Data Cleaning Report

The data on bike trips for fictional company Cyclistic has been made available by Motivate International Inc. under this license, granting users non-exclusive, royalty-free, limited, perpetual license to access, reproduce, analyze, copy, modify, distribute in a product or service and use the Data for any lawful purpose.

Motivate International Inc. is a bike share operator powering 8 bike share systems in the United States, including New York, Chicago, Washington, D.C., San Francisco, and Boston. The company was formed on 2009, it is headquartered in Brooklyn, NY. Motivate International Inc. runs Divvy bike-sharing program in Chicago.

Trip history data is stored by Divvy on Amazon cloud services - https://divvy-tripdata.s3.amazonaws.com/index.html as compressed zip files. Files 2013 and 2017 include 6 months of trip history data in each file. Starting Q1 2018 until Q1 2020 each zip file includes 3 months of data. From April 2020 to August 2021 data is organized monthly. Trip data is collected directly by Divvy thus as first-party internal data it is reliable and original. To address privacy concerns, the trip dataset was anonymized. Any personally identifying information was removed from the dataset. Instead, an original key is used to identify individual rides

To answer main business question of how casual and riders and members differ in service usage, we will be using 12 months of data from 2019. Analyzing entire year worth of data will help reduce sampling bias. 2019 was chosen as the target year for that reason as well. Due to the COVID 19 pandemic and government stay-at-home mandates, 2020 data will not be representative of a typical year of bike share usage.

Files from 2019 are broken down per quarter. To simplify the analysis, four datasets were merged into one table in Big Query application. This will simplify SQL queries and allow us to query the entire year at once to see patterns in the data.

Dataset is structured into 13 columns below. Each row represents one ride and includes information about when the ride started, ended and type of user. These pieces of data will be most detrimental to identify patterns is members and casual riders service usage.

Field name	**Type**
ride_id	STRING
rideable_type	STRING
started_at	TIMESTAMP
ended_at	TIMESTAMP
start_station_name	STRING
start_station_id	INTEGER
end_station_name	STRING
end_station_id	INTEGER
start_lat	FLOAT
start_lng	FLOAT
end_lat	FLOAT
end_lng	FLOAT
member_casual	STRING

Cleaning or manipulation of data

Data Cleaning Change log

Date	Affected files and columns	Change
09/22/2021	December 2020 station unique IDs	In December 2020 file station IDs did not match previous data. Used Vlookup to revise station IDs in December 2020 file to be consistent with the rest of the year
09/22/2021	2019 files column names	Column names throughout 2019 files were not consistent. Copied column names form Q1 2019 to the rest of the 2019 data
09/22/2021	Merging tables	Merged months and quarters files into one query table for each 2019 and 2020 years using "Append table" function in BigQuery. Named tables trip_data2019 and trip_data2020
09/24/2021	All records in 2019 dataset	Exclude records where variance between tripduration field and calculated duration is over 5 seconds
09/24/2021	Trip_data2019 start and end times	Extracted date and time from TIMASTAMP value and created start_date, start_time, end_date, end_time columns
09/24/2021	Trip_data2019	Adding day of the week for start and end times
09/24/2021	Trip_data2019	Adding weekday type for start and end times (start_weekday)
09/24/2021	Trip_data2019	Adding calculated tripduration column as end_time-start_time
09/24/2021	Trip_data2020 start and end times	Extracted date and time from TIMASTAMP value and created start_date, start_time, end_date, end_time columns
09/24/2021	Trip_data2020	Adding day of the week for start and end times
09/24/2021	Trip_data2020	Adding weekday type for start and end times (start_weekday)
09/24/2021	Trip_data2020	Adding calculated tripduration column as ended_at-started_at
	All records in 2020 dataset	Exclude records with negative trip duration calculation
	New locations table	Create a new table for location data
	All records in 2020 dataset	Exclude records with negative trip duration calculation

Determining data constraints for Cyclistic trip_data fields

Field name	Type	Constraint
trip_id	INTEGER	Values can't have a duplicate

start_time	TIMESTAMP	Values must fall between predefined maximum and minimum values Values must match a prescribed pattern Can't be NULL
end_time	TIMESTAMP	Values must fall between predefined maximum and minimum values Values must match a prescribed pattern Can't be NULL end_time is a larger value than start_time
bikeid	INTEGER	Does not contribute to the business task, will be excluded during cleaning
tripduration	FLOAT	Can't be NULL Values can't be negative
from_station_id	INTEGER	Does not contribute to the business task, will be excluded during cleaning
from_station_name	STRING	Does not contribute to the business task, will be excluded during cleaning
to_station_id	INTEGER	Does not contribute to the business task, will be excluded during cleaning
to_station_name	STRING	Does not contribute to the business task, will be excluded during cleaning
start_lat	FLOAT	Range between 40 and 43
start_lng	FLOAT	Range between -86 and -88
end_lat	FLOAT	Range between 40 and 43
end_lng	FLOAT	Range between -86 and -88
usertype	STRING	Two values: member or casual Can't be NULL
gender	STRING	Does not contribute to the business task, will be excluded during cleaning
birthyear	INTEGER	Does not contribute to the business task, will be excluded during cleaning

Data cleaning process

Checking for duplicate data in trip_id and ride_id

```
SELECT
    DISTINCT trip_id,
    COUNT(1) AS count_duplicates
FROM `infinite-badge-322819.Cyclistic.trip_data2019` as tripdata19
GROUP BY
    trip_id
ORDER BY
    count_duplicates DESC
```

2019 data does not contain duplicate values.

Using the same query for 2020 table we find duplicated values. The following query identified 209 duplicated ride_ids:

```
SELECT
    DISTINCT ride_id,
    COUNT(1) AS count_duplicates
```

```
FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
GROUP BY
    ride_id
HAVING
    count_duplicates=2
```

The next query returns the entire record for all duplicated values to check if they were duplicates. This query returns 418 records.

```
SELECT*
FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
JOIN (SELECT
    DISTINCT ride_id,
    COUNT(1) AS count_duplicates,
    FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
    GROUP BY
        ride_id
    ORDER BY
        count_duplicates=2) AS duplicates
ON
    tripdata20.ride_id = duplicates.ride_id
WHERE
    count_duplicates=2
ORDER BY
    tripdata20.ride_id DESC
```

It appears that duplicated ride_id entries appeared between December started_at and November stared_at. Furthermore, ended_at values are smaller for December start time, which is impossible. Without access to the data engineers collecting the data to verify if these entries were correct, we will remove the entries where the value of ended_at is smaller than the value of started_at

With the following query we can verify if all duplicated values from December also have negative trip duration. This query returns 211 rows, 209 from December and 2 from November. Thus, duplicate issue will be resolved after removing rows with negative trip duration

```
SELECT
    *,
    TIMESTAMP_DIFF(ended_at, started_at, second) AS duration
FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
JOIN (SELECT
    DISTINCT ride_id,
    COUNT(1) AS count_duplicates,
    FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
    GROUP BY
        ride_id
    ORDER BY
        count_duplicates=2) AS duplicates
ON
    tripdata20.ride_id = duplicates.ride_id
WHERE
    count_duplicates=2
    AND TIMESTAMP_DIFF(ended_at, started_at, second) < 0
ORDER BY
    duration ASC
```

Checking start and end time, tripduration values

Using the following query, we will review the range of start and end times to make sure they fall into correct timeframe.

```
SELECT
    MIN(start_time) as min_start,
    MAX(start_time) as max_start,
    MIN(end_time) as min_end,
    MAX(end_time) as max_end
FROM `infinite-badge-322819.Cyclistic.trip_data2019` as tripdata19
```

Maximum end time for 2019 table is 01/21/2020. Using the query below we will review the records so make sure that values with end time in 2020 are applicable to the dataset

```
SELECT
    *
FROM `infinite-badge-322819.Cyclistic.trip_data2019` as tripdata19
WHERE
    end_time > '2020-01-01'
```

The query returned 29 values with start time ranging from 10/06/2019 to 12/31/2019. Without being able to verify this data with data engineers, we will treat this data as outliers and keep it in the dataset

Using same queries for 2020 dataset, we only see 17 values with ended time in 2021. We will treat these values the same way as in the 2019 dataset.

The next step of the cleaning process is to make sure all the timestamp data is valid and tripduration column equals the time difference between end_time and start_time using the following query

```
SELECT
    CAST(tripduration AS int64) AS tripduration,
    TIMESTAMP_DIFF(end_time, start_time, second) AS duration_calculation,
    tripduration-TIMESTAMP_DIFF(end_time, start_time, second) AS var
FROM `infinite-badge-322819.Cyclistic.trip_data2019` as tripdata19
WHERE
    tripduration-TIMESTAMP_DIFF(end_time, start_time, second) > 0
```

After running this query, we find that there are 80 values where calculated trip duration does not equal tripduration column. Out of 80 values, 19 have very low variances between 1 and 4 seconds, which are likely due to rounding for original field calculations. Remaining 61 values have variance of 1 hour. We hypothesize that end_time column was reduced by 1 hour which caused this error

Using this query, we can review records where tripduration column did not equal the calculation between the end_time and start_time. It also adds new columns, increasing end_time by 1 hour (new_end_time) and calculating the variance between trip duration and start_time and new_end_time to check our hypothesis

```
WITH mismatched_duration AS (

SELECT
    trip_id, start_time, end_time, tripduration,
    TIMESTAMP_DIFF(end_time, start_time, second) AS calculation,
CASE
    WHEN tripduration-TIMESTAMP_DIFF(end_time, start_time, second)>5
    THEN TIMESTAMP_ADD(end_time, INTERVAL 1 HOUR)
    ELSE end_time
END AS new_end_time,
```

```
FROM `infinite-badge-322819.Cyclistic.trip_data2019` as tripdata19
WHERE
    tripduration-TIMESTAMP_DIFF(end_time, start_time, second)>5
)
SELECT
    trip_id, start_time, end_time, tripduration, new_end_time,
    TIMESTAMP_DIFF(end_time, start_time, second) AS calculation,
    TIMESTAMP_DIFF(new_end_time, start_time, second) AS newcalculation,
    TIMESTAMP_DIFF(new_end_time, start_time, second)-tripduration AS new_var
FROM mismatched_duration
```

The query returns 61 values with new variance column values being between 0 and 1. It is likely that due to data collection error end_time was reduced by 1 hour for these 61 values. However, without being able to verify the values with data engineering team, we will not be using these values for our analysis and exclude records where variance between tripduration field and calculated duration is over 5 seconds.

As 2020 data does not include tripduration column, we are unable to verify the started_at and ended_at column values in the same way. Below query calculates trip duration and returns negative values

```
SELECT
    ride_id, started_at, ended_at,
    TIMESTAMP_DIFF(ended_at, started_at, second) AS duration_calculation,
FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
WHERE
    TIMESTAMP_DIFF(ended_at, started_at, second) < 0
```

The query returns 8,405 values, which include ride_id duplicates we previously found.

Using the query below we will identify the percent of negative values in each start month's total number of rides. Largest percent of errors was in October – 0.49%. Considering that our population sizes for each month are large, we can still get reliable results from our analysis that are within 99% confidence level and 1% margin of error even if negative values are removed. Given that we do not have access to data engineers and are unable to verify the accuracy of records in question, we will remove all negative values from 2020 dataset

```
WITH total_negatives AS
(WITH negative_duration AS
(SELECT
    ride_id, started_at, ended_at,
    TIMESTAMP_DIFF(ended_at, started_at, second) AS duration_calculation,
FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
WHERE
    TIMESTAMP_DIFF(ended_at, started_at, second) < 0
)
SELECT
    DISTINCT(EXTRACT(month FROM started_at)) AS start_month,
    COUNT(EXTRACT(month FROM started_at)) AS number_negatives
FROM negative_duration
GROUP BY
    start_month
ORDER BY
    start_month
)
SELECT
    start_month,
    number_negatives,
    total_rides,
    ROUND((number_negatives/total_rides)*100,2) AS percent_of_negatives
```

```
FROM total_negatives
JOIN (SELECT
    DISTINCT(EXTRACT(month FROM started_at)) AS original_start_month,
    COUNT(EXTRACT(month FROM started_at)) AS total_rides
    FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
    GROUP BY original_start_month) AS original_data
ON
    total_negatives.start_month=original_data.original_start_month
```

Using the following queries, we will check if start and end times include any NULL records:

```
SELECT
    *
FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
WHERE
    started_at IS NULL
    OR ended_at IS NULL


SELECT
    *
FROM `infinite-badge-322819.Cyclistic.trip_data2019` as tripdata19
WHERE
    start_time IS NULL
    OR end_time IS NULL
```

Both queries returned no values, which indicates that all records in both 2019 and 2020 datasets have start and end times.

Checking lat and lng values

Latitude values for Chicago area should be between 40 and 43 and longitude values between Range between -86 and -88. 2019 dataset does not include location data, so we will only be checking 2020 dataset. Using the query below we will review maximum and minimum values for latitude and longitude. All latitude and longitude fall within the criteria.

```
SELECT
    MIN(start_lat) AS min_start_lat,
    MAX(start_lat) AS max_start_lat,
    MIN(start_lng) AS min_start_lng,
    MAX(start_lng) AS max_start_lng
FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
```

Latitude and longitude will not be required to answer the main business question, however they might help illustrate if there is a location type difference between members and casual riders. A separate table will be created using latitude and longitude values to create map visualization in Tableau. Using below query, we will check for NULL values in lan and lng columns to later have them excluded from the dataset. The query returned 4,255 values.

```
SELECT
    *
FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
WHERE
    start_lat IS NULL
    OR start_lng IS NULL
    OR end_lat IS NULL
    OR end_lng IS NULL
```

Checking usertype values

There should be two user types and no values can be null. Using the query below we will check the values in usertype column. This query returns two values for both 2019 and 2020 datasets.

```
SELECT
    DISTINCT(usertype)
FROM `infinite-badge-322819.Cyclistic.trip_data2019` as tripdata19
```

The next query will check for NULL values. It returned no NULL values for both 2019 and 2020 datasets.

```
SELECT
    *
FROM `infinite-badge-322819.Cyclistic.trip_data2020` as tripdata20
WHERE
    member_casual is NULL
```

trip_data2019 cleaning query

```
--Cleaning query for trip_data2019 table

SELECT
    trip_id,
    EXTRACT(date FROM start_time) AS start_date, --
Extracting date, time, and day of the week into separate columns for future analysis
    EXTRACT(time FROM start_time) AS start_time,
    FORMAT_DATE("%A",start_time) AS start_day,
CASE --Adding weekday type to distinguish between the weekend and weekday
    WHEN FORMAT_DATE("%A",start_time)="Saturday" OR FORMAT_DATE("%A",start_time)="Sunday"
    THEN "Weekend"
    ELSE "Weekday"
END AS start_weekday,
    EXTRACT(date FROM end_time) AS end_date,
    EXTRACT(time FROM end_time) AS end_time,
    FORMAT_DATE("%A",end_time) AS end_day,
CASE --Adding weekday type to distinguish between the weekend and weekday
    WHEN FORMAT_DATE("%A",end_time)="Saturday" OR FORMAT_DATE("%A",end_time)="Sunday"
    THEN "Weekend"
    ELSE "Weekday"
END AS end_weekday,
    TIMESTAMP_DIFF(end_time, start_time, second) AS tripduration, --
Using calculation field for trip duration column to ensure accuracy
CASE --Renaming the user types to stay consistent with the business task
    WHEN usertype="Customer"
    THEN "casual"
    ELSE "member"
END AS user_type
FROM `infinite-badge-322819.Cyclistic.trip_data2019` AS trip_data2019
WHERE
    tripduration-TIMESTAMP_DIFF(end_time, start_time, second)<5 --
Excluding records that have over 5 seconds variance between the calculated trip lenght and tripduration colum
n
```

trip_data2020 cleaning query

```
--Cleaning query for trip_data2020 table

SELECT
    ride_id AS trip_id, --Renaming the column to stay consistent with 2019 cleaned dataset
    EXTRACT(date FROM started_at) AS start_date, --
Extracting date, time, and day of the week into separate columns for future analysis
    EXTRACT(time FROM started_at) AS start_time,
    FORMAT_DATE("%A",started_at) AS start_day,
CASE --Adding weekday type to distinguish between the weekend and weekday
    WHEN FORMAT_DATE("%A",started_at)="Saturday" OR FORMAT_DATE("%A",started_at)="Sunday"
    THEN "Weekend"
    ELSE "Weekday"
END AS start_weekday,
    EXTRACT(date FROM ended_at) AS end_date,
    EXTRACT(time FROM ended_at) AS end_time,
    FORMAT_DATE("%A",ended_at) AS end_day,
CASE --Adding weekday type to distinguish between the weekend and weekday
    WHEN FORMAT_DATE("%A",ended_at)="Saturday" OR FORMAT_DATE("%A",ended_at)="Sunday"
    THEN "Weekend"
    ELSE "Weekday"
END AS end_weekday,
    TIMESTAMP_DIFF(ended_at, started_at, second) AS tripduration, --
Using calculation field for trip duration column to ensure accuracy
    member_casual AS user_type
FROM `infinite-badge-322819.Cyclistic.trip_data2020` AS trip_data2020
WHERE
    TIMESTAMP_DIFF(ended_at, started_at, second)>0 --
Excluding records that have negative calculated trip duration
```

Station locations

To create a map of trips per station, the following query was set-up. After running the query, I realized that the same station had different location information in the data.

```
SELECT
    start_station_name,
    start_station_id,
    CONCAT(start_lat, ',', start_lng) AS start_location,
    end_station_name,
    end_station_id,
    CONCAT(end_lat, ',', end_lng) AS end_location
FROM `infinite-badge-322819.Cyclistic.trip_data2020`
WHERE
    start_station_name IS NOT NULL
    AND start_station_id IS NOT NULL
    AND end_station_name IS NOT NULL
    AND end_station_id IS NOT NULL
```

To mitigate the issue, Divvy station information JSON feed was used, where longitude and latitude data was extracted. Afterwards, the following queries were ran to get collect information on the number of trips started and ended in each station

```
SELECT
    DISTINCT(end_station_name),
    end_station_id,
```

```
    member_casual,
    COUNT(*) AS trips_started
FROM `infinite-badge-322819.Cyclistic.trip_data2020`
GROUP BY
    end_station_name,
    end_station_id,
    member_casual
HAVING
    end_station_name IS NOT NULL
    AND  end_station_id IS NOT NULL
```

Query results were joined with the JSON feed using Excel and vlookup.

Reviewing clean data

After original datasets were cleaned, two new tables were created: clean_trips2019 and clean_trips_2020. Both tables now follow the same schema:

Field name	Description	Restrictions
trip_id	Unique ID for each trip in the dataset	Can't duplicate, can't be NULL
start_date	Date when trip started	Can't be NULL
start_time	Time when trip started	Can't be NULL
start_day	Day of the week when trip started	Can't be NULL
start_weekday	Weekday or weekend start	Can't be NULL
end_date	Date when trip ended	Can't be NULL
end_time	Time when trip ended	Can't be NULL
end_day	Day of the week when trip ended	Can't be NULL
end_weekday	Weekday or weekend end	Can't be NULL
tripduration	Duration of the trip	Can't be negative, can't be NULL
user_type	Casual or Member user	Can't be NULL

All steps described above were repeated for cleaned tables to doublecheck the process and make sure the values follow restrictions