Cyclistic Data Cleaning and Exploration with SQL Queries

```sql
/*

Cyclistic Bike Data Cleaning and Exploration in SQL Queries

Skills Used: Temporary Table, CTE, Aggregate Functions, Window Function, Timestamp Function, Conditional Function, Subquery

Platform: GCP BigQuery

*/


---------------------------------------------------------------------------------------------

-- DATA CLEANING IN SQL QUERIES


-- Preview of the four tables

SELECT
    *
FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q1` LIMIT 20;

SELECT
    *
FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q2` LIMIT 20;

SELECT
    *
FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q3` LIMIT 20;

SELECT
    *
FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q4` LIMIT 20;


-- Change DivvyTrips2019Q2 column names to same as Q1, Q3 and Q4 using temp table

BEGIN
CREATE OR REPLACE TEMPORARY TABLE DivvyTrips2019Q2Temp
AS
SELECT
    _01___Rental_Details_Rental_ID AS trip_id,
    _01___Rental_Details_Local_Start_Time AS start_time,
    _01___Rental_Details_Local_End_Time AS end_time,
    _01___Rental_Details_Bike_ID AS bikeid,
    _01___Rental_Details_Duration_In_Seconds_Uncapped AS tripduration,
    _03___Rental_Start_Station_ID AS from_station_id,
    _03___Rental_Start_Station_Name AS from_station_name,
    _02___Rental_End_Station_ID AS to_station_id,
    _02___Rental_End_Station_Name AS to_station_name,
    User_Type AS usertype,
    Member_Gender AS gender,
    _05___Member_Details_Member_Birthday_Year AS birthyear

FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q2`;
End;


-- Temp table DivvyTrips2019Q2Temp stored as causal-bison-323215._script2cb74db8e872be8c7189bb08b16c7d5c3fba71e6.DivvyTrips2019Q2Temp

SELECT
    *
FROM
    `causal-bison-323215._script2cb74db8e872be8c7189bb08b16c7d5c3fba71e6.DivvyTrips2019Q2Temp` LIMIT 10


-- Merge DivvyTrips2019Q, DivvyTrips2019Q2Temp, DivvyTrips2019Q3 and DivvyTrips2019Q4 into one table

BEGIN
CREATE OR REPLACE TEMP TABLE DivvyTrips2019
AS
SELECT
    *
FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q1`
UNION ALL
SELECT
    *
FROM
    `causal-bison-323215._script2cb74db8e872be8c7189bb08b16c7d5c3fba71e6.DivvyTrips2019Q2Temp`
UNION ALL
SELECT
```

```sql
    *
FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q3`
UNION ALL
SELECT
    *
FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q4`;
END;


-- Temp table DivvyTrips2019 stored`causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`

SELECT
    *
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`


-- Alternatively, one could create a CTE to rename the columns in DivvyTrips2019Q2, union the CTE with the other
three tables and save the result into a new table, DivvyTrips2019V2

WITH NewDivvyTrips2019Q2 AS (SELECT
    _01___Rental_Details_Rental_ID AS trip_id,
    _01___Rental_Details_Local_Start_Time AS start_time,
    _01___Rental_Details_Local_End_Time AS end_time,
    _01___Rental_Details_Bike_ID AS bikeid,
    _01___Rental_Details_Duration_In_Seconds_Uncapped AS tripduration,
    _03___Rental_Start_Station_ID AS from_station_id,
    _03___Rental_Start_Station_Name AS from_station_name,
    _02___Rental_End_Station_ID AS to_station_id,
    _02___Rental_End_Station_Name AS to_station_name,
    User_Type AS usertype,
    Member_Gender AS gender,
    _05___Member_Details_Member_Birthday_Year AS birthyear

FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q2`
)
SELECT
    *
FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q1`
UNION ALL
SELECT
    *
FROM
    NewDivvyTrips2019Q2
UNION ALL
SELECT
    *
FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q3`
UNION ALL
SELECT
    *
FROM
    `causal-bison-323215.Cyclistic.DivvyTrips2019Q4`;


-- Check if the primary key column, trip-id, has duplicate value

SELECT
    COUNT(DISTINCT trip_id) AS unique_count_tripId,
    COUNT(*) AS total_number_rows
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`


-- Check if columns contained nulls

SELECT
    COUNTIF(trip_id IS NULL) AS trip_id,
    COUNTIF(start_time IS NULL) AS start_time,
    COUNTIF(end_time IS NULL) AS end_time,
    COUNTIF(tripduration IS NULL) AS tripduration,
    COUNTIF(bikeid IS NULL) AS bikeid,
    COUNTIF(from_station_id IS NULL) AS from_station_id,
    COUNTIF(from_station_name IS NULL) AS from_station_name,
    COUNTIF(to_station_id IS NULL) AS to_station_id,
    COUNTIF(to_station_name IS NULL) AS to_station_name,
    COUNTIF(usertype IS NULL) AS usertype,
    COUNTIF(gender IS NULL) AS gender,
    COUNTIF(birthyear IS NULL) AS birthyear
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`;
```

```sql
-- Examine the birthyear column

SELECT
    MIN(birthyear) AS min_birthyear,
    MAX(birthyear) AS max_birthyear
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`;


-- Update the gender column; replace nulls with 'Unknown'

UPDATE `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
SET gender = 'Unknown'
WHERE gender IS NULL;

SELECT
    *
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
WHERE
    gender IS NULL;


-- Create and populate colum ride_length

ALTER TABLE `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
ADD COLUMN ride_length INTEGER;

UPDATE `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
SET ride_length = TIMESTAMP_DIFF(end_time, start_time, SECOND)
WHERE trip_id IS NOT NULL;

SELECT
    MIN(ride_length) AS min_ride_duration,
    MAX(ride_length) AS max_ride_duration
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`;

SELECT
    COUNT(*) AS rows_with_negative_ride_length,
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
WHERE
    ride_length <= 0;


-- Delete rows with negative ride_length value

DELETE
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
WHERE
    ride_length <= 0;

SELECT
    COUNT(*) AS rows_with_negative_ride_length,
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
WHERE
    ride_length <= 0;

SELECT
    MIN(ride_length) AS min_ride_duration,
    MAX(ride_length) AS max_ride_duration
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`;


-- Create and populate day_of_week column

ALTER TABLE `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
ADD COLUMN day_of_week INTEGER;

UPDATE `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
SET day_of_week = EXTRACT(DAYOFWEEK FROM start_time)
WHERE trip_id IS NOT NULL;

SELECT
    *
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019` LIMIT 10;

SELECT
    MIN(day_of_week) AS min_day_of_week,
    MAX(day_of_week) AS max_day_of_week
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`;
```

```sql
-- Create and populate day column

ALTER TABLE `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
ADD COLUMN day STRING;

UPDATE `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
SET day = CASE day_of_week
    WHEN 1 THEN 'Sunday'
    WHEN 2 THEN 'Monday'
    WHEN 3 THEN 'Tuesday'
    WHEN 4 THEN 'Wednesday'
    WHEN 5 THEN 'Thursday'
    WHEN 6 THEN 'Friday'
    WHEN 7 THEN 'Saturday'
    ELSE NULL
    END
WHERE trip_id IS NOT NULL;

SELECT
    *
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019` LIMIT 10;

SELECT
    COUNTIF(day IS NULL) AS count
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`;



-------------------------------------------------------------------------------------------

-- EXPLORATORY DATA ANALYSIS WITH SQL QUERIES


-- Different types of usertype and their breakdown by count and gender

SELECT
    usertype AS user,
    gender AS gender,
    COUNT(*) AS number
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
GROUP BY usertype, gender
ORDER BY usertype, gender;


-- Average, lowest and highest ride duration, number of trips and busiest day of 2019

SELECT
    ROUND(AVG(ride_length), 2) AS average_ride_duration_sec,
    MIN(ride_length) AS lowest_ride_duration_sec,
    MAX(ride_length) AS highest_ride_duration_sec,
    COUNT(trip_id) AS total_trip,
    APPROX_TOP_COUNT(day, 1) AS busiest_day
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`;


-- Average, lowest and highest ride duration, number of trips and busiest day for each type of user

SELECT
    usertype AS user,
    ROUND(AVG(ride_length), 2) AS average_ride_duration_sec,
    MIN(ride_length) AS lowest_ride_duration_sec,
    MAX(ride_length) AS highest_ride_duration_sec,
    COUNT(trip_id) AS total_trip,
    APPROX_TOP_COUNT(day, 1) AS busiest_day
FROM
    `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
GROUP BY usertype;


-- Proportion of rides above average ride duration for each type of user

SELECT
    usertype AS user,
    ROUND(AVG(ride_length), 2) AS avg_ride_duration_sec,
    COUNTIF((ride_length - avg_ride_duration) > 0) AS ride_above_avg_duration,
    COUNT(trip_id) AS total_trip,
    ROUND(COUNTIF((ride_length - avg_ride_duration) > 0)/COUNT(trip_id)*100, 2) AS percent_ride_above_avg_duration,
FROM (
    SELECT
    usertype,
    ride_length,
    trip_id,
    AVG(ride_length) OVER (
    PARTITION BY usertype
```

```sql
        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
        AS avg_ride_duration,
        FROM
        `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`)
    GROUP BY usertype;


    -- Average ride duration for users by the day of the week

    SELECT
        usertype AS user,
        day AS day,
        ROUND(AVG(ride_length), 2) AS avg_ride_duration,
    FROM
        `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
    GROUP BY usertype, day
    ORDER BY usertype;


    -- Number of rides for users by day of the week

    SELECT
        usertype AS user,
        day AS day,
        COUNT(trip_id) AS total_trip
    FROM
        `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
    GROUP BY usertype, day
    ORDER BY usertype;


    -- Average ride duration for users by the day of the week by quarter

    SELECT
        usertype AS user,
        EXTRACT(QUARTER FROM start_time) AS quarter,
        day AS day,
        ROUND(AVG(ride_length), 2) AS avg_ride_duration
    FROM
        `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
    GROUP BY usertype, quarter, day
    ORDER BY usertype, quarter;


    -- Number of rides for users by day of the week by quarter

    SELECT
        usertype AS user,
        EXTRACT(QUARTER FROM start_time) AS quarter,
        day AS day,
        COUNT(trip_id) AS total_trip
    FROM
        `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`
    GROUP BY usertype, quarter, day
    ORDER BY usertype, quarter;


    -- Save temporary table as BigQuery table for export in order to visualize with Tableau

    SELECT
        trip_id,
        start_time,
        end_time,
        bikeid,
        from_station_id,
        from_station_name,
        to_station_id,
        to_station_name,
        usertype,
        ride_length,
        day
    FROM
        `causal-bison-323215._script203775267b33d1a1aef3c416b8f33c8849058ea0.DivvyTrips2019`;
```