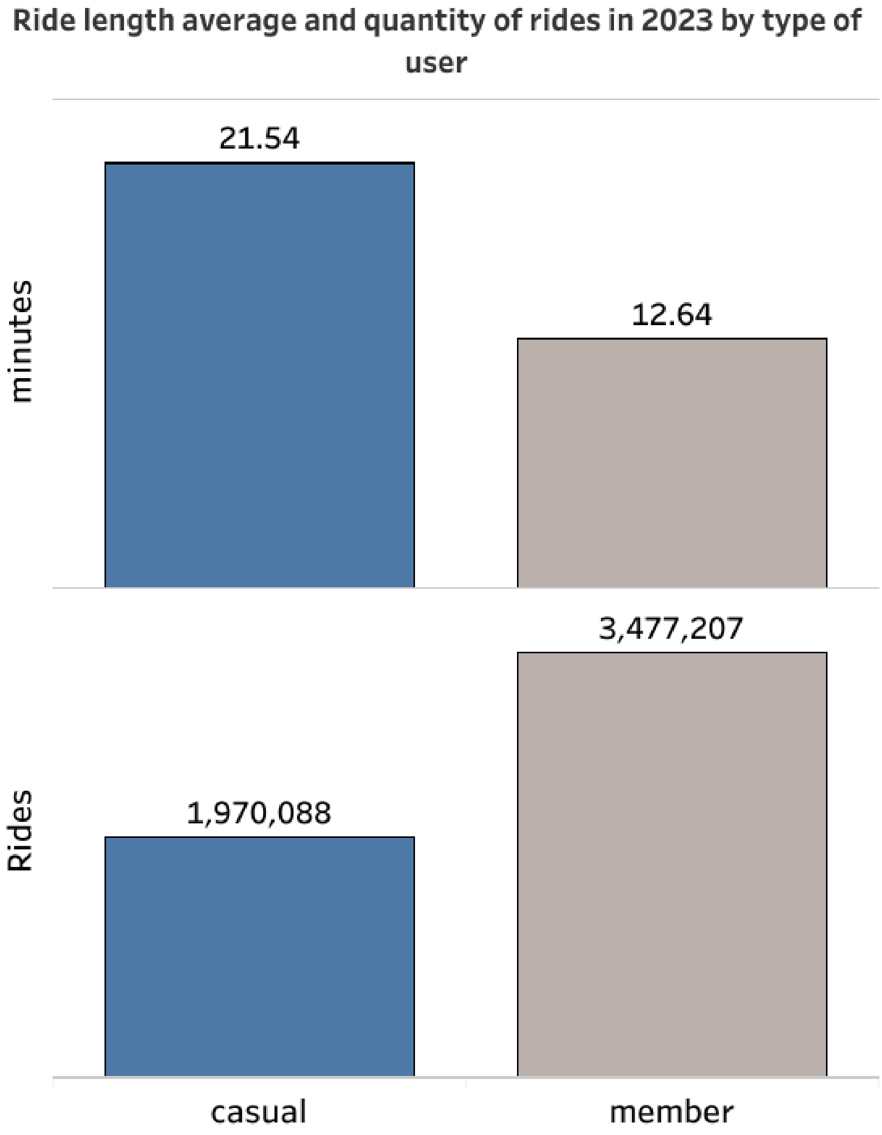
Once again, casual users show the longest rides and member users with the highest number of rides. While certain patterns have been identified, it will be the graphs that help to see the complete picture, and supported with the context of the data itself, will give clarity to the pending decisions to be made.

Results

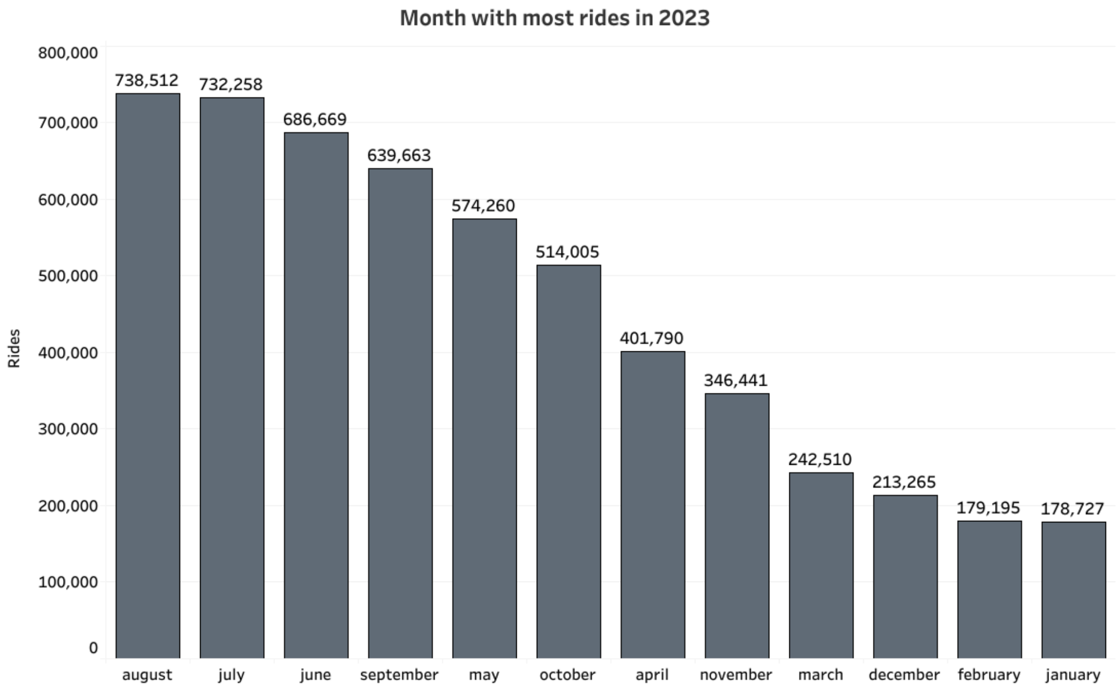
To clearly see the relationship of rides between both users, the ring graph shows the percentage of rides made by each one throughout the year 2023. 63.83% of the total rides were made by member users. This confirms that this type of user makes more than half of the rides in the Cyclistic program.



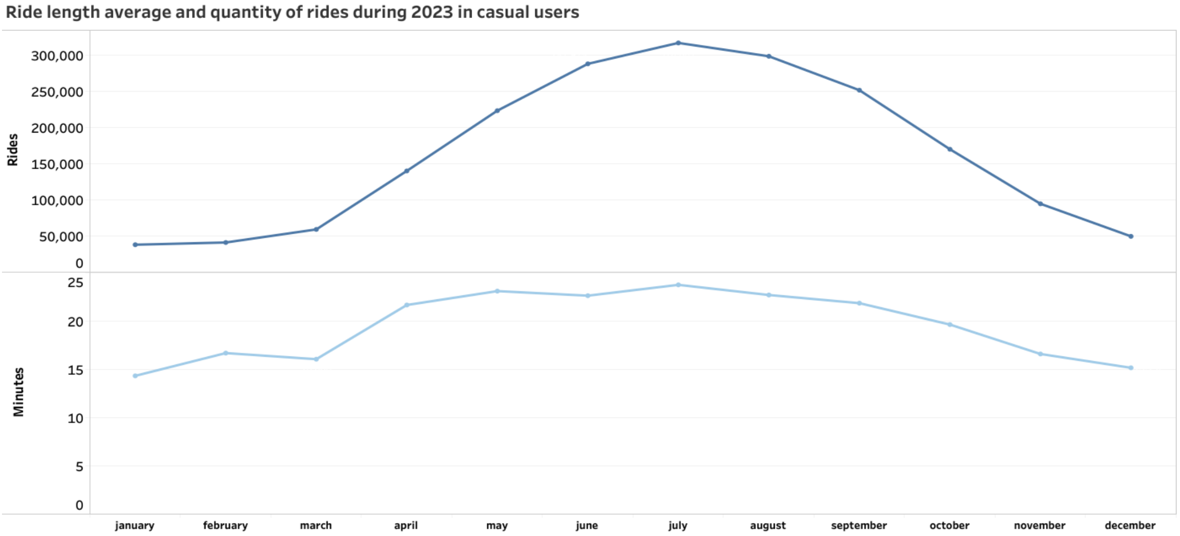
The first differences between user types are now visible in the following bar charts. In the upper bar chart, it is observed that member users have, on average, fewer minutes per ride than casual users, meaning they take shorter rides. However, the lower bar chart shows that member users take more rides than casual users.



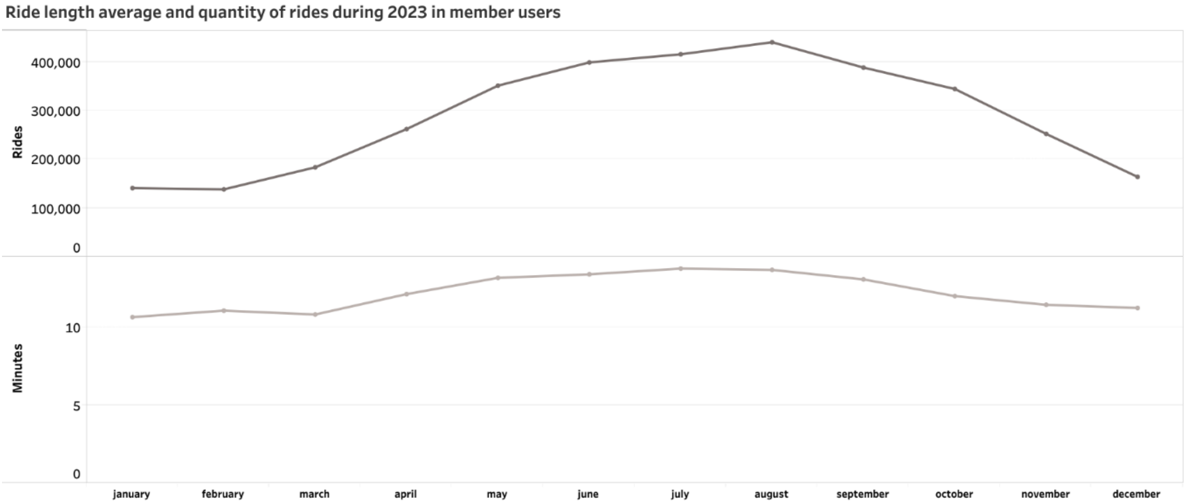
Next, it is observed that August is the month with the most rides followed by July and June, coincidentally with the summer months in Chicago.



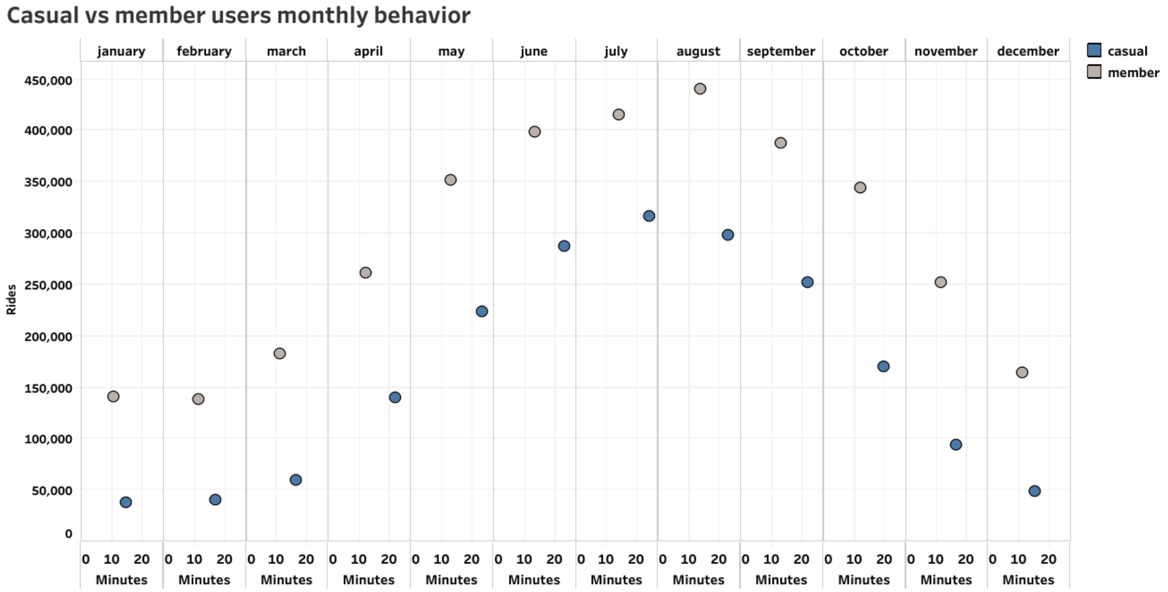
Taking into account the separate behavior of each user, it can be seen that casual users indeed increase the number of rides as the summer in Chicago approaches and decrease their activity in the colder times. Although the length of their rides follows this same pattern, it is also possible to say that it takes some stability in the spring and summer seasons.



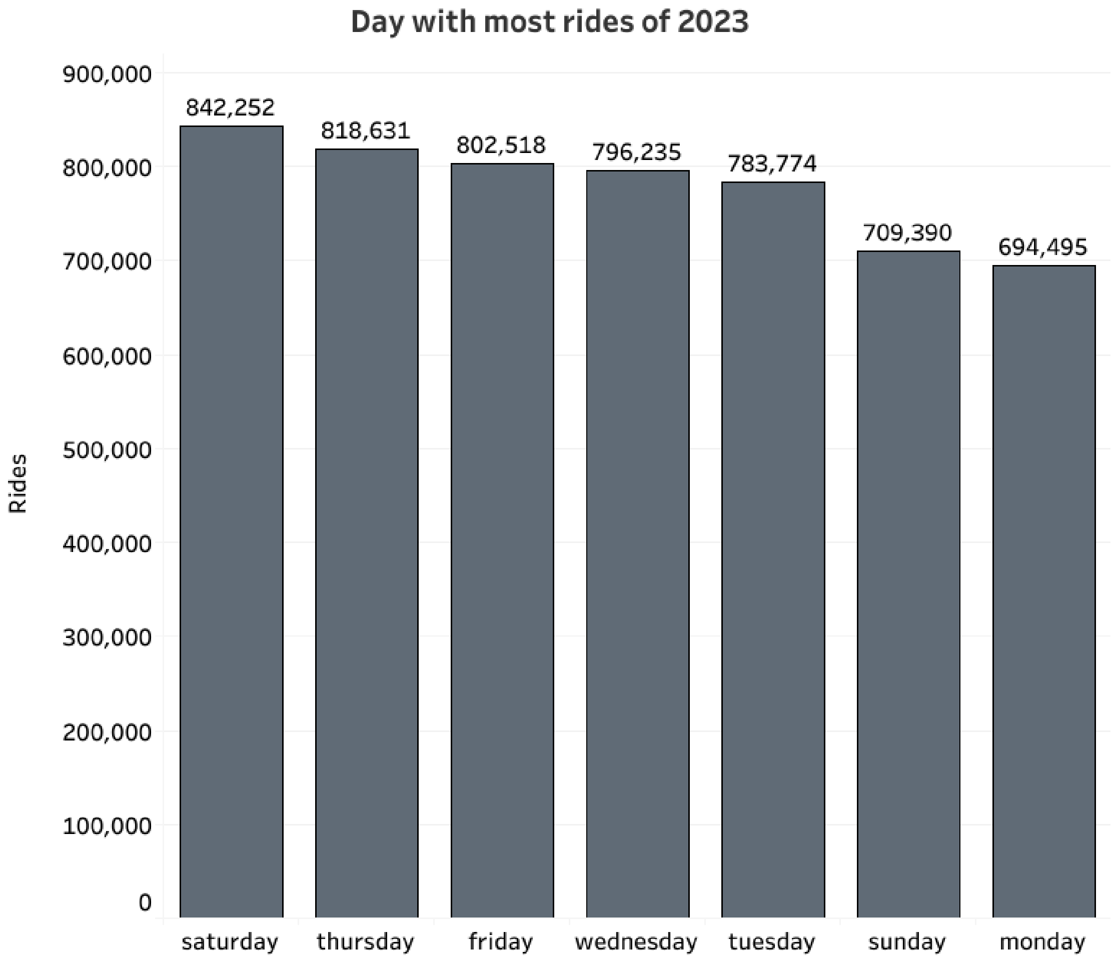
For member users, the number of rides made is higher than that of casual users, but their average ride duration is lower. It is also possible to affirm that their behavior follows the same pattern as casual users, which is an increase in rides in warmer seasons and consistent duration during that time period.



In the following image, the behavior of both types of users is observed simultaneously, considering their average ride duration per month as well as the number of rides made. For example, member users made more rides in July, so their position is above casual users. However, casual users made longer rides, so their position is more oriented to the right side.

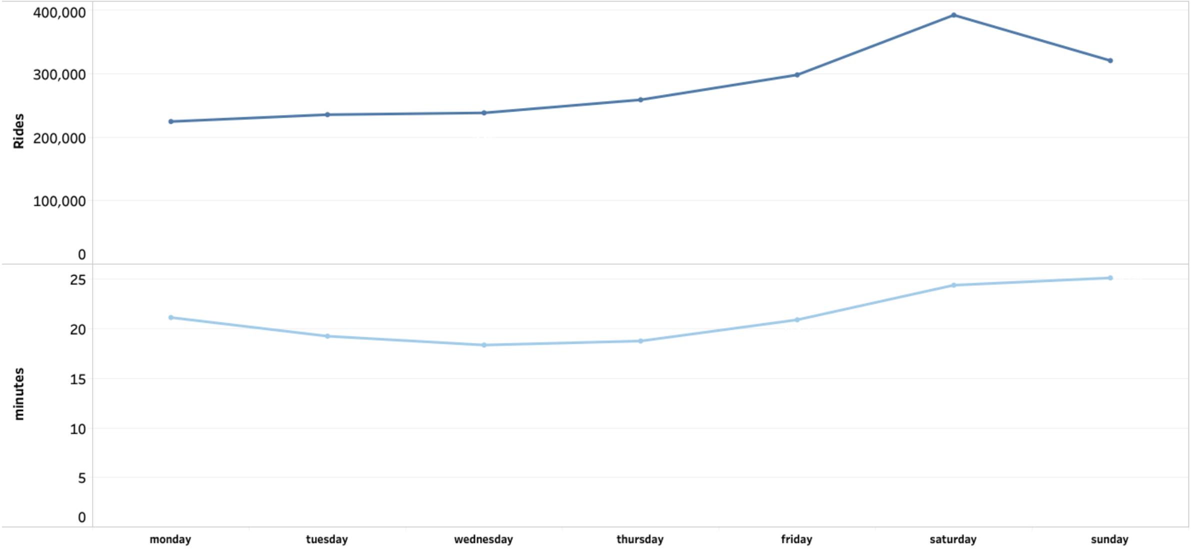


To have a weekly perspective, the day of the week with the most rides in 2023 is Saturday, followed by Thursday and Friday. This indicates that users increase the use of the service as Saturday approaches.



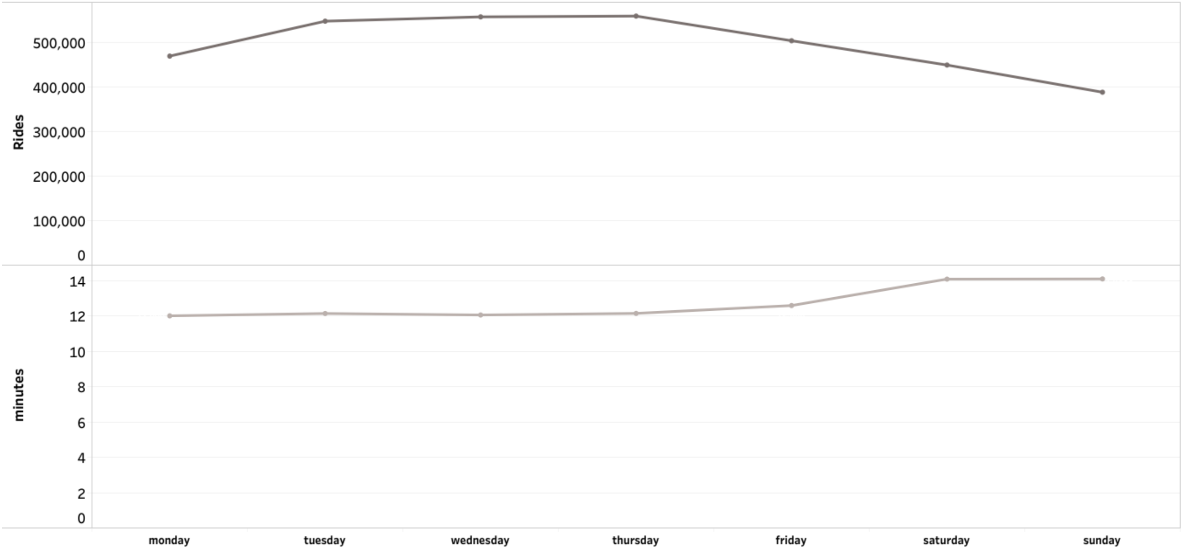
Separately, the behavior of casual users shows that the number of rides, as well as the average duration, increases as the weekend approaches. This also suggests that casual users may prefer to use the service recreationally.

Ride length average and quantity of rides during the week in casual users

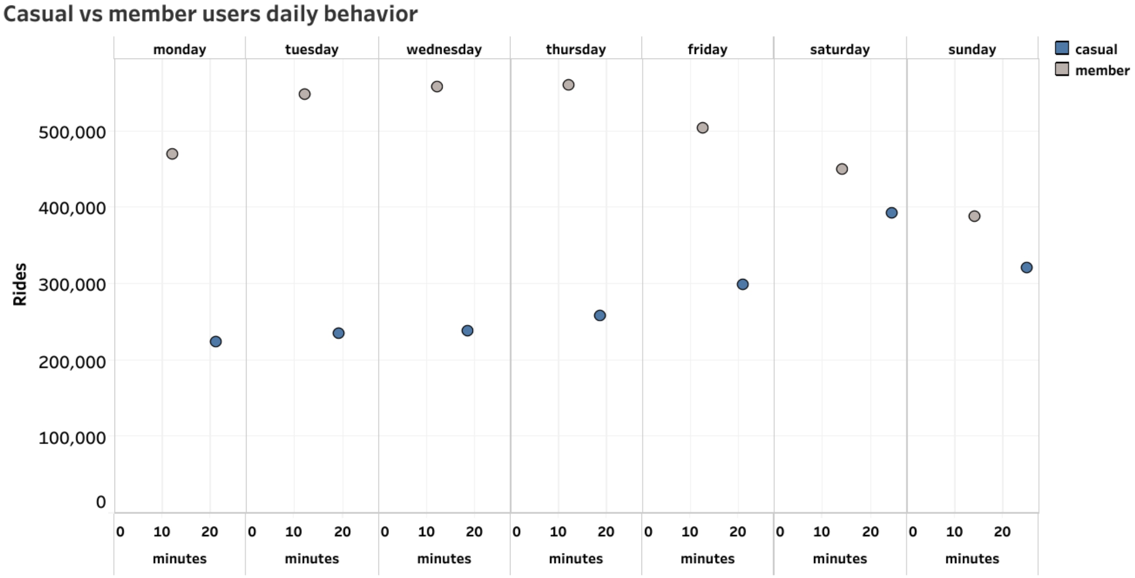


Member users decrease their rides as the weekend approaches; however, the duration of these rides increases. possibly indicating that users transition from using the service to cover daily needs to using it recreationally.

Ride length average and quantity of rides during the week in member users



Once again, the behavior of the two types of users is observed considering their average ride duration and the number of rides made, now shown on a weekly basis. It is interesting to note that Sunday, being the day with the least activity for member users, is comparable to Saturday, which is the day with the most rides for casual users.



Conclusions

Looking at the data and addressing the question, “How do annual members and casual riders differ in their use of Cyclistic bikes?” The answer is as follows:

- There is an average ride duration of 15 minutes overall, suggesting that the destinations for both types of users may not be far from their starting point.
- Of all the rides made in the Cyclistic program in 2023, more than half of these rides are made by members, confirming that these users are the most frequent users of the program.
- Casual users make fewer rides, but they are long rides, and member users make more rides, but they are short rides.
- The months with the most rides are June, July, and August, indicating that users prefer to use the program during warmer months.
- Both casual users and members increase the number and duration of their rides in warmer months. However, during these periods, the duration of their rides shows a period of stability by neither increasing nor decreasing significantly.
- The days with the most rides are Thursday, Friday, and Saturday, indicating that users prefer to use the service as the weekend approaches.
- Casual users notably increase both the number and duration of their rides on Thursdays, Fridays, and Saturdays, indicating a preference for recreational use of the service. Member users decrease the number of their rides on these days, and the duration of rides changes from being stable during the week to increasing on the mentioned days. This suggests that members may use the service for daily needs during the week and switch to recreational use on weekends.

Recommendations

The following actions take into account the points seen in the conclusions and aim to fulfill the business objective of designing a marketing program with strategies aimed at converting casual riders into annual members.

These actions involve creating promotions, discounts, and exclusive benefits for both types of users. All of this will serve as a hook for casual users to persuade them to upgrade their subscription.

1. Establish agreements with selected businesses within a 15-minute radius of each Cyclistic station. These agreements should offer slight benefits to member users. This will create a symbiotic relationship between member users and their preferred local businesses, reinforcing the idea of conversion for casual users.
2. Highlight the cost-benefit ratio of an annual subscription to casual users through digital advertising via email or social media. Additionally, create loyalty programs and incentives for member users to encourage continued service usage.
3. Since casual users tend to take longer rides,incentives can be created for them to reach a certain mileage threshold to earn their first membership discount.
4. Launch generous membership change promotions and discounts to casual users during peak rides seasons. By targeting promotions during periods of high program activity, more casual users will be aware of these offers and may consider making the change.
5. During the summer months when rides volume increases and rides duration stabilizes, it is possible to offer member users exclusive rewards or contests for consistency by reaching a specified number of rides or distances. This incentivizes casual users to consider a membership to access these benefits.
6. Increase the frequency of notifications about the benefits of annual membership and upcoming promotions on days when casual users increase their rides volume. This ensures more users are informed and creates a sense of urgency if promotions are already in effect.
7. Utilize printed and digital advertising to promote various city landmarks strategically selected to serve as tourist attractions and meeting points for both types of users, especially on Saturdays and Sundays when longer rides are common. This encourages casual users to spend more time exploring the service and interacting with member users, creating a sense of community and potentially motivating them to convert.