Cyclistic Bikeshare Data Analysis: Section One

Twelve Months of Chicago Ridership

Patrick Holland-Stergar

16 August 2023

Overview

This is a record of all the work I did to complete a case study of the “Cyclistic Bike Share” as provided by Google and Coursera in Course 8 of the Google Data Analytics Certificate. The details are provided in the document linked here. The following excerpt provides the scenario and instructions.

Scenario

You are a junior data analyst working in the marketing analyst team at Cyclistic, a bike-share company in Chicago. The director of marketing believes the company’s future success depends on maximizing the number of annual memberships. Therefore, your team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, your team will design a new marketing strategy to convert casual riders into annual members. But first, Cyclistic executives must approve your recommendations, so they must be backed up with compelling data insights and professional data visualizations.

Characters and teams

● *Cyclistic: A bike-share program that features more than 5,800 bicycles and 600 docking stations. Cyclistic sets itself apart by also offering reclining bikes, hand tricycles, and cargo bikes, making bike-share more inclusive to people with disabilities and riders who can’t use a standard two-wheeled bike. The majority of riders opt for traditional bikes; about 8% of riders use the assistive options. Cyclistic users are more likely to ride for leisure, but about 30% use them to commute to work each day.*
● *Lily Moreno: The director of marketing and your manager. Moreno is responsible for the development of campaigns and initiatives to promote the bike-share program. These may include email, social media, and other channels*
. ● *Cyclistic marketing analytics team: A team of data analysts who are responsible for collecting, analyzing, and reporting data that helps guide Cyclistic marketing strategy. You joined this team six*

months ago and have been busy learning about Cyclistic's mission and business goals — as well as how you, as a junior data analyst, can help Cyclistic achieve them.
● Cyclistic executive team: The notoriously detail-oriented executive team will decide whether to approve the recommended marketing program.

About the company

In 2016, Cyclistic launched a successful bike-share offering. Since then, the program has grown to a fleet of 5,824 bicycles that are geotracked and locked into a network of 692 stations across Chicago. The bikes can be unlocked from one station and returned to any other station in the system anytime. Until now, Cyclistic's marketing strategy relied on building general awareness and appealing to broad consumer segments. One approach that helped make these things possible was the flexibility of its pricing plans: single-ride passes, full-day passes, and annual memberships. Customers who purchase single-ride or full-day passes are referred to as casual riders. Customers who purchase annual memberships are Cyclistic members. Cyclistic's finance analysts have concluded that annual members are much more profitable than casual riders. Although the pricing flexibility helps Cyclistic attract more customers, Moreno believes that maximizing the number of annual members will be key to future growth. Rather than creating a marketing campaign that targets all-new customers, Moreno believes there is a very good chance to convert casual riders into members. She notes that casual riders are already aware of the Cyclistic program and have chosen Cyclistic for their mobility needs. Moreno has set a clear goal: Design marketing strategies aimed at converting casual riders into annual members. In order to do that, however, the marketing analyst team needs to better understand how annual members and casual riders differ, why casual riders would buy a membership, and how digital media could affect their marketing tactics. Moreno and her team are interested in analyzing the Cyclistic historical bike trip data to identify trends.

Ask

Three questions will guide the future marketing program:

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

2. Why would casual riders buy Cyclistic annual memberships?

3. How can Cyclistic use digital media to influence casual riders to become members?

Specific Task

Moreno has assigned you the first question to answer: How do annual members and casual riders use Cyclistic bikes differently? You will produce a report with the following deliverables:

1. A clear statement of the business task

2. A description of all data sources used

3. Documentation of any cleaning or manipulation of data

4. A summary of your analysis

5. Supporting visualizations and key findings

6. Your top three recommendations based on your analysis

Business Task

The business task at hand is to perform analysis of the last 12 months of ridership data from Cyclistic in order to understand the behaviours of riders, specifically patterns in how casual riders use Cyclistic bicycles in comparison to how members use the bicycles. The business will then use this analysis to inform a marketing approach to convert more casual riders to memberships.

Data Sources

The data used in this case study was collected directly by Cyclistic as part of normal business operations, therefore it is first-party data. The data format for the period in question, i.e. July 2022 through June 2023 is that data for each month is collected and stored as a .zip file on an AWS server (https://divvy-tripdata.s3.amazonaws.com/index.html). When unzipped, each file is in .csv format and each utilizes the same columns.

Privacy and License

Cyclistic is a fictional company invented by Google for the purposes of this case study, but the ridership data is taken from actual bicycle rides and according to the prompt document:

> For the purposes of this case study, the datasets are appropriate and will enable you to answer the business questions. The data has been made available by Motivate International Inc. under this license.) This is public data that you can use to explore how different customer types are using Cyclistic bikes. But note that data-privacy issues prohibit you from using riders' personally identifiable information. This means that you won't be able to connect pass purchases to credit card numbers to determine if casual riders live in the Cyclistic service area or if they have purchased multiple single passes.

Data Analysis Process

Tools used:

Because I learned how to use BigQuery in my Google Data Analytics Certificate course, I chose to clean and manipulate data in BigQuery with the goal of then doing visualizations in Tableau. I later came to discover that RStudio was a better option for doing my visualizations.

Throughout the course of the analysis, I utilized the following tools:

- RStudio Desktop (version 2023.06.1 Build 524)
- Google BigQuery, specifically the SQL Workspace
- Google Cloud Storage (as this is the main method for exporting data from the BigQuery SQL Workspace)
- Google Drive (for upload of files after downloading them from the webpage on which they are hosted)
- Tableau Desktop (I attempted to use this program for visualizations, but was encountering issues and therefore switched to using RStudio Desktop, therefore no analysis or visualizations were done via Tableau)

Import of Data

I downloaded data from July 2022 through June 2023 (so 12 months of user data) from the data source link provided in the guidelines document. I then copied the .csv files for each of those 12 months to a single centralized folder on my local drive, then uploaded those 12 files to Google Drive. From Google Drive, I uploaded each month's data to BigQuery under the project "cyclistic-rider-analysis" and the dataset "last_decade_rider_data".

At this point, the data for each month was located in a separate table within BigQuery. To consolidate the data into a single table called "12_months_data" so that I could perform analysis across the data from all 12 months in question, I provided BigQuery the following instructions:

```
INSERT INTO `cyclistic-rider-analysis.Last_decade_rider_data.12_months_data_new`

SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2022_07`
UNION ALL
SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2022_08`
UNION ALL
SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2022_09`
UNION ALL
SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2022_10`
UNION ALL
SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2022_11`
```

```
UNION ALL
SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2022_12`
UNION ALL
SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2023_01`
UNION ALL
SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2023_02`
UNION ALL
SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2023_03`
UNION ALL
SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2023_04`
UNION ALL
SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2023_05`
UNION ALL
SELECT *
FROM `cyclistic-rider-analysis.Last_decade_rider_data.2023_06`
;
```

This resulted in a table with 5,779,444 rows and featuring a schema of 13 columns.

Data Notes

- The data represents all the rides that *began* from 00:00 on July 1st, 2022 to 23:59 on June 30th, 2023. Rides that began before 23:59 on June 30th, 2023 but ended after 00:00 on July 1st, 2023, are shown in this dataset - the maximum value for the column "ended_at" is 2023-07-10 20:26:44 UTC (that ride is shown because it began at 2023-06-29 18:34:56 UTC).
  - I checked this with (multiple versions of) the query:

```
SELECT *

FROM `cyclistic-rider-analysis.Last_decade_rider_data.12_months_data_new`

WHERE started_at = (SELECT MIN(started_at) FROM
`cyclistic-rider-analysis.Last_decade_rider_data.12_months_data_new`)
```

Where I replaced started_at with ended_at and MIN() with MAX() as needed.

Data Examination and Cleaning

1. I wanted to see how many rides had no information about the starting station, ending station, so I performed the following queries:

   ```
   SELECT COUNT(ride_id) AS number_of_rides_with_unknown_start_station

   FROM `cyclistic-rider-analysis.Last_decade_rider_data.12_months_data_new`

   WHERE start_station_name IS NULL
   ```

   This yielded an answer of 857,860 rides where the start_station_name was NULL

AND

```
SELECT COUNT(ride_id) AS number_of_rides_with_unknown_end_station


FROM `cyclistic-rider-analysis.Last_decade_rider_data.12_months_data_new`


WHERE end_station_name IS NULL
```

This yielded an answer of 915,655 rides where the end_station_name was NULL

I then realized that I could combine the queries to determine how many rows had a value of NULL in the ride_id, start_station_name, end_station_name, start_station_id, or end_station_id column.

That query is:

```
SELECT COUNT(ride_id) AS number_of_rides_with_unknown_start_station


FROM `cyclistic-rider-analysis.Last_decade_rider_data.12_months_data_new`


WHERE end_station_name IS NULL OR start_station_name IS NULL OR start_station_id IS
NULL OR end_station_ID IS NULL or ride_id IS NULL
```

And running the query showed that 1370345 rides had a NULL value in one of those columns.

The ratio of those rows with a NULL value to the overall number of rows in the data table is 1370345 /5779444 or 23.7%

I also checked to see how many rides were under 60 seconds, because the company stated "the data has been processed to remove trips that are taken by staff as they service and inspect the system; and any trips that were below 60 seconds in length (potentially false starts or users trying to re-dock a bike to ensure it was secure)."

I used the following script:

```
SELECT COUNT(TIMESTAMP_DIFF(TIMESTAMP(ended_at), TIMESTAMP(started_at), SECOND)) AS
ride_duration



FROM `cyclistic-rider-analysis.Last_decade_rider_data.12_months_data_new`

WHERE TIMESTAMP_DIFF(TIMESTAMP(ended_at), TIMESTAMP(started_at), SECOND) < 60
```

And found that 149372 rides were below 60 seconds.

I wanted to clean the data to have a table containing only rows that had values (i.e. were NOT NULL) for starting station, starting id, ending station, and ending id and only had rides that were longer than 60 seconds. I used the following query to achieve this in a new table called "filtered_12_months"

```
CREATE TABLE cyclistic-rider-analysis.Last_decade_rider_data.filtered_12_months AS

SELECT *

FROM `cyclistic-rider-analysis.Last_decade_rider_data.12_months_data_new`

WHERE (TIMESTAMP_DIFF(TIMESTAMP(ended_at), TIMESTAMP(started_at), SECOND) > 60) AND
(end_station_name IS NOT NULL AND start_station_name IS NOT NULL AND start_station_id
IS NOT NULL AND end_station_ID IS NOT NULL AND ride_id IS NOT NULL)
```

Then, I wanted to see if there were any rides where the starting or ending latitude were outside the area of Chicago, so I took the following approach:

```
-- CHECK FOR LATITUDE AND LONGITUDE
-- SHOULD BE near Chicago so approximately (41.881832, -87.623177)
-- I got those coordinates from https://www.latlong.net/place/chicago-il-usa-1855.html

SELECT MIN(start_lat) AS min_start_lat, MIN(end_lat) AS min_end_lat, MAX(start_lat) AS
max_start_lat, MAX(end_lat) AS max_end_lat

FROM `Last_decade_rider_data.filtered_12_months`
```

This resulted in a table with 4317262 rows

From this query I saw that while the MIN and MAX for starting latitude were 41.648500762664092 and 42.064856333333331, which are within the expected range, the MIN for ending latitude was 0.00, which was invalid. The MAX for ending latitude was 42.064854, which was within the expected range.

41 degrees Latitude is about 30 miles south of the southernmost edge of the Chicago metro area, so I used it as a reference, in the sense that I decided to filter out any rides that recorded an ending latitude less than 41.00

Since I also wanted to ensure that I had the member type information for every ride, I also filtered to ensure that every value in that column was either “member” or “casual”.

I used the following script for that filtering:

```
CREATE TABLE cyclistic-rider-analysis.Last_decade_rider_data.new_filtered_12_months AS

SELECT *

FROM `Last_decade_rider_data.filtered_12_months`

WHERE (end_lat > 41.00) AND (member_casual= "member" OR member_casual = "casual")
```

That resulted in a table with 4317244 rows.

Next, I wanted to check if any rides had durations that were unrealistic. Since the company said that the longest possible duration for a casual rider using a bike is 24 hours, I wanted to filter out any rides by casual riders with a duration longer than that. I used the following query to do so.

```
CREATE TABLE cyclistic-rider-analysis.Last_decade_rider_data.new1_filtered_12_months
AS

SELECT *
```

```
FROM `Last_decade_rider_data.new_filtered_12_months`

WHERE (TIMESTAMP_DIFF(TIMESTAMP(ended_at), TIMESTAMP(started_at), SECOND) < 86400)
```

This created a new table called "new1_filtered_12_months" with 4317139 rows.

Now that the data has been cleaned, we can see what transformations are required to answer relevant questions pertinent to the business task. Those questions are:

1. **How long do customers use the bikes?**
   a. Information needed: the difference between the ride's end datetime and start time.
2. **How far do the customers ride?**
   a. Information needed: the ride distance (meters), i.e. how far are the start and end points from each other.
3. **In what time of the day do the customers use the bikes most?**
   a. Information needed: which the hour at which the trip started.
4. **What days of the week do the customers use bikes more often?**
   a. Information needed: which day of the week the ride started.
5. **What months do the customers use bikes more often?**
   a. Information needed: which month the trip started.

For how far customers' rides are, I used the geography package in the following queries:

```
ALTER TABLE Last_decade_rider_data.new1_filtered_12_months
ADD COLUMN GeoLocationStart GEOGRAPHY, ADD COLUMN GeoLocationEnd GEOGRAPHY;

-- This inputs the lat and long for starting and ending position in the single columns
GeoLocation Start and GeoLocationEnd

UPDATE `Last_decade_rider_data.new1_filtered_12_months`

SET GeoLocationStart = ST_GEOGPOINT(start_lng,start_lat), GeoLocationEnd =
ST_GEOGPOINT(end_lng, end_lat)

WHERE true

-- This adds a distance_travelled_column
```

```sql
ALTER TABLE Last_decade_rider_data.new1_filtered_12_months
ADD COLUMN Distance_travelled2 FLOAT64

-- This calculates how far the bike ride was

UPDATE `Last_decade_rider_data.new1_filtered_12_months`

SET Distance_travelled2 = ST_DISTANCE(GeoLocationStart,GeoLocationEnd)

WHERE true
```

Data note: I realized that even though I had filtered out rides that lasted less than 60 seconds, there were rides that lasted more than 60 seconds but had a distance traveled of 0.0, meaning they began and ended at the same place. I am interpreting that some of these rides are legitimate - they would be ones where someone is using the bike for exercise or recreation and wants to begin and end in the same place, whereas some are not legitimate rides. To filter out the "illegitimate rides", I decided that a recreational ride would not last less than 5 minutes (i.e. 300 seconds).

I first used the following query to check how many records met this criteria.

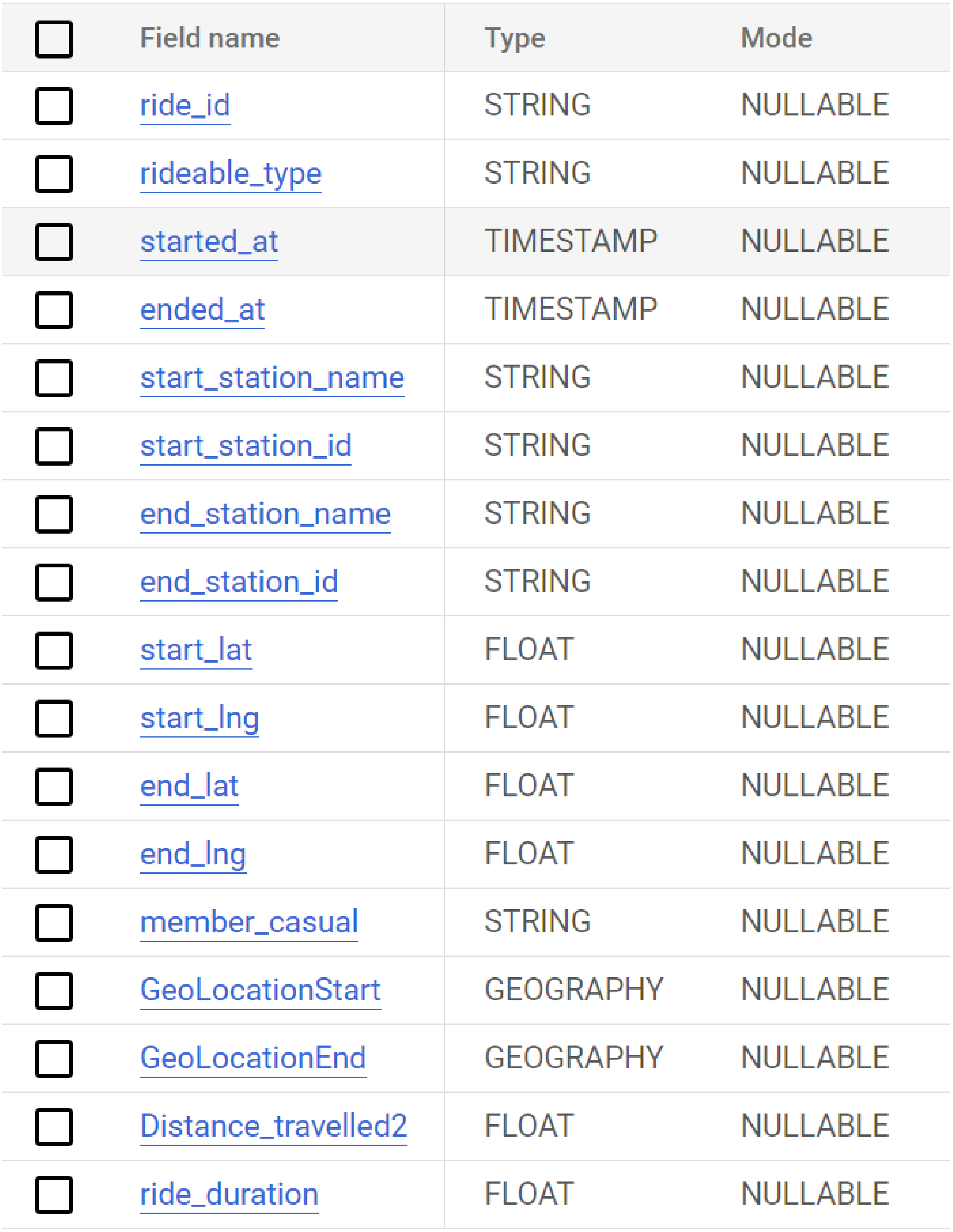
```sql
SELECT COUNT(*)
FROM `Last_decade_rider_data.new1_filtered_12_months`
WHERE Distance_travelled2 = 0.0 AND ride_duration < 300
```

Which showed that 27053 rows met my criteria for "illegitimate" rides that started and finished in the same spot.

I then used the following query to delete those rows from the table.

```sql
DELETE FROM `Last_decade_rider_data.new1_filtered_12_months`
WHERE Distance_travelled2 = 0.0 AND ride_duration < 300
```

This left a table with a total of 4290086 rows with the following schema.



☐	Field name	Type	Mode
☐	ride_id	STRING	NULLABLE
☐	rideable_type	STRING	NULLABLE
☐	started_at	TIMESTAMP	NULLABLE
☐	ended_at	TIMESTAMP	NULLABLE
☐	start_station_name	STRING	NULLABLE
☐	start_station_id	STRING	NULLABLE
☐	end_station_name	STRING	NULLABLE
☐	end_station_id	STRING	NULLABLE
☐	start_lat	FLOAT	NULLABLE
☐	start_lng	FLOAT	NULLABLE
☐	end_lat	FLOAT	NULLABLE
☐	end_lng	FLOAT	NULLABLE
☐	member_casual	STRING	NULLABLE
☐	GeoLocationStart	GEOGRAPHY	NULLABLE
☐	GeoLocationEnd	GEOGRAPHY	NULLABLE
☐	Distance_travelled2	FLOAT	NULLABLE
☐	ride_duration	FLOAT	NULLABLE

I could now return to continue to answer the aforementioned questions. We had already done the calculations that will allow us to answer questions 1 and 2, so it was time to turn to question 3 “In what time of the day do the customers use the bikes most?”

The following code created and calculated for each row a column startHour that shows the starting hour of the ride as a value from 0 to 23, where all rides are rounded down (i.e. a ride beginning at 19:47 will have a value of 19 for this column).

```
ALTER TABLE Last_decade_rider_data.new1_filtered_12_months
ADD COLUMN startHour INT64;

UPDATE `Last_decade_rider_data.new1_filtered_12_months`

SET startHour = EXTRACT(HOUR FROM started_at)

WHERE true
```

Now, I had the data needed to answer question 3, so looking at question 4 “What days of the week do the customers use bikes more often?” I realized I needed to similarly create a column showing the day of the week when the ride began.

I used the following query to do this.

```
ALTER TABLE Last_decade_rider_data.new1_filtered_12_months
ADD COLUMN startDay INT64;

UPDATE `Last_decade_rider_data.new1_filtered_12_months`

SET startDay = EXTRACT(DAYOFWEEK FROM started_at)

WHERE true
```

This resulted in a new column startDay which shows the day the ride started as an integer from 1 to 7, with one being Sunday.

Finally, I needed to calculate the data to answer question 5 “What months do the customers use bikes more often?”

I accomplished this with the following query.

```
ALTER TABLE Last_decade_rider_data.new1_filtered_12_months
ADD COLUMN startMonth INT64;

UPDATE `Last_decade_rider_data.new1_filtered_12_months`

SET startMonth = EXTRACT(MONTH FROM started_at)

WHERE true
```

This resulted in a new column startMonth which shows the starting month as an integer from 0 to 12.

Finally, I may want to have a column where day/month/year are all in the same column (but no time of day), so I will add one more column.

Analysis Notes: a mistake in my process and its correction

At first, I created the column for the date as an INT64 data type, but then when BigQuery tried to write the result of extracting the DATE from the starting time to this column, it would not work since DATE is a DATE type and cannot go into a INT64 type column.

To correct this I had to drop the startDATE column I had created and write that result to a table with a new name (BigQuery does not provide a direct way to drop a column from a table).

So first here is the query with the error. startDate should be specified as a DATE type, not an INT64 type.

```
ALTER TABLE Last_decade_rider_data.new1_filtered_12_months
ADD COLUMN startDate INT64;

UPDATE `Last_decade_rider_data.new1_filtered_12_months`

SET startDate = EXTRACT(DATE FROM started_at)

WHERE true
```

To correct this I needed to drop the startDate column I had created and save the resulting table as a new table called new2_filtered_12_months

I did this with the following query, then used the “save results” button in BigQuery to save the table.

```
select *
except(startDate)
from `Last_decade_rider_data.new1_filtered_12_months`
```

Then, I used the following to calculate and fill the starting date for each ride.

```
ALTER TABLE Last_decade_rider_data.new1_filtered_12_months
ADD COLUMN startDate DATE;

UPDATE `Last_decade_rider_data.new2_filtered_12_months`


SET startDate = EXTRACT(DATE FROM started_at)


WHERE true
```

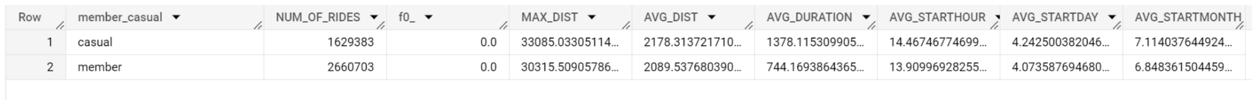
This gave me a column startDate which showed the date (i.e. year, month, and day) for each ride.

I wanted to see a few summary statistics using SQL before I started investigating the data using Tableau, so I ran the following query.

```
SELECT
 member_casual,
 COUNT(ride_id) AS NUM_OF_RIDES,
 MIN(Distance_travelled2),
 MAX(Distance_travelled2) AS MAX_DIST,
 AVG(Distance_travelled2) AS AVG_DIST,
 AVG(ride_duration) AS AVG_DURATION,
 AVG(startHour) AS AVG_STARTHOUR,
 AVG(startDay) AS AVG_STARTDAY,
 AVG(startMonth) AS AVG_STARTMONTH


from `Last_decade_rider_data.new2_filtered_12_months`
group by member_casual;
```

Which resulted in:



Row	member_casual	NUM_OF_RIDES	f0_	MAX_DIST	AVG_DIST	AVG_DURATION	AVG_STARTHOUR	AVG_STARTDAY	AVG_STARTMONTH
1	casual	1629383	0.0	33085.03305114...	2178.313721710...	1378.115309905...	14.46746774699...	4.242500382046...	7.114037644924...
2	member	2660703	0.0	30315.50905786...	2089.537680390...	744.1693864365...	13.90996928255...	4.073587694680...	6.848361504459...

From this, I could see a few major differences that stood out to me. First, that there were roughly 100,000 more rides taken by members than by casual riders in the past year. Second, the average duration for a ride by a member was nearly 2x that of a ride by a casual user.

Data Export for Visualization

I desired to use a visualization tool, in my case Tableau, to help me further.

There are limited options for exporting data from BigQuery - the only way to directly export data is to export it to Google Cloud Storage, which I proceeded to do.

Since my table was over 1GB in size, I had to break it into smaller files using a wildcard designator in my query. The export query was:

```
EXPORT DATA OPTIONS(
  uri='gs://chicago_bike_riders_analysis_phs/analysis123-*.csv',
  format='CSV',
  overwrite=true,
  header=true,
  field_delimiter=',') AS
SELECT * from `Last_decade_rider_data.new2_filtered_12_months` WHERE true
```

Where the wildcard is the “-*” at the end of the uri

That exported my table as 23 csv files in my google cloud storage bucket.

To consolidate them all back into a single file, I used the gsutil tool in the google command shell and used the following script.

```
 gsutil compose gs://chicago_bike_riders_analysis_phs/analysis123-*
gs://chicago_bike_riders_analysis_phs/analysis123-composite
```

I then downloaded the combined .csv file

Data Visualizations In RStudio

Note: I attempted to utilize Tableau Desktop for visualizations, but was encountering many errors when working with that program. For reasons of time and convenience, I pivoted to using RStudio desktop.

In RStudio, I first loaded the tidyverse and data.table packages to make it easier to work with the data.

Then, I uploaded the data using fread into a data frame called data1:

```
> data1 <- fread('C:\\Users\\pholl\\OneDrive\\Desktop\\bikedata.csv')
```

And then used glimpse() to refresh my memory about the data set.

```
> glimpse(data1)
```

Resulting in:

```
Rows: 4,290,108
Columns: 21
$ ride_id              <chr> "D3D9A8C793F90BED", "BEC831D7D0474E2A",
"A0F4E9CF47049D6B", "992B5BABE1651E54", "2635FAFEE21CE52B", "15A5040EAF7D2BA1",
"DE9D3DD26D4…
$ rideable_type        <chr> "electric_bike", "electric_bike", "electric_bike",
"electric_bike", "classic_bike", "electric_bike", "classic_bike",
"classic_bike",…
$ started_at           <chr> "2023-04-27 08:02:30 UTC", "2022-07-31 14:26:58
UTC", "2022-07-23 13:40:23 UTC", "2023-06-28 07:09:05 UTC", "2022-07-16
13:43:20 UTC…
$ ended_at             <chr> "2023-04-27 08:21:25 UTC", "2022-07-31 14:36:24
UTC", "2022-07-23 13:55:50 UTC", "2023-06-28 07:21:13 UTC", "2022-07-16
13:52:52 UTC…
$ start_station_name   <chr> "410", "Smith Park", "Smith Park", "Smith Park",
"Smith Park", "Smith Park", "Smith Park", "Smith Park", "Smith Park", "Smith
Park",…
$ start_station_id     <chr> "410", "643", "643", "643", "643", "643", "643",
"643", "643", "643", "643", "643", "643", "643", "643", "643", "643", "643",
"643",…
```

```
$ end_station_name    <chr> "LaSalle St & Washington St", "Damen Ave & Walnut
(Lake) St", "Western Ave & Division St", "Sangamon St & Washington Blvd",
"Ashland…
$ end_station_id      <chr> "13006", "KA17018054", "13241", "13409", "13073",
"13073", "13073", "13028", "20246.0", "13022", "428", "TA1308000005", "13157",
"13…
$ start_lat           <chr> "41.9", "41.892028666666668", "41.892012666666666",
"41.891957045", "41.892048", "41.891999841", "41.892048", "41.892048",
"41.89203…
$ start_lng           <chr> "-87.69", "-87.689388333333326", "-87.689362",
"-87.6893363", "-87.689397", "-87.689410448", "-87.689397", "-87.689397",
"-87.689417…
$ end_lat             <chr> "41.882664", "41.885951", "41.902893", "41.883165",
"41.88592", "41.88592", "41.88592", "41.874754", "41.89", "41.892278", "41.92",
…
$ end_lng             <chr> "-87.63253", "-87.677009", "-87.687275",
"-87.6511", "-87.66717", "-87.66717", "-87.66717", "-87.649807", "-87.65",
"-87.612043", "-…
$ member_casual       <chr> "member", "casual", "casual", "casual", "member",
"member", "member", "member", "casual", "member", "casual", "casual", "member",
"m…
$ GeoLocationStart    <chr> "POINT(-87.69 41.9)", "POINT(-87.6893883333333
41.8920286666667)", "POINT(-87.689362 41.8920126666667)", "POINT(-87.6893363
41.89195…
$ GeoLocationEnd      <chr> "POINT(-87.63253 41.882664)", "POINT(-87.677009
41.885951)", "POINT(-87.687275 41.902893)", "POINT(-87.6511 41.883165)",
"POINT(-87.…
$ Distance_travelled2 <chr> "5132.8122999977932", "1227.5183306807753",
"1222.1086762813716", "3312.73929379445", "1962.0329148366427",
"1961.2251544375015", "1…
$ ride_duration       <chr> "1135", "566", "927", "728", "572", "570", "922",
"1118", "1443", "2072", "1043", "1172", "606", "1101", "1411", "587", "573",
"1082…
$ startHour           <chr> "8", "14", "13", "7", "13", "22", "22", "9", "19",
"8", "17", "13", "19", "14", "15", "12", "21", "7", "19", "7", "5", "15", "15",
"…
$ startDay            <chr> "5", "1", "7", "4", "7", "2", "1", "2", "6", "3",
"5", "5", "7", "2", "7", "6", "3", "3", "2", "6", "3", "6", "3", "2", "7", "5",
"6…
$ startMonth          <chr> "4", "7", "7", "6", "7", "3", "6", "9", "8", "5",
"5", "10", "4", "5", "8", "6", "8", "9", "6", "8", "5", "9", "10", "9", "9",
"7", …
```

```
$ startDate           <chr> "2023-04-27", "2022-07-31", "2022-07-23",
"2023-06-28", "2022-07-16", "2023-03-27", "2023-06-18", "2022-09-12",
"2022-08-05", "2023-…
```

From this point forward, the record of my continued analysis, creation of visualization (and the visualizations themselves), business recommendations, and all other components of this case study are located in a second document, which is an RMarkdown document.

Please refer to this document for the continued analysis.

References

The work shown on the Kaggle page of “go.olga” (available here) in response to this same case study prompt and data set was useful in framing my approach to this data set.