```sql
-- Changing the names of the columns in 2019_q2 table in order to match with the other tables in the dataset
ALTER TABLE `data-analytics-project-2205.Cyclist_Case_Study.2019_q2`
RENAME COLUMN _01___Rental_Details_Rental_ID TO trip_id,
RENAME COLUMN _01___Rental_Details_Local_Start_Time TO start_time,
RENAME COLUMN _01___Rental_Details_Local_End_Time TO end_time,
RENAME COLUMN _01___Rental_Details_Bike_ID TO bikeid,
RENAME COLUMN _01___Rental_Details_Duration_In_Seconds_Uncapped TO tripduration,
RENAME COLUMN _03___Rental_Start_Station_ID TO from_station_id,
RENAME COLUMN _03___Rental_Start_Station_Name TO from_station_name,
RENAME COLUMN _02___Rental_End_Station_ID TO to_station_id,
RENAME COLUMN _02___Rental_End_Station_Name TO to_station_name,
RENAME COLUMN User_Type TO usertype,
RENAME COLUMN Member_Gender TO gender,
RENAME COLUMN _05___Member_Details_Member_Birthday_Year TO birthyear;

-- Combining all quarters and saving them into a new table to make a table with whole year's data
SELECT * FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_q1`
UNION ALL
SELECT * FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_q2`
UNION ALL
SELECT * FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_q3`
UNION ALL
SELECT * FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_q4`
-- Created a new table named 2019_full_year

Select count(*) FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year` AS cyclist
WHERE usertype = 'Subscriber' -- Total subscriber rentals are 2,937,367
```

```sql
Select count(*) FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year` AS cyclist
WHERE usertype = 'Customer' -- Total non-subscriber rentals are 880,637

-- Checking the integrity of data to see if there are any unusual numbers
SELECT
min(from_station_id) as Minimum,
Max(from_station_id) as Maximum,
Min(to_station_id) as Minimum1,
Max(to_station_id) as Maximum1
FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year` -- All seems fine

SELECT --extract(Month from date(start_time)) as rental_month,
min(extract(Month from date(start_time))) AS min_month,
max(extract(Month from date(start_time))) AS max_month
FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year` -- All seems fine

SELECT
min(tripduration),
max(tripduration)
FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year` -- Longest trip is 2952 hours. Needs to be investigated

SELECT count(*) FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year`
WHERE tripduration >= 86400 -- There are 1849 (0.0048% of total records) trips that are longer than a day

-- I delete the records of multiple-day rents for the purpose of this analysis as they can be an error in registry or people
failing to return the bike.
DELETE FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year`
WHERE tripduration >= 86400
```

```sql
SELECT
2019 - min(birthyear) AS oldest,
2019 - max(birthyear) AS youngest
FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year` -- According to the data, oldest client is 260 years old and youngest is 5 years old. Need to clean it

-- I delete the records for people over the age of 80 and below the minimum allowed age of 16
DELETE FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year`
WHERE birthyear > 2003 OR birthyear < 1939

SELECT * FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year`
WHERE NOT gender = 'Female' AND NOT gender = 'Male' AND gender IS NOT NULL -- all seems fine

SELECT * FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year`
WHERE NOT usertype = 'Customer' AND NOT usertype = 'Subscriber' AND usertype IS NOT NULL -- all seems fine

SELECT DISTINCT(trip_id)
--SELECT trip_id
FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year` -- No duplicate IDs

-- Data is ready for the analysis

-- Calculating the average minutes spent on rent each day of the week by customers and subscribers to see if they have any different habits

SELECT
days.Day as day_of_the_week,
```

```sql
usertype,
round(avg(average_minutes), 1) AS average
FROM ( -- creating a subquery to save date from start_time so I can join it with days of the week

SELECT
date(start_time) as date,
round(avg(tripduration)/ 60,1) as average_minutes, -- It may not seem neccessary to get the calculation here but I want to save
it like this in case I want to use this line again
usertype,
gender,
birthyear
FROM `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year`
GROUP BY date,usertype, gender, birthyear
ORDER BY date
) sub

INNER JOIN `data-analytics-project-2205.Cyclist_Case_Study.days_of_2019` as days
ON sub.date = days.Date -- Joining days of the week so I can analyze for different days and group by the days later on

GROUP BY 1, 2
HAVING usertype = 'Customer' --'Subscriber'
ORDER BY
     CASE
          WHEN Day = 'Monday' THEN 1
          WHEN Day = 'Tuesday' THEN 2
          WHEN Day = 'Wednesday' THEN 3
          WHEN Day = 'Thursday' THEN 4
          WHEN Day = 'Friday' THEN 5
```

```
          WHEN Day = 'Saturday' THEN 6
          WHEN Day = 'Sunday' THEN 7
     END ASC -- I used the case here so the days are not ordered alphabetically


-- Now calculating the number of trips that happened each day by customers and subscribers.

SELECT
days.Day as day_of_the_week,
usertype,
sum(total_trips) AS num_of_trips
FROM (

SELECT
date(start_time) as date,
count(trip_id) AS total_trips,
usertype,
gender

FROM `Cyclist_Case_Study.2019_full_year`
GROUP BY 1,3,4
) sub

INNER JOIN `Cyclist_Case_Study.days_of_2019` as days
ON sub.date = days.Date

WHERE usertype = 'Customer' -- 'Subscriber'
GROUP BY 1,2
```

```sql
ORDER BY
     CASE
          WHEN Day = 'Monday' THEN 1
          WHEN Day = 'Tuesday' THEN 2
          WHEN Day = 'Wednesday' THEN 3
          WHEN Day = 'Thursday' THEN 4
          WHEN Day = 'Friday' THEN 5
          WHEN Day = 'Saturday' THEN 6
          WHEN Day = 'Sunday' THEN 7
     END ASC



-- At this point I have decided to standardize the date format and make a column for age to make it easier for further analysis

ALTER TABLE `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year`
ADD column date date

UPDATE `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year`
set date = date(start_time)
where date is null

ALTER TABLE `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year`
ADD column age int

UPDATE `data-analytics-project-2205.Cyclist_Case_Study.2019_full_year`
set age = 2019 - birthyear
```

```sql
where age is null

SELECT count(*)
FROM `Cyclist_Case_Study.2019_full_year`
where age is null

-------------------------------------
--Next up, I will analyze how age plays different role in customers and subscribers habits

SELECT
COUNT(trip_id) as num_of_trips,
CASE
     WHEN age <20 then "16-19"
     WHEN age between 20 and 29 then "20s"
     WHEN age between 30 and 39 then "30s"
     WHEN age between 40 and 49 then "40s"
     WHEN age between 50 and 59 then "50s"
     WHEN age between 60 and 69 then "60s"
     WHEN age between 70 and 79 then "70s"
     ELSE '80+'
END AS age_bracket
FROM `Cyclist_Case_Study.2019_full_year`
WHERE age is not null
GROUP BY age_bracket
ORDER BY age_bracket

-- Now I will run the same code for both subscribers and customers to find patterns
```

```sql
SELECT
COUNT(trip_id) as num_of_trips,
CASE
     WHEN age <20 then "16-19"
     WHEN age between 20 and 29 then "20s"
     WHEN age between 30 and 39 then "30s"
     WHEN age between 40 and 49 then "40s"
     WHEN age between 50 and 59 then "50s"
     WHEN age between 60 and 69 then "60s"
     WHEN age between 70 and 79 then "70s"
     ELSE '80+'
END AS age_bracket
FROM `Cyclist_Case_Study.2019_full_year`
WHERE age is not null AND usertype = 'Subscriber' --'Customer'
GROUP BY age_bracket
ORDER BY age_bracket

-- Checking the difference between customers and subscribers in different months

SELECT
EXTRACT(month from date) AS month,
usertype,
count(trip_id) AS num_of_trips
FROM `Cyclist_Case_Study.2019_full_year`
GROUP BY month, usertype
ORDER BY month, usertype ASC

-- Checking the difference between customers and subscribers in different seasons
```

```sql
SELECT
CASE
     WHEN EXTRACT(month from date) between 3 AND 5 THEN "spring"
     WHEN EXTRACT(month from date) between 6 AND 8 THEN "summer"
     WHEN EXTRACT(month from date) between 9 AND 11 THEN "autumn"
ELSE "winter" END as seasons,
usertype,
count(trip_id) AS num_of_trips
FROM `Cyclist_Case_Study.2019_full_year`
GROUP BY seasons, usertype
ORDER BY
CASE
     WHEN seasons = "spring" THEN 1
     WHEN seasons = "summer" THEN 2
     WHEN seasons = "autumn" THEN 3
     WHEN seasons = "winter" THEN 4
END, usertype ASC

-- Checking the difference between customers and subscribers in different hours of the days

SELECT
usertype,
COUNT (*) AS trips,
extract(hour from start_time) AS hour
FROM `Cyclist_Case_Study.2019_full_year`
GROUP BY 1,3
ORDER BY hour
```

```
------------

SELECT
CASE
     WHEN extract(hour from start_time) between 5 and 10 THEN "morning"
     WHEN extract(hour from start_time) between 11 and 16 THEN "afternoon"
     WHEN extract(hour from start_time) between 17 and 22 THEN "evening"
ELSE "night" END as part_of_day,
usertype,
count(trip_id)

FROM `Cyclist_Case_Study.2019_full_year`
GROUP BY 1,2
ORDER BY
CASE
     WHEN part_of_day = "morning" THEN 1
     WHEN part_of_day = "afternoon" THEN 2
     WHEN part_of_day = "evening" THEN 3
     WHEN part_of_day = "night" THEN 4
END, usertype ASC

-- Finally I want to check the stations

SELECT
from_station_name,
count(*) as num_of_trips
FROM `Cyclist_Case_Study.2019_full_year`
```

```
GROUP BY from_station_name
ORDER BY num_of_trips DESC

-- top 3 routes taken by both user types

SELECT
concat(from_station_name, " TO ", to_station_name) AS route,
count(*) as num_of_trips,
round(avg(tripduration)/60, 1) as avg_minutes
FROM `Cyclist_Case_Study.2019_full_year`
WHERE usertype = "Subscriber"
GROUP BY route
ORDER BY num_of_trips DESC
LIMIT 3

SELECT
concat(from_station_name, " TO ", to_station_name) AS route,
count(*) as num_of_trips,
round(avg(tripduration)/60, 1) as avg_minutes
FROM `Cyclist_Case_Study.2019_full_year`
WHERE usertype = "Customer"
GROUP BY route
ORDER BY num_of_trips DESC
LIMIT 3

-- Average minutes of rental by user type

SELECT
```

```sql
usertype,
round(avg(tripduration)/60, 1) as avg_minutes
FROM `Cyclist_Case_Study.2019_full_year`
GROUP BY usertype

--top 3 start point of customers

SELECT
from_station_name AS station,
count (*) num_of_trips
FROM `Cyclist_Case_Study.2019_full_year`
WHERE usertype = 'Customer'
GROUP BY station
ORDER BY num_of_trips DESC
LIMIT 10

--Checking the new data I found with locations of the stations to use in a map

SELECT distinct(ID)
FROM `Cyclist_Case_Study.stations` -- no duplicate id. It can be used as primary key to join with the main data

SELECT
A.usertype,
A.from_station_name,
count(*) as trips,
B.Latitude,
B.Longitude
FROM `Cyclist_Case_Study.2019_full_year` AS A
```

```
JOIN `Cyclist_Case_Study.stations` AS B
ON A.from_station_id = B.ID
Where from_station_name LIKE '%Streeter%'
GROUP BY 1,2,4,5
```